

## CS User's Guide

Generated by Doxygen 1.8.17

---

<b>1 CFS Checksum (CS) Documentation</b>	<b>1</b>
1.1 CFS Checksum Introduction . . . . .	2
1.2 CFS Checksum Overview . . . . .	3
1.3 CFS Checksum Operation . . . . .	4
1.4 CFS Checksum Commands . . . . .	4
1.5 CFS Checksum Telemetry . . . . .	4
1.6 CFS Checksum Events . . . . .	4
1.7 CFS Checksum Deployment Guide . . . . .	5
1.8 CFS Checksum Configuration . . . . .	5
1.9 CFS Checksum Table Definitions . . . . .	5
1.10 CFS Checksum Constraints . . . . .	6
1.11 CFS Checksum Frequently Asked Questions . . . . .	7
<b>2 Core Flight Executive Documentation</b>	<b>7</b>
2.1 Background . . . . .	9
2.1.1 Core Flight Executive (cFE) Goals . . . . .	9
2.2 Applicable Documents . . . . .	10
2.3 Version Numbers . . . . .	10
2.3.1 Version Number Semantics . . . . .	10
2.3.2 How and Where Defined . . . . .	11
2.3.3 Identifying Development Builds . . . . .	11
2.3.4 Templates for the short and long version string . . . . .	11
2.4 Dependencies . . . . .	12
2.5 Acronyms . . . . .	12
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 . . . . .	16
2.6.5 Application and Child Task Management . . . . .	17
2.6.6 Starting an Application . . . . .	17
2.6.7 Stopping an Application . . . . .	17
2.6.8 Restarting an Application . . . . .	18
2.6.9 Reloading an Application . . . . .	18
2.6.10 Listing Current Applications . . . . .	18
2.6.11 Listing Current Tasks . . . . .	19
2.6.12 Loading Common Libraries . . . . .	19
2.6.13 Basic File System . . . . .	19
2.6.14 Performance Data Collection . . . . .	20

2.6.15 Critical Data Store . . . . .	21
2.6.16 Memory Pool . . . . .	21
2.6.17 System Log . . . . .	24
2.6.18 Version Identification . . . . .	24
2.6.19 Frequently Asked Questions about Executive Services . . . . .	24
2.7 cFE Executive Services Commands . . . . .	24
2.8 cFE Executive Services Telemetry . . . . .	26
2.9 cFE Executive Services Configuration Parameters . . . . .	26
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 . . . . .	34
2.10.5 EVS Registry . . . . .	35
2.10.6 EVS Counters . . . . .	35
2.10.7 Resetting EVS Counters . . . . .	36
2.10.8 Effects of a Processor Reset on EVS . . . . .	36
2.10.9 EVS squelching of misbehaving apps . . . . .	37
2.10.10 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 . . . . .	42
2.14.2 Autonomous Actions . . . . .	43
2.14.3 Operation of the SB Software . . . . .	44
2.14.4 Frequently Asked Questions about Software Bus . . . . .	47
2.15 cFE Software Bus Commands . . . . .	48
2.16 cFE Software Bus Telemetry . . . . .	49
2.17 cFE Software Bus Configuration Parameters . . . . .	50
2.18 cFE Table Services Overview . . . . .	51
2.18.1 Managing Tables . . . . .	51
2.18.2 cFE Table Types and Table Options . . . . .	52
2.18.3 Table Registry . . . . .	54
2.18.4 Table Services Telemetry . . . . .	55
2.18.5 Effects of Processor Reset on Tables . . . . .	55
2.18.6 Frequently Asked Questions about Table Services . . . . .	55
2.19 cFE Table Services Commands . . . . .	57
2.20 cFE Table Services Telemetry . . . . .	57

---

2.21 cFE Table Services Configuration Parameters . . . . .	58
2.22 cFE Time Services Overview . . . . .	59
2.22.1 Time Components . . . . .	61
2.22.2 Time Structure . . . . .	61
2.22.3 Time Formats . . . . .	62
2.22.4 Time Configuration . . . . .	62
2.22.5 Time Format Selection . . . . .	67
2.22.6 Enabling Fake Tone Signal . . . . .	67
2.22.7 Selecting Tone and Data Ordering . . . . .	67
2.22.8 Specifying Tone and Data Window . . . . .	68
2.22.9 Specifying Time Server/Client . . . . .	68
2.22.10 Specifying Time Tone Byte Order . . . . .	68
2.22.11 Virtual MET . . . . .	69
2.22.12 Specifying Time Source . . . . .	69
2.22.13 Specifying Time Signal . . . . .	70
2.22.14 Time Services Paradigm(s) . . . . .	70
2.22.15 Flywheeling . . . . .	71
2.22.16 Time State . . . . .	71
2.22.17 Initialization . . . . .	71
2.22.18 Power-On Reset . . . . .	72
2.22.19 Processor Reset . . . . .	72
2.22.20 Initialization . . . . .	73
2.22.21 Power-On Reset . . . . .	73
2.22.22 Processor Reset . . . . .	74
2.22.23 Normal Operation . . . . .	74
2.22.24 Client . . . . .	76
2.22.25 Server . . . . .	76
2.22.26 Setting Time . . . . .	77
2.22.27 Adjusting Time . . . . .	78
2.22.28 Setting MET . . . . .	78
2.22.29 Frequently Asked Questions about Time Services . . . . .	78
2.23 cFE Time Services Commands . . . . .	78
2.24 cFE Time Services Telemetry . . . . .	79
2.25 cFE Time Services Configuration Parameters . . . . .	80
2.26 cFE Event Message Cross Reference . . . . .	81
2.27 cFE Command Mnemonic Cross Reference . . . . .	81
2.28 cFE Telemetry Mnemonic Cross Reference . . . . .	85

<b>4 cFE Application Programmer's Interface (API) Reference</b>	<b>97</b>
4.1 Executive Services API . . . . .	97
4.2 Events Services API . . . . .	99
4.3 File Services API . . . . .	100
4.4 Message API . . . . .	100
4.5 Resource ID API . . . . .	101
4.6 Software Bus Services API . . . . .	101
4.7 Table Services API . . . . .	102
4.8 Time Services API . . . . .	103
<b>5 Osal API Documentation</b>	<b>104</b>
5.1 OSAL Introduction . . . . .	105
5.2 File System Overview . . . . .	106
5.3 File Descriptors In Osal . . . . .	106
5.4 Timer Overview . . . . .	107
<b>6 cFE Mission Configuration Parameters</b>	<b>107</b>
<b>7 Module Index</b>	<b>108</b>
7.1 Modules . . . . .	108
<b>8 Data Structure Index</b>	<b>111</b>
8.1 Data Structures . . . . .	111
<b>9 File Index</b>	<b>120</b>
9.1 File List . . . . .	120
<b>10 Module Documentation</b>	<b>125</b>
10.1 CFS Checksum Event IDs . . . . .	125
10.1.1 Detailed Description . . . . .	132
10.1.2 Macro Definition Documentation . . . . .	132
10.2 CFS Checksum Mission Configuration . . . . .	172
10.2.1 Detailed Description . . . . .	172
10.2.2 Macro Definition Documentation . . . . .	172
10.3 CFS Checksum Telemetry . . . . .	173
10.3.1 Detailed Description . . . . .	173
10.4 CFS Checksum Command Structures . . . . .	174
10.4.1 Detailed Description . . . . .	174
10.5 CFS Checksum Command Codes . . . . .	175
10.5.1 Detailed Description . . . . .	176
10.5.2 Macro Definition Documentation . . . . .	176

---

10.6 CFS Checksum Command Message IDs . . . . .	208
10.6.1 Detailed Description . . . . .	208
10.6.2 Macro Definition Documentation . . . . .	208
10.7 CFS Checksum Telemetry Message IDs . . . . .	209
10.7.1 Detailed Description . . . . .	209
10.7.2 Macro Definition Documentation . . . . .	209
10.8 CFS Checksum Platform Configuration . . . . .	210
10.8.1 Detailed Description . . . . .	211
10.8.2 Macro Definition Documentation . . . . .	211
10.9 CFS Checksum Version . . . . .	217
10.9.1 Detailed Description . . . . .	217
10.9.2 Macro Definition Documentation . . . . .	217
10.10 cFE Return Code Defines . . . . .	218
10.10.1 Detailed Description . . . . .	223
10.10.2 Macro Definition Documentation . . . . .	223
10.11 cFE Resource ID APIs . . . . .	241
10.11.1 Detailed Description . . . . .	241
10.11.2 Function Documentation . . . . .	241
10.12 cFE Entry/Exit APIs . . . . .	244
10.12.1 Detailed Description . . . . .	244
10.12.2 Function Documentation . . . . .	244
10.13 cFE Application Control APIs . . . . .	246
10.13.1 Detailed Description . . . . .	246
10.13.2 Function Documentation . . . . .	246
10.14 cFE Application Behavior APIs . . . . .	249
10.14.1 Detailed Description . . . . .	249
10.14.2 Function Documentation . . . . .	249
10.15 cFE Information APIs . . . . .	253
10.15.1 Detailed Description . . . . .	253
10.15.2 Function Documentation . . . . .	253
10.16 cFE Child Task APIs . . . . .	262
10.16.1 Detailed Description . . . . .	262
10.16.2 Function Documentation . . . . .	262
10.17 cFE Miscellaneous APIs . . . . .	267
10.17.1 Detailed Description . . . . .	267
10.17.2 Function Documentation . . . . .	267
10.18 cFE Critical Data Store APIs . . . . .	270
10.18.1 Detailed Description . . . . .	270
10.18.2 Function Documentation . . . . .	270

10.19 cFE Memory Manager APIs . . . . .	275
10.19.1 Detailed Description . . . . .	275
10.19.2 Function Documentation . . . . .	275
10.20 cFE Performance Monitor APIs . . . . .	282
10.20.1 Detailed Description . . . . .	282
10.20.2 Macro Definition Documentation . . . . .	282
10.20.3 Function Documentation . . . . .	283
10.21 cFE Generic Counter APIs . . . . .	284
10.21.1 Detailed Description . . . . .	284
10.21.2 Function Documentation . . . . .	284
10.22 cFE Registration APIs . . . . .	290
10.22.1 Detailed Description . . . . .	290
10.22.2 Function Documentation . . . . .	290
10.23 cFE Send Event APIs . . . . .	292
10.23.1 Detailed Description . . . . .	292
10.23.2 Function Documentation . . . . .	292
10.24 cFE Reset Event Filter APIs . . . . .	297
10.24.1 Detailed Description . . . . .	297
10.24.2 Function Documentation . . . . .	297
10.25 cFE File Header Management APIs . . . . .	299
10.25.1 Detailed Description . . . . .	299
10.25.2 Function Documentation . . . . .	299
10.26 cFE File Utility APIs . . . . .	303
10.26.1 Detailed Description . . . . .	303
10.26.2 Function Documentation . . . . .	303
10.27 cFE Generic Message APIs . . . . .	308
10.27.1 Detailed Description . . . . .	308
10.27.2 Function Documentation . . . . .	308
10.28 cFE Message Primary Header APIs . . . . .	310
10.28.1 Detailed Description . . . . .	310
10.28.2 Function Documentation . . . . .	310
10.29 cFE Message Extended Header APIs . . . . .	319
10.29.1 Detailed Description . . . . .	319
10.29.2 Function Documentation . . . . .	319
10.30 cFE Message Secondary Header APIs . . . . .	325
10.30.1 Detailed Description . . . . .	325
10.30.2 Function Documentation . . . . .	325
10.31 cFE Message Id APIs . . . . .	330
10.31.1 Detailed Description . . . . .	330

---

10.31.2 Function Documentation . . . . .	330
10.32 cFE Message Checking APIs . . . . .	332
10.32.1 Detailed Description . . . . .	332
10.32.2 Function Documentation . . . . .	332
10.33 cFE Pipe Management APIs . . . . .	333
10.33.1 Detailed Description . . . . .	333
10.33.2 Function Documentation . . . . .	333
10.34 cFE Message Subscription Control APIs . . . . .	338
10.34.1 Detailed Description . . . . .	338
10.34.2 Function Documentation . . . . .	338
10.35 cFE Send/Receive Message APIs . . . . .	343
10.35.1 Detailed Description . . . . .	343
10.35.2 Function Documentation . . . . .	343
10.36 cFE Zero Copy APIs . . . . .	346
10.36.1 Detailed Description . . . . .	346
10.36.2 Function Documentation . . . . .	346
10.37 cFE Message Characteristics APIs . . . . .	349
10.37.1 Detailed Description . . . . .	349
10.37.2 Function Documentation . . . . .	349
10.38 cFE Message ID APIs . . . . .	354
10.38.1 Detailed Description . . . . .	354
10.38.2 Function Documentation . . . . .	354
10.39 cFE SB Pipe options . . . . .	356
10.39.1 Detailed Description . . . . .	356
10.39.2 Macro Definition Documentation . . . . .	356
10.40 cFE Registration APIs . . . . .	357
10.40.1 Detailed Description . . . . .	357
10.40.2 Function Documentation . . . . .	357
10.41 cFE Manage Table Content APIs . . . . .	362
10.41.1 Detailed Description . . . . .	362
10.41.2 Function Documentation . . . . .	362
10.42 cFE Access Table Content APIs . . . . .	368
10.42.1 Detailed Description . . . . .	368
10.42.2 Function Documentation . . . . .	368
10.43 cFE Get Table Information APIs . . . . .	373
10.43.1 Detailed Description . . . . .	373
10.43.2 Function Documentation . . . . .	373
10.44 cFE Table Type Defines . . . . .	376
10.44.1 Detailed Description . . . . .	376

10.44.2 Macro Definition Documentation . . . . .	376
10.45 cFE Get Current Time APIs . . . . .	378
10.45.1 Detailed Description . . . . .	378
10.45.2 Function Documentation . . . . .	378
10.46 cFE Get Time Information APIs . . . . .	381
10.46.1 Detailed Description . . . . .	381
10.46.2 Function Documentation . . . . .	381
10.47 cFE Time Arithmetic APIs . . . . .	384
10.47.1 Detailed Description . . . . .	384
10.47.2 Function Documentation . . . . .	384
10.48 cFE Time Conversion APIs . . . . .	387
10.48.1 Detailed Description . . . . .	387
10.48.2 Function Documentation . . . . .	387
10.49 cFE External Time Source APIs . . . . .	389
10.49.1 Detailed Description . . . . .	389
10.49.2 Function Documentation . . . . .	389
10.50 cFE Miscellaneous Time APIs . . . . .	394
10.50.1 Detailed Description . . . . .	394
10.50.2 Function Documentation . . . . .	394
10.51 cFE Resource ID base values . . . . .	397
10.51.1 Detailed Description . . . . .	397
10.51.2 Enumeration Type Documentation . . . . .	397
10.52 cFE Clock State Flag Defines . . . . .	399
10.52.1 Detailed Description . . . . .	399
10.52.2 Macro Definition Documentation . . . . .	399
10.53 OSAL Semaphore State Defines . . . . .	401
10.53.1 Detailed Description . . . . .	401
10.53.2 Macro Definition Documentation . . . . .	401
10.54 OSAL Binary Semaphore APIs . . . . .	402
10.54.1 Detailed Description . . . . .	402
10.54.2 Function Documentation . . . . .	402
10.55 OSAL BSP low level access APIs . . . . .	407
10.55.1 Detailed Description . . . . .	407
10.55.2 Function Documentation . . . . .	407
10.56 OSAL Real Time Clock APIs . . . . .	408
10.56.1 Detailed Description . . . . .	409
10.56.2 Function Documentation . . . . .	409
10.57 OSAL Core Operation APIs . . . . .	419
10.57.1 Detailed Description . . . . .	419

---

10.57.2 Function Documentation . . . . .	419
10.58 OSAL Condition Variable APIs . . . . .	422
10.58.1 Detailed Description . . . . .	422
10.58.2 Function Documentation . . . . .	422
10.59 OSAL Counting Semaphore APIs . . . . .	428
10.59.1 Detailed Description . . . . .	428
10.59.2 Function Documentation . . . . .	428
10.60 OSAL Directory APIs . . . . .	433
10.60.1 Detailed Description . . . . .	433
10.60.2 Function Documentation . . . . .	433
10.61 OSAL Return Code Defines . . . . .	437
10.61.1 Detailed Description . . . . .	439
10.61.2 Macro Definition Documentation . . . . .	439
10.62 OSAL Error Info APIs . . . . .	444
10.62.1 Detailed Description . . . . .	444
10.62.2 Function Documentation . . . . .	444
10.63 OSAL File Access Option Defines . . . . .	446
10.63.1 Detailed Description . . . . .	446
10.63.2 Macro Definition Documentation . . . . .	446
10.64 OSAL Reference Point For Seek Offset Defines . . . . .	447
10.64.1 Detailed Description . . . . .	447
10.64.2 Macro Definition Documentation . . . . .	447
10.65 OSAL Standard File APIs . . . . .	448
10.65.1 Detailed Description . . . . .	448
10.65.2 Function Documentation . . . . .	448
10.66 OSAL File System Level APIs . . . . .	459
10.66.1 Detailed Description . . . . .	459
10.66.2 Function Documentation . . . . .	459
10.67 OSAL Heap APIs . . . . .	467
10.67.1 Detailed Description . . . . .	467
10.67.2 Function Documentation . . . . .	467
10.68 OSAL Object Type Defines . . . . .	468
10.68.1 Detailed Description . . . . .	468
10.68.2 Macro Definition Documentation . . . . .	468
10.69 OSAL Object ID Utility APIs . . . . .	471
10.69.1 Detailed Description . . . . .	471
10.69.2 Function Documentation . . . . .	471
10.70 OSAL Dynamic Loader and Symbol APIs . . . . .	476
10.70.1 Detailed Description . . . . .	476

---

10.70.2 Function Documentation . . . . .	476
10.71 OSAL Mutex APIs . . . . .	480
10.71.1 Detailed Description . . . . .	480
10.71.2 Function Documentation . . . . .	480
10.72 OSAL Network ID APIs . . . . .	484
10.72.1 Detailed Description . . . . .	484
10.72.2 Function Documentation . . . . .	484
10.73 OSAL Printf APIs . . . . .	486
10.73.1 Detailed Description . . . . .	486
10.73.2 Function Documentation . . . . .	486
10.74 OSAL Message Queue APIs . . . . .	487
10.74.1 Detailed Description . . . . .	487
10.74.2 Function Documentation . . . . .	487
10.75 OSAL Select APIs . . . . .	491
10.75.1 Detailed Description . . . . .	491
10.75.2 Function Documentation . . . . .	491
10.76 OSAL Shell APIs . . . . .	495
10.76.1 Detailed Description . . . . .	495
10.76.2 Function Documentation . . . . .	495
10.77 OSAL Socket Address APIs . . . . .	496
10.77.1 Detailed Description . . . . .	496
10.77.2 Function Documentation . . . . .	496
10.78 OSAL Socket Management APIs . . . . .	500
10.78.1 Detailed Description . . . . .	500
10.78.2 Function Documentation . . . . .	500
10.79 OSAL Task APIs . . . . .	508
10.79.1 Detailed Description . . . . .	508
10.79.2 Function Documentation . . . . .	508
10.80 OSAL Time Base APIs . . . . .	514
10.80.1 Detailed Description . . . . .	514
10.80.2 Function Documentation . . . . .	514
10.81 OSAL Timer APIs . . . . .	519
10.81.1 Detailed Description . . . . .	519
10.81.2 Function Documentation . . . . .	519
<b>11 Data Structure Documentation</b>	<b>525</b>
11.1 CCSDS_ExtendedHeader Struct Reference . . . . .	525
11.1.1 Detailed Description . . . . .	525
11.1.2 Field Documentation . . . . .	525

---

11.2 CCSDS_PrimaryHeader Struct Reference . . . . .	525
11.2.1 Detailed Description . . . . .	525
11.2.2 Field Documentation . . . . .	526
11.3 CFE_ES_AppInfo Struct Reference . . . . .	526
11.3.1 Detailed Description . . . . .	527
11.3.2 Field Documentation . . . . .	527
11.4 CFE_ES_AppNameCmd Struct Reference . . . . .	530
11.4.1 Detailed Description . . . . .	530
11.4.2 Field Documentation . . . . .	530
11.5 CFE_ES_AppNameCmd_Payload Struct Reference . . . . .	531
11.5.1 Detailed Description . . . . .	531
11.5.2 Field Documentation . . . . .	531
11.6 CFE_ES_AppReloadCmd_Payload Struct Reference . . . . .	531
11.6.1 Detailed Description . . . . .	531
11.6.2 Field Documentation . . . . .	531
11.7 CFE_ES_BlockStats Struct Reference . . . . .	532
11.7.1 Detailed Description . . . . .	532
11.7.2 Field Documentation . . . . .	532
11.8 CFE_ES_CDSRegDumpRec Struct Reference . . . . .	533
11.8.1 Detailed Description . . . . .	533
11.8.2 Field Documentation . . . . .	533
11.9 CFE_ES_DeleteCDSCmd Struct Reference . . . . .	534
11.9.1 Detailed Description . . . . .	534
11.9.2 Field Documentation . . . . .	534
11.10 CFE_ES_DeleteCDSCmd_Payload Struct Reference . . . . .	534
11.10.1 Detailed Description . . . . .	534
11.10.2 Field Documentation . . . . .	535
11.11 CFE_ES_DumpCDSRegistryCmd Struct Reference . . . . .	535
11.11.1 Detailed Description . . . . .	535
11.11.2 Field Documentation . . . . .	535
11.12 CFE_ES_DumpCDSRegistryCmd_Payload Struct Reference . . . . .	535
11.12.1 Detailed Description . . . . .	536
11.12.2 Field Documentation . . . . .	536
11.13 CFE_ES_FileNameCmd Struct Reference . . . . .	536
11.13.1 Detailed Description . . . . .	536
11.13.2 Field Documentation . . . . .	536
11.14 CFE_ES_FileNameCmd_Payload Struct Reference . . . . .	536
11.14.1 Detailed Description . . . . .	537
11.14.2 Field Documentation . . . . .	537

11.15 CFE_ES_HousekeepingTlm Struct Reference . . . . .	537
11.15.1 Detailed Description . . . . .	537
11.15.2 Field Documentation . . . . .	537
11.16 CFE_ES_HousekeepingTlm_Payload Struct Reference . . . . .	538
11.16.1 Detailed Description . . . . .	539
11.16.2 Field Documentation . . . . .	540
11.17 CFE_ES_MemPoolStats Struct Reference . . . . .	546
11.17.1 Detailed Description . . . . .	546
11.17.2 Field Documentation . . . . .	546
11.18 CFE_ES_MemStatsTlm Struct Reference . . . . .	547
11.18.1 Detailed Description . . . . .	547
11.18.2 Field Documentation . . . . .	547
11.19 CFE_ES_NoArgsCmd Struct Reference . . . . .	547
11.19.1 Detailed Description . . . . .	548
11.19.2 Field Documentation . . . . .	548
11.20 CFE_ES_OneAppTlm Struct Reference . . . . .	548
11.20.1 Detailed Description . . . . .	548
11.20.2 Field Documentation . . . . .	548
11.21 CFE_ES_OneAppTlm_Payload Struct Reference . . . . .	549
11.21.1 Detailed Description . . . . .	549
11.21.2 Field Documentation . . . . .	549
11.22 CFE_ES_OverWriteSysLogCmd Struct Reference . . . . .	549
11.22.1 Detailed Description . . . . .	549
11.22.2 Field Documentation . . . . .	549
11.23 CFE_ES_OverWriteSysLogCmd_Payload Struct Reference . . . . .	550
11.23.1 Detailed Description . . . . .	550
11.23.2 Field Documentation . . . . .	550
11.24 CFE_ES_PoolAlign Union Reference . . . . .	550
11.24.1 Detailed Description . . . . .	551
11.24.2 Field Documentation . . . . .	551
11.25 CFE_ES_PoolStatsTlm_Payload Struct Reference . . . . .	551
11.25.1 Detailed Description . . . . .	551
11.25.2 Field Documentation . . . . .	551
11.26 CFE_ES_ReloadAppCmd Struct Reference . . . . .	552
11.26.1 Detailed Description . . . . .	552
11.26.2 Field Documentation . . . . .	552
11.27 CFE_ES_RestartCmd Struct Reference . . . . .	552
11.27.1 Detailed Description . . . . .	553
11.27.2 Field Documentation . . . . .	553

---

11.28 CFE_ES_RestartCmd_Payload Struct Reference . . . . .	553
11.28.1 Detailed Description . . . . .	553
11.28.2 Field Documentation . . . . .	553
11.29 CFE_ES_SendMemPoolStatsCmd Struct Reference . . . . .	553
11.29.1 Detailed Description . . . . .	554
11.29.2 Field Documentation . . . . .	554
11.30 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference . . . . .	554
11.30.1 Detailed Description . . . . .	554
11.30.2 Field Documentation . . . . .	554
11.31 CFE_ES_SetMaxPRCountCmd Struct Reference . . . . .	555
11.31.1 Detailed Description . . . . .	555
11.31.2 Field Documentation . . . . .	555
11.32 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference . . . . .	555
11.32.1 Detailed Description . . . . .	555
11.32.2 Field Documentation . . . . .	556
11.33 CFE_ES_SetPerfFilterMaskCmd Struct Reference . . . . .	556
11.33.1 Detailed Description . . . . .	556
11.33.2 Field Documentation . . . . .	556
11.34 CFE_ES_SetPerfFilterMaskCmd_Payload Struct Reference . . . . .	556
11.34.1 Detailed Description . . . . .	557
11.34.2 Field Documentation . . . . .	557
11.35 CFE_ES_SetPerfTriggerMaskCmd Struct Reference . . . . .	557
11.35.1 Detailed Description . . . . .	557
11.35.2 Field Documentation . . . . .	557
11.36 CFE_ES_SetPerfTrigMaskCmd_Payload Struct Reference . . . . .	558
11.36.1 Detailed Description . . . . .	558
11.36.2 Field Documentation . . . . .	558
11.37 CFE_ES_StartApp Struct Reference . . . . .	558
11.37.1 Detailed Description . . . . .	558
11.37.2 Field Documentation . . . . .	558
11.38 CFE_ES_StartAppCmd_Payload Struct Reference . . . . .	559
11.38.1 Detailed Description . . . . .	559
11.38.2 Field Documentation . . . . .	559
11.39 CFE_ES_StartPerfCmd_Payload Struct Reference . . . . .	560
11.39.1 Detailed Description . . . . .	560
11.39.2 Field Documentation . . . . .	560
11.40 CFE_ES_StartPerfDataCmd Struct Reference . . . . .	560
11.40.1 Detailed Description . . . . .	561
11.40.2 Field Documentation . . . . .	561

11.41 CFE_ES_StopPerfCmd_Payload Struct Reference . . . . .	561
11.41.1 Detailed Description . . . . .	561
11.41.2 Field Documentation . . . . .	561
11.42 CFE_ES_StopPerfDataCmd Struct Reference . . . . .	562
11.42.1 Detailed Description . . . . .	562
11.42.2 Field Documentation . . . . .	562
11.43 CFE_ES_TaskInfo Struct Reference . . . . .	562
11.43.1 Detailed Description . . . . .	563
11.43.2 Field Documentation . . . . .	563
11.44 CFE_EVS_AppDataCmd_Payload Struct Reference . . . . .	564
11.44.1 Detailed Description . . . . .	564
11.44.2 Field Documentation . . . . .	564
11.45 CFE_EVS_AppNameBitMaskCmd Struct Reference . . . . .	564
11.45.1 Detailed Description . . . . .	564
11.45.2 Field Documentation . . . . .	564
11.46 CFE_EVS_AppNameBitMaskCmd_Payload Struct Reference . . . . .	565
11.46.1 Detailed Description . . . . .	565
11.46.2 Field Documentation . . . . .	565
11.47 CFE_EVS_AppNameCmd Struct Reference . . . . .	566
11.47.1 Detailed Description . . . . .	566
11.47.2 Field Documentation . . . . .	566
11.48 CFE_EVS_AppNameCmd_Payload Struct Reference . . . . .	566
11.48.1 Detailed Description . . . . .	566
11.48.2 Field Documentation . . . . .	566
11.49 CFE_EVS_AppNameEventIDCmd Struct Reference . . . . .	567
11.49.1 Detailed Description . . . . .	567
11.49.2 Field Documentation . . . . .	567
11.50 CFE_EVS_AppNameEventIDCmd_Payload Struct Reference . . . . .	567
11.50.1 Detailed Description . . . . .	568
11.50.2 Field Documentation . . . . .	568
11.51 CFE_EVS_AppNameEventIDMaskCmd Struct Reference . . . . .	568
11.51.1 Detailed Description . . . . .	568
11.51.2 Field Documentation . . . . .	568
11.52 CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference . . . . .	569
11.52.1 Detailed Description . . . . .	569
11.52.2 Field Documentation . . . . .	569
11.53 CFE_EVS_AppTlmData Struct Reference . . . . .	569
11.53.1 Detailed Description . . . . .	570
11.53.2 Field Documentation . . . . .	570

---

11.54 CFE_EVS_BinFilter Struct Reference . . . . .	570
11.54.1 Detailed Description . . . . .	571
11.54.2 Field Documentation . . . . .	571
11.55 CFE_EVS_BitMaskCmd Struct Reference . . . . .	571
11.55.1 Detailed Description . . . . .	571
11.55.2 Field Documentation . . . . .	571
11.56 CFE_EVS_BitMaskCmd_Payload Struct Reference . . . . .	572
11.56.1 Detailed Description . . . . .	572
11.56.2 Field Documentation . . . . .	572
11.57 CFE_EVS_HousekeepingTlm Struct Reference . . . . .	572
11.57.1 Detailed Description . . . . .	572
11.57.2 Field Documentation . . . . .	572
11.58 CFE_EVS_HousekeepingTlm_Payload Struct Reference . . . . .	573
11.58.1 Detailed Description . . . . .	574
11.58.2 Field Documentation . . . . .	574
11.59 CFE_EVS_LogFileCmd_Payload Struct Reference . . . . .	576
11.59.1 Detailed Description . . . . .	576
11.59.2 Field Documentation . . . . .	576
11.60 CFE_EVS_LongEventTlm Struct Reference . . . . .	576
11.60.1 Detailed Description . . . . .	577
11.60.2 Field Documentation . . . . .	577
11.61 CFE_EVS_LongEventTlm_Payload Struct Reference . . . . .	577
11.61.1 Detailed Description . . . . .	577
11.61.2 Field Documentation . . . . .	577
11.62 CFE_EVS_NoArgsCmd Struct Reference . . . . .	578
11.62.1 Detailed Description . . . . .	578
11.62.2 Field Documentation . . . . .	578
11.63 CFE_EVS_PacketID Struct Reference . . . . .	578
11.63.1 Detailed Description . . . . .	579
11.63.2 Field Documentation . . . . .	579
11.64 CFE_EVS_SetEventFormatCode_Payload Struct Reference . . . . .	580
11.64.1 Detailed Description . . . . .	580
11.64.2 Field Documentation . . . . .	580
11.65 CFE_EVS_SetEventFormatModeCmd Struct Reference . . . . .	580
11.65.1 Detailed Description . . . . .	581
11.65.2 Field Documentation . . . . .	581
11.66 CFE_EVS_SetLogMode_Payload Struct Reference . . . . .	581
11.66.1 Detailed Description . . . . .	581
11.66.2 Field Documentation . . . . .	581

11.67 CFE_EVS_SetLogModeCmd Struct Reference . . . . .	582
11.67.1 Detailed Description . . . . .	582
11.67.2 Field Documentation . . . . .	582
11.68 CFE_EVS_ShortEventTlm Struct Reference . . . . .	582
11.68.1 Detailed Description . . . . .	582
11.68.2 Field Documentation . . . . .	582
11.69 CFE_EVS_ShortEventTlm_Payload Struct Reference . . . . .	583
11.69.1 Detailed Description . . . . .	583
11.69.2 Field Documentation . . . . .	583
11.70 CFE_EVS_WriteAppDataFileCmd Struct Reference . . . . .	583
11.70.1 Detailed Description . . . . .	583
11.70.2 Field Documentation . . . . .	584
11.71 CFE_EVS_WriteLogDataFileCmd Struct Reference . . . . .	584
11.71.1 Detailed Description . . . . .	584
11.71.2 Field Documentation . . . . .	584
11.72 CFE_FS_FileWriteMetaData Struct Reference . . . . .	584
11.72.1 Detailed Description . . . . .	585
11.72.2 Field Documentation . . . . .	585
11.73 CFE_FS_Header Struct Reference . . . . .	586
11.73.1 Detailed Description . . . . .	586
11.73.2 Field Documentation . . . . .	586
11.74 CFE_SB_AllSubscriptionsTlm Struct Reference . . . . .	587
11.74.1 Detailed Description . . . . .	587
11.74.2 Field Documentation . . . . .	587
11.75 CFE_SB_AllSubscriptionsTlm_Payload Struct Reference . . . . .	588
11.75.1 Detailed Description . . . . .	588
11.75.2 Field Documentation . . . . .	588
11.76 CFE_SB_HousekeepingTlm Struct Reference . . . . .	589
11.76.1 Detailed Description . . . . .	589
11.76.2 Field Documentation . . . . .	589
11.77 CFE_SB_HousekeepingTlm_Payload Struct Reference . . . . .	589
11.77.1 Detailed Description . . . . .	590
11.77.2 Field Documentation . . . . .	590
11.78 CFE_SB_Msg Union Reference . . . . .	593
11.78.1 Detailed Description . . . . .	593
11.78.2 Field Documentation . . . . .	593
11.79 CFE_SB_MsgId_t Struct Reference . . . . .	594
11.79.1 Detailed Description . . . . .	594
11.79.2 Field Documentation . . . . .	594

---

---

11.80 CFE_SB_MsgMapFileEntry Struct Reference . . . . .	594
11.80.1 Detailed Description . . . . .	594
11.80.2 Field Documentation . . . . .	595
11.81 CFE_SB_PipeDepthStats Struct Reference . . . . .	595
11.81.1 Detailed Description . . . . .	595
11.81.2 Field Documentation . . . . .	595
11.82 CFE_SB_PipeInfoEntry Struct Reference . . . . .	596
11.82.1 Detailed Description . . . . .	597
11.82.2 Field Documentation . . . . .	597
11.83 CFE_SB_Qos_t Struct Reference . . . . .	598
11.83.1 Detailed Description . . . . .	598
11.83.2 Field Documentation . . . . .	598
11.84 CFE_SB_RouteCmd Struct Reference . . . . .	598
11.84.1 Detailed Description . . . . .	599
11.84.2 Field Documentation . . . . .	599
11.85 CFE_SB_RouteCmd_Payload Struct Reference . . . . .	599
11.85.1 Detailed Description . . . . .	599
11.85.2 Field Documentation . . . . .	599
11.86 CFE_SB_RoutingFileEntry Struct Reference . . . . .	600
11.86.1 Detailed Description . . . . .	600
11.86.2 Field Documentation . . . . .	600
11.87 CFE_SB_SingleSubscriptionTlm Struct Reference . . . . .	601
11.87.1 Detailed Description . . . . .	601
11.87.2 Field Documentation . . . . .	601
11.88 CFE_SB_SingleSubscriptionTlm_Payload Struct Reference . . . . .	602
11.88.1 Detailed Description . . . . .	602
11.88.2 Field Documentation . . . . .	602
11.89 CFE_SB_StatsTlm Struct Reference . . . . .	603
11.89.1 Detailed Description . . . . .	603
11.89.2 Field Documentation . . . . .	603
11.90 CFE_SB_StatsTlm_Payload Struct Reference . . . . .	603
11.90.1 Detailed Description . . . . .	604
11.90.2 Field Documentation . . . . .	604
11.91 CFE_SB_SubEntries Struct Reference . . . . .	607
11.91.1 Detailed Description . . . . .	607
11.91.2 Field Documentation . . . . .	607
11.92 CFE_SB_WriteFileInfoCmd Struct Reference . . . . .	607
11.92.1 Detailed Description . . . . .	608
11.92.2 Field Documentation . . . . .	608

11.93 CFE_SB_WriteFileInfoCmd_Payload Struct Reference . . . . .	608
11.93.1 Detailed Description . . . . .	608
11.93.2 Field Documentation . . . . .	608
11.94 CFE_TBL_AbortLoadCmd Struct Reference . . . . .	608
11.94.1 Detailed Description . . . . .	609
11.94.2 Field Documentation . . . . .	609
11.95 CFE_TBL_AbortLoadCmd_Payload Struct Reference . . . . .	609
11.95.1 Detailed Description . . . . .	609
11.95.2 Field Documentation . . . . .	609
11.96 CFE_TBL_ActivateCmd Struct Reference . . . . .	610
11.96.1 Detailed Description . . . . .	610
11.96.2 Field Documentation . . . . .	610
11.97 CFE_TBL_ActivateCmd_Payload Struct Reference . . . . .	610
11.97.1 Detailed Description . . . . .	610
11.97.2 Field Documentation . . . . .	610
11.98 CFE_TBL_DelCDSCmd_Payload Struct Reference . . . . .	611
11.98.1 Detailed Description . . . . .	611
11.98.2 Field Documentation . . . . .	611
11.99 CFE_TBL_DeleteCDSCmd Struct Reference . . . . .	611
11.99.1 Detailed Description . . . . .	611
11.99.2 Field Documentation . . . . .	612
11.100 CFE_TBL_DumpCmd Struct Reference . . . . .	612
11.100.1 Detailed Description . . . . .	612
11.100.2 Field Documentation . . . . .	612
11.101 CFE_TBL_DumpCmd_Payload Struct Reference . . . . .	612
11.101.1 Detailed Description . . . . .	613
11.101.2 Field Documentation . . . . .	613
11.102 CFE_TBL_DumpRegistryCmd Struct Reference . . . . .	613
11.102.1 Detailed Description . . . . .	614
11.102.2 Field Documentation . . . . .	614
11.103 CFE_TBL_DumpRegistryCmd_Payload Struct Reference . . . . .	614
11.103.1 Detailed Description . . . . .	614
11.103.2 Field Documentation . . . . .	614
11.104 CFE_TBL_File_Hdr Struct Reference . . . . .	614
11.104.1 Detailed Description . . . . .	615
11.104.2 Field Documentation . . . . .	615
11.105 CFE_TBL_FileDef Struct Reference . . . . .	615
11.105.1 Detailed Description . . . . .	616
11.105.2 Field Documentation . . . . .	616

---

---

11.106 CFE_TBL_HousekeepingTlm Struct Reference . . . . .	617
11.106.1 Detailed Description . . . . .	617
11.106.2 Field Documentation . . . . .	617
11.107 CFE_TBL_HousekeepingTlm_Payload Struct Reference . . . . .	617
11.107.1 Detailed Description . . . . .	618
11.107.2 Field Documentation . . . . .	619
11.108 CFE_TBL_Info Struct Reference . . . . .	621
11.108.1 Detailed Description . . . . .	622
11.108.2 Field Documentation . . . . .	622
11.109 CFE_TBL_LoadCmd Struct Reference . . . . .	623
11.109.1 Detailed Description . . . . .	624
11.109.2 Field Documentation . . . . .	624
11.110 CFE_TBL_LoadCmd_Payload Struct Reference . . . . .	624
11.110.1 Detailed Description . . . . .	624
11.110.2 Field Documentation . . . . .	624
11.111 CFE_TBL_NoArgsCmd Struct Reference . . . . .	625
11.111.1 Detailed Description . . . . .	625
11.111.2 Field Documentation . . . . .	625
11.112 CFE_TBL_NotifyCmd Struct Reference . . . . .	625
11.112.1 Detailed Description . . . . .	625
11.112.2 Field Documentation . . . . .	625
11.113 CFE_TBL_NotifyCmd_Payload Struct Reference . . . . .	626
11.113.1 Detailed Description . . . . .	626
11.113.2 Field Documentation . . . . .	626
11.114 CFE_TBL_SendRegistryCmd Struct Reference . . . . .	626
11.114.1 Detailed Description . . . . .	627
11.114.2 Field Documentation . . . . .	627
11.115 CFE_TBL_SendRegistryCmd_Payload Struct Reference . . . . .	627
11.115.1 Detailed Description . . . . .	627
11.115.2 Field Documentation . . . . .	627
11.116 CFE_TBL_TableRegistryTlm Struct Reference . . . . .	627
11.116.1 Detailed Description . . . . .	628
11.116.2 Field Documentation . . . . .	628
11.117 CFE_TBL_TblRegPacket_Payload Struct Reference . . . . .	628
11.117.1 Detailed Description . . . . .	629
11.117.2 Field Documentation . . . . .	629
11.118 CFE_TBL_ValidateCmd Struct Reference . . . . .	632
11.118.1 Detailed Description . . . . .	632
11.118.2 Field Documentation . . . . .	632

11.119 CFE_TBL_ValidateCmd_Payload Struct Reference . . . . .	632
11.119.1 Detailed Description . . . . .	632
11.119.2 Field Documentation . . . . .	632
11.120 CFE_TIME_DiagnosticTlm Struct Reference . . . . .	633
11.120.1 Detailed Description . . . . .	633
11.120.2 Field Documentation . . . . .	633
11.121 CFE_TIME_DiagnosticTlm_Payload Struct Reference . . . . .	633
11.121.1 Detailed Description . . . . .	635
11.121.2 Field Documentation . . . . .	635
11.122 CFE_TIME_HousekeepingTlm Struct Reference . . . . .	642
11.122.1 Detailed Description . . . . .	642
11.122.2 Field Documentation . . . . .	642
11.123 CFE_TIME_HousekeepingTlm_Payload Struct Reference . . . . .	642
11.123.1 Detailed Description . . . . .	643
11.123.2 Field Documentation . . . . .	643
11.124 CFE_TIME_LeapsCmd_Payload Struct Reference . . . . .	645
11.124.1 Detailed Description . . . . .	645
11.124.2 Field Documentation . . . . .	645
11.125 CFE_TIME_NoArgsCmd Struct Reference . . . . .	645
11.125.1 Detailed Description . . . . .	645
11.125.2 Field Documentation . . . . .	646
11.126 CFE_TIME_OneHzAdjustmentCmd Struct Reference . . . . .	646
11.126.1 Detailed Description . . . . .	646
11.126.2 Field Documentation . . . . .	646
11.127 CFE_TIME_OneHzAdjustmentCmd_Payload Struct Reference . . . . .	646
11.127.1 Detailed Description . . . . .	647
11.127.2 Field Documentation . . . . .	647
11.128 CFE_TIME_SetLeapSecondsCmd Struct Reference . . . . .	647
11.128.1 Detailed Description . . . . .	647
11.128.2 Field Documentation . . . . .	647
11.129 CFE_TIME_SetSignalCmd Struct Reference . . . . .	648
11.129.1 Detailed Description . . . . .	648
11.129.2 Field Documentation . . . . .	648
11.130 CFE_TIME_SetSourceCmd Struct Reference . . . . .	648
11.130.1 Detailed Description . . . . .	648
11.130.2 Field Documentation . . . . .	648
11.131 CFE_TIME_SetStateCmd Struct Reference . . . . .	649
11.131.1 Detailed Description . . . . .	649
11.131.2 Field Documentation . . . . .	649

---

11.132 CFE_TIME_SignalCmd_Payload Struct Reference . . . . .	649
11.132.1 Detailed Description . . . . .	650
11.132.2 Field Documentation . . . . .	650
11.133 CFE_TIME_SourceCmd_Payload Struct Reference . . . . .	650
11.133.1 Detailed Description . . . . .	650
11.133.2 Field Documentation . . . . .	650
11.134 CFE_TIME_StateCmd_Payload Struct Reference . . . . .	651
11.134.1 Detailed Description . . . . .	651
11.134.2 Field Documentation . . . . .	651
11.135 CFE_TIME_SysTime Struct Reference . . . . .	651
11.135.1 Detailed Description . . . . .	651
11.135.2 Field Documentation . . . . .	652
11.136 CFE_TIME_TimeCmd Struct Reference . . . . .	652
11.136.1 Detailed Description . . . . .	652
11.136.2 Field Documentation . . . . .	652
11.137 CFE_TIME_TimeCmd_Payload Struct Reference . . . . .	652
11.137.1 Detailed Description . . . . .	653
11.137.2 Field Documentation . . . . .	653
11.138 CFE_TIME_ToneDataCmd Struct Reference . . . . .	653
11.138.1 Detailed Description . . . . .	653
11.138.2 Field Documentation . . . . .	653
11.139 CFE_TIME_ToneDataCmd_Payload Struct Reference . . . . .	654
11.139.1 Detailed Description . . . . .	654
11.139.2 Field Documentation . . . . .	654
11.140 CS_AppData_t Struct Reference . . . . .	654
11.140.1 Detailed Description . . . . .	656
11.140.2 Field Documentation . . . . .	656
11.141 CS_AppNameCmd_Payload_t Struct Reference . . . . .	662
11.141.1 Detailed Description . . . . .	662
11.141.2 Field Documentation . . . . .	662
11.142 CS_AppNameCmd_t Struct Reference . . . . .	662
11.142.1 Detailed Description . . . . .	663
11.142.2 Field Documentation . . . . .	663
11.143 CS_Def_App_Table_Entry_t Struct Reference . . . . .	663
11.143.1 Detailed Description . . . . .	663
11.143.2 Field Documentation . . . . .	663
11.144 CS_Def_EepromMemory_Table_Entry_t Struct Reference . . . . .	664
11.144.1 Detailed Description . . . . .	664
11.144.2 Field Documentation . . . . .	664

11.145 CS_Def_Tables_Table_Entry_t Struct Reference . . . . .	665
11.145.1 Detailed Description . . . . .	665
11.145.2 Field Documentation . . . . .	665
11.146 CS_EntryCmd_Payload_t Struct Reference . . . . .	665
11.146.1 Detailed Description . . . . .	666
11.146.2 Field Documentation . . . . .	666
11.147 CS_EntryCmd_t Struct Reference . . . . .	666
11.147.1 Detailed Description . . . . .	666
11.147.2 Field Documentation . . . . .	666
11.148 CS_GetEntryIDCmd_Payload_t Struct Reference . . . . .	667
11.148.1 Detailed Description . . . . .	667
11.148.2 Field Documentation . . . . .	667
11.149 CS_GetEntryIDCmd_t Struct Reference . . . . .	667
11.149.1 Detailed Description . . . . .	667
11.149.2 Field Documentation . . . . .	667
11.150 CS_HkPacket_Payload_t Struct Reference . . . . .	668
11.150.1 Detailed Description . . . . .	669
11.150.2 Field Documentation . . . . .	669
11.151 CS_HkPacket_t Struct Reference . . . . .	673
11.151.1 Detailed Description . . . . .	673
11.151.2 Field Documentation . . . . .	673
11.152 CS_NoArgsCmd_t Struct Reference . . . . .	674
11.152.1 Detailed Description . . . . .	674
11.152.2 Field Documentation . . . . .	674
11.153 CS_OneShotCmd_Payload_t Struct Reference . . . . .	675
11.153.1 Detailed Description . . . . .	675
11.153.2 Field Documentation . . . . .	675
11.154 CS_OneShotCmd_t Struct Reference . . . . .	675
11.154.1 Detailed Description . . . . .	676
11.154.2 Field Documentation . . . . .	676
11.155 CS_Res_App_Table_Entry_t Struct Reference . . . . .	676
11.155.1 Detailed Description . . . . .	676
11.155.2 Field Documentation . . . . .	676
11.156 CS_Res_EepromMemory_Table_Entry_t Struct Reference . . . . .	678
11.156.1 Detailed Description . . . . .	678
11.156.2 Field Documentation . . . . .	678
11.157 CS_Res_Tables_Table_Entry_t Struct Reference . . . . .	679
11.157.1 Detailed Description . . . . .	680
11.157.2 Field Documentation . . . . .	680

---

11.158 CS_TableNameCmd_Payload_t Struct Reference . . . . .	682
11.158.1 Detailed Description . . . . .	682
11.158.2 Field Documentation . . . . .	682
11.159 CS_TableNameCmd_t Struct Reference . . . . .	682
11.159.1 Detailed Description . . . . .	682
11.159.2 Field Documentation . . . . .	683
11.160 OS_bin_sem_prop_t Struct Reference . . . . .	683
11.160.1 Detailed Description . . . . .	683
11.160.2 Field Documentation . . . . .	683
11.161 OS_condvar_prop_t Struct Reference . . . . .	683
11.161.1 Detailed Description . . . . .	684
11.161.2 Field Documentation . . . . .	684
11.162 OS_count_sem_prop_t Struct Reference . . . . .	684
11.162.1 Detailed Description . . . . .	684
11.162.2 Field Documentation . . . . .	684
11.163 os_dirent_t Struct Reference . . . . .	685
11.163.1 Detailed Description . . . . .	685
11.163.2 Field Documentation . . . . .	685
11.164 OS_FdSet Struct Reference . . . . .	685
11.164.1 Detailed Description . . . . .	685
11.164.2 Field Documentation . . . . .	685
11.165 OS_file_prop_t Struct Reference . . . . .	686
11.165.1 Detailed Description . . . . .	686
11.165.2 Field Documentation . . . . .	686
11.166 os_fsinfo_t Struct Reference . . . . .	686
11.166.1 Detailed Description . . . . .	687
11.166.2 Field Documentation . . . . .	687
11.167 os_fstat_t Struct Reference . . . . .	687
11.167.1 Detailed Description . . . . .	687
11.167.2 Field Documentation . . . . .	688
11.168 OS_heap_prop_t Struct Reference . . . . .	688
11.168.1 Detailed Description . . . . .	688
11.168.2 Field Documentation . . . . .	688
11.169 OS_module_address_t Struct Reference . . . . .	689
11.169.1 Detailed Description . . . . .	689
11.169.2 Field Documentation . . . . .	689
11.170 OS_module_prop_t Struct Reference . . . . .	690
11.170.1 Detailed Description . . . . .	690
11.170.2 Field Documentation . . . . .	690

11.171 OS_mut_sem_prop_t Struct Reference . . . . .	690
11.171.1 Detailed Description . . . . .	691
11.171.2 Field Documentation . . . . .	691
11.172 OS_queue_prop_t Struct Reference . . . . .	691
11.172.1 Detailed Description . . . . .	691
11.172.2 Field Documentation . . . . .	691
11.173 OS_SockAddr_t Struct Reference . . . . .	692
11.173.1 Detailed Description . . . . .	692
11.173.2 Field Documentation . . . . .	692
11.174 OS_SockAddrData_t Union Reference . . . . .	692
11.174.1 Detailed Description . . . . .	692
11.174.2 Field Documentation . . . . .	693
11.175 OS_socket_prop_t Struct Reference . . . . .	693
11.175.1 Detailed Description . . . . .	693
11.175.2 Field Documentation . . . . .	693
11.176 OS_static_symbol_record_t Struct Reference . . . . .	694
11.176.1 Detailed Description . . . . .	694
11.176.2 Field Documentation . . . . .	694
11.177 OS_statvfs_t Struct Reference . . . . .	694
11.177.1 Detailed Description . . . . .	694
11.177.2 Field Documentation . . . . .	695
11.178 OS_task_prop_t Struct Reference . . . . .	695
11.178.1 Detailed Description . . . . .	695
11.178.2 Field Documentation . . . . .	695
11.179 OS_time_t Struct Reference . . . . .	696
11.179.1 Detailed Description . . . . .	696
11.179.2 Field Documentation . . . . .	696
11.180 OS_timebase_prop_t Struct Reference . . . . .	696
11.180.1 Detailed Description . . . . .	697
11.180.2 Field Documentation . . . . .	697
11.181 OS_timer_prop_t Struct Reference . . . . .	697
11.181.1 Detailed Description . . . . .	697
11.181.2 Field Documentation . . . . .	697
<b>12 File Documentation</b> . . . . .	<b>698</b>
12.1 apps/cs/docs/dox_src/cfs_cs.dox File Reference . . . . .	698
12.2 apps/cs/fsw/inc/cs_events.h File Reference . . . . .	698
12.2.1 Detailed Description . . . . .	705
12.3 apps/cs/fsw/inc/cs_mission_cfg.h File Reference . . . . .	705

---

---

12.3.1 Detailed Description . . . . .	705
12.4 apps/cs/fsw/inc/cs_msg.h File Reference . . . . .	705
12.4.1 Detailed Description . . . . .	706
12.5 apps/cs/fsw/inc/cs_msgdefs.h File Reference . . . . .	706
12.5.1 Detailed Description . . . . .	708
12.5.2 Macro Definition Documentation . . . . .	708
12.6 apps/cs/fsw/inc/cs_msgids.h File Reference . . . . .	710
12.6.1 Detailed Description . . . . .	710
12.7 apps/cs/fsw/inc/cs_perfids.h File Reference . . . . .	710
12.7.1 Detailed Description . . . . .	710
12.8 apps/cs/fsw/inc/cs_platform_cfg.h File Reference . . . . .	710
12.8.1 Detailed Description . . . . .	712
12.9 apps/cs/fsw/inc/cs_tbldefs.h File Reference . . . . .	712
12.9.1 Detailed Description . . . . .	713
12.9.2 Macro Definition Documentation . . . . .	713
12.9.3 Function Documentation . . . . .	714
12.10 apps/cs/fsw/src/cs_app.c File Reference . . . . .	724
12.10.1 Detailed Description . . . . .	725
12.10.2 Macro Definition Documentation . . . . .	725
12.10.3 Function Documentation . . . . .	725
12.10.4 Variable Documentation . . . . .	735
12.11 apps/cs/fsw/src/cs_app.h File Reference . . . . .	736
12.11.1 Detailed Description . . . . .	737
12.11.2 Macro Definition Documentation . . . . .	737
12.11.3 Function Documentation . . . . .	739
12.11.4 Variable Documentation . . . . .	748
12.12 apps/cs/fsw/src/cs_app_cmds.c File Reference . . . . .	749
12.12.1 Detailed Description . . . . .	749
12.12.2 Function Documentation . . . . .	749
12.13 apps/cs/fsw/src/cs_app_cmds.h File Reference . . . . .	755
12.13.1 Detailed Description . . . . .	755
12.13.2 Function Documentation . . . . .	755
12.14 apps/cs/fsw/src/cs_cmds.c File Reference . . . . .	761
12.14.1 Detailed Description . . . . .	762
12.14.2 Function Documentation . . . . .	762
12.15 apps/cs/fsw/src/cs_cmds.h File Reference . . . . .	776
12.15.1 Detailed Description . . . . .	777
12.15.2 Function Documentation . . . . .	777
12.16 apps/cs/fsw/src/cs_compute.c File Reference . . . . .	791

12.16.1 Detailed Description . . . . .	792
12.16.2 Function Documentation . . . . .	792
12.17 apps/cs/fsw/src/cs_compute.h File Reference . . . . .	799
12.17.1 Detailed Description . . . . .	799
12.17.2 Function Documentation . . . . .	799
12.18 apps/cs/fsw/src/cs_eeprom_cmds.c File Reference . . . . .	807
12.18.1 Detailed Description . . . . .	807
12.18.2 Function Documentation . . . . .	808
12.19 apps/cs/fsw/src/cs_eeprom_cmds.h File Reference . . . . .	814
12.19.1 Detailed Description . . . . .	814
12.19.2 Function Documentation . . . . .	814
12.20 apps/cs/fsw/src/cs_init.c File Reference . . . . .	820
12.20.1 Function Documentation . . . . .	821
12.21 apps/cs/fsw/src/cs_init.h File Reference . . . . .	824
12.21.1 Detailed Description . . . . .	824
12.21.2 Function Documentation . . . . .	824
12.22 apps/cs/fsw/src/cs_memory_cmds.c File Reference . . . . .	828
12.22.1 Detailed Description . . . . .	829
12.22.2 Function Documentation . . . . .	829
12.23 apps/cs/fsw/src/cs_memory_cmds.h File Reference . . . . .	835
12.23.1 Detailed Description . . . . .	835
12.23.2 Function Documentation . . . . .	835
12.24 apps/cs/fsw/src/cs_table_cmds.c File Reference . . . . .	842
12.24.1 Detailed Description . . . . .	842
12.24.2 Function Documentation . . . . .	843
12.25 apps/cs/fsw/src/cs_table_cmds.h File Reference . . . . .	848
12.25.1 Detailed Description . . . . .	848
12.25.2 Function Documentation . . . . .	848
12.26 apps/cs/fsw/src/cs_table_processing.c File Reference . . . . .	854
12.26.1 Detailed Description . . . . .	855
12.26.2 Function Documentation . . . . .	855
12.27 apps/cs/fsw/src/cs_utils.c File Reference . . . . .	865
12.27.1 Detailed Description . . . . .	866
12.27.2 Function Documentation . . . . .	866
12.28 apps/cs/fsw/src/cs_utils.h File Reference . . . . .	884
12.28.1 Detailed Description . . . . .	886
12.28.2 Function Documentation . . . . .	886
12.29 apps/cs/fsw/src/cs_verify.h File Reference . . . . .	904
12.29.1 Detailed Description . . . . .	904

---

12.30 apps/cs/fsw/src/cs_version.h File Reference . . . . .	905
12.30.1 Detailed Description . . . . .	905
12.31 apps/cs/fsw/tables/cs_apptbl.c File Reference . . . . .	905
12.31.1 Detailed Description . . . . .	905
12.31.2 Variable Documentation . . . . .	905
12.32 apps/cs/fsw/tables/cs_eepromtbl.c File Reference . . . . .	906
12.32.1 Detailed Description . . . . .	906
12.32.2 Variable Documentation . . . . .	906
12.33 apps/cs/fsw/tables/cs_memorytbl.c File Reference . . . . .	906
12.33.1 Detailed Description . . . . .	906
12.33.2 Variable Documentation . . . . .	906
12.34 apps/cs/fsw/tables/cs_tablestbl.c File Reference . . . . .	907
12.34.1 Detailed Description . . . . .	907
12.34.2 Variable Documentation . . . . .	907
12.35 build/osal_public_api/inc/osconfig.h File Reference . . . . .	907
12.35.1 Macro Definition Documentation . . . . .	909
12.36 cfe/cmake/sample_defs/example_mission_cfg.h File Reference . . . . .	913
12.36.1 Detailed Description . . . . .	914
12.36.2 Macro Definition Documentation . . . . .	915
12.37 cfe/cmake/sample_defs/example_platform_cfg.h File Reference . . . . .	924
12.37.1 Detailed Description . . . . .	928
12.37.2 Macro Definition Documentation . . . . .	928
12.38 cfe/cmake/sample_defs/sample_perfids.h File Reference . . . . .	968
12.38.1 Detailed Description . . . . .	969
12.38.2 Macro Definition Documentation . . . . .	969
12.39 cfe/docs/src/cfe_api.dox File Reference . . . . .	971
12.40 cfe/docs/src/cfe_es.dox File Reference . . . . .	971
12.41 cfe/docs/src/cfe_evs.dox File Reference . . . . .	971
12.42 cfe/docs/src/cfe_frontpage.dox File Reference . . . . .	971
12.43 cfe/docs/src/cfe_glossary.dox File Reference . . . . .	971
12.44 cfe/docs/src/cfe_sb.dox File Reference . . . . .	971
12.45 cfe/docs/src/cfe_tbl.dox File Reference . . . . .	971
12.46 cfe/docs/src/cfe_time.dox File Reference . . . . .	971
12.47 cfe/docs/src/cfe_xref.dox File Reference . . . . .	971
12.48 cfe/docs/src/cfs_versions.dox File Reference . . . . .	971
12.49 cfe/modules/core_api/config/default_cfe_core_api_base_msgids.h File Reference . . . . .	971
12.49.1 Detailed Description . . . . .	971
12.49.2 Macro Definition Documentation . . . . .	971
12.50 cfe/modules/core_api/config/default_cfe_core_api_interface_cfg.h File Reference . . . . .	972

---

12.50.1 Detailed Description . . . . .	972
12.50.2 Macro Definition Documentation . . . . .	972
12.51 cfe/modules/core_api/config/default_cfe_mission_cfg.h File Reference . . . . .	974
12.51.1 Detailed Description . . . . .	974
12.52 cfe/modules/core_api/config/default_cfe_msgids.h File Reference . . . . .	974
12.52.1 Detailed Description . . . . .	974
12.53 cfe/modules/core_api/fsw/inc/cfe.h File Reference . . . . .	974
12.53.1 Detailed Description . . . . .	975
12.54 cfe/modules/core_api/fsw/inc/cfe_config.h File Reference . . . . .	975
12.54.1 Detailed Description . . . . .	975
12.54.2 Function Documentation . . . . .	975
12.55 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h File Reference . . . . .	978
12.55.1 Detailed Description . . . . .	978
12.55.2 Macro Definition Documentation . . . . .	978
12.55.3 Typedef Documentation . . . . .	978
12.56 cfe/modules/core_api/fsw/inc/cfe_endian.h File Reference . . . . .	979
12.56.1 Detailed Description . . . . .	979
12.56.2 Macro Definition Documentation . . . . .	979
12.57 cfe/modules/core_api/fsw/inc/cfe_error.h File Reference . . . . .	979
12.57.1 Detailed Description . . . . .	986
12.57.2 Macro Definition Documentation . . . . .	986
12.57.3 Typedef Documentation . . . . .	987
12.57.4 Function Documentation . . . . .	987
12.58 cfe/modules/core_api/fsw/inc/cfe_es.h File Reference . . . . .	988
12.58.1 Detailed Description . . . . .	991
12.58.2 Macro Definition Documentation . . . . .	991
12.59 cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h File Reference . . . . .	991
12.59.1 Detailed Description . . . . .	993
12.59.2 Macro Definition Documentation . . . . .	993
12.59.3 Typedef Documentation . . . . .	995
12.59.4 Enumeration Type Documentation . . . . .	996
12.60 cfe/modules/core_api/fsw/inc/cfe_evs.h File Reference . . . . .	996
12.60.1 Detailed Description . . . . .	997
12.60.2 Macro Definition Documentation . . . . .	997
12.61 cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h File Reference . . . . .	998
12.61.1 Detailed Description . . . . .	998
12.61.2 Macro Definition Documentation . . . . .	998
12.61.3 Typedef Documentation . . . . .	1000
12.62 cfe/modules/core_api/fsw/inc/cfe_fs.h File Reference . . . . .	1000

---

12.62.1 Detailed Description . . . . .	1001
12.63 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h File Reference . . . . .	1001
12.63.1 Detailed Description . . . . .	1001
12.63.2 Typedef Documentation . . . . .	1001
12.63.3 Enumeration Type Documentation . . . . .	1002
12.64 cfe/modules/core_api/fsw/inc/cfe_msg.h File Reference . . . . .	1003
12.64.1 Detailed Description . . . . .	1005
12.65 cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h File Reference . . . . .	1005
12.65.1 Detailed Description . . . . .	1007
12.65.2 Macro Definition Documentation . . . . .	1007
12.65.3 Typedef Documentation . . . . .	1007
12.65.4 Enumeration Type Documentation . . . . .	1009
12.66 cfe/modules/core_api/fsw/inc/cfe_resourceid.h File Reference . . . . .	1010
12.66.1 Detailed Description . . . . .	1010
12.66.2 Macro Definition Documentation . . . . .	1011
12.66.3 Function Documentation . . . . .	1011
12.67 cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h File Reference . . . . .	1015
12.67.1 Detailed Description . . . . .	1015
12.67.2 Macro Definition Documentation . . . . .	1015
12.68 cfe/modules/core_api/fsw/inc/cfe_sb.h File Reference . . . . .	1016
12.68.1 Detailed Description . . . . .	1017
12.68.2 Macro Definition Documentation . . . . .	1018
12.69 cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h File Reference . . . . .	1018
12.69.1 Detailed Description . . . . .	1019
12.69.2 Macro Definition Documentation . . . . .	1019
12.69.3 Typedef Documentation . . . . .	1021
12.70 cfe/modules/core_api/fsw/inc/cfe_tbl.h File Reference . . . . .	1021
12.70.1 Detailed Description . . . . .	1022
12.71 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h File Reference . . . . .	1022
12.71.1 Detailed Description . . . . .	1024
12.71.2 Macro Definition Documentation . . . . .	1024
12.71.3 Typedef Documentation . . . . .	1024
12.71.4 Enumeration Type Documentation . . . . .	1024
12.72 cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h File Reference . . . . .	1025
12.72.1 Detailed Description . . . . .	1025
12.72.2 Macro Definition Documentation . . . . .	1025
12.72.3 Typedef Documentation . . . . .	1026
12.73 cfe/modules/core_api/fsw/inc/cfe_time.h File Reference . . . . .	1026
12.73.1 Detailed Description . . . . .	1027

12.73.2 Macro Definition Documentation . . . . .	1028
12.74 cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h File Reference . . . . .	1028
12.74.1 Detailed Description . . . . .	1028
12.74.2 Macro Definition Documentation . . . . .	1028
12.74.3 Typedef Documentation . . . . .	1029
12.74.4 Enumeration Type Documentation . . . . .	1029
12.75 cfe/modules/core_api/fsw/inc/cfe_version.h File Reference . . . . .	1029
12.75.1 Detailed Description . . . . .	1030
12.75.2 Macro Definition Documentation . . . . .	1030
12.76 cfe/modules/es/config/default_cfe_es_extern_typedefs.h File Reference . . . . .	1031
12.76.1 Detailed Description . . . . .	1033
12.76.2 Macro Definition Documentation . . . . .	1034
12.76.3 Typedef Documentation . . . . .	1034
12.76.4 Enumeration Type Documentation . . . . .	1038
12.77 cfe/modules/es/config/default_cfe_es_fcncodes.h File Reference . . . . .	1040
12.77.1 Detailed Description . . . . .	1040
12.77.2 Macro Definition Documentation . . . . .	1040
12.78 cfe/modules/es/config/default_cfe_es_interface_cfg.h File Reference . . . . .	1060
12.78.1 Detailed Description . . . . .	1061
12.78.2 Macro Definition Documentation . . . . .	1061
12.79 cfe/modules/es/config/default_cfe_es_internal_cfg.h File Reference . . . . .	1063
12.79.1 Detailed Description . . . . .	1065
12.79.2 Macro Definition Documentation . . . . .	1065
12.80 cfe/modules/es/config/default_cfe_es_mission_cfg.h File Reference . . . . .	1083
12.80.1 Detailed Description . . . . .	1083
12.81 cfe/modules/es/config/default_cfe_es_msg.h File Reference . . . . .	1083
12.81.1 Detailed Description . . . . .	1084
12.82 cfe/modules/es/config/default_cfe_es_msgdefs.h File Reference . . . . .	1084
12.82.1 Detailed Description . . . . .	1084
12.83 cfe/modules/es/config/default_cfe_es_msgids.h File Reference . . . . .	1084
12.83.1 Detailed Description . . . . .	1084
12.83.2 Macro Definition Documentation . . . . .	1084
12.84 cfe/modules/es/config/default_cfe_es_msgstruct.h File Reference . . . . .	1085
12.84.1 Detailed Description . . . . .	1088
12.84.2 Typedef Documentation . . . . .	1088
12.85 cfe/modules/es/config/default_cfe_es_platform_cfg.h File Reference . . . . .	1092
12.85.1 Detailed Description . . . . .	1093
12.86 cfe/modules/es/config/default_cfe_es_topicids.h File Reference . . . . .	1093
12.86.1 Detailed Description . . . . .	1093

---

12.86.2 Macro Definition Documentation . . . . .	1093
12.87 cfe/modules/es/fsw/inc/cfe_es_eventids.h File Reference . . . . .	1094
12.87.1 Detailed Description . . . . .	1097
12.87.2 Macro Definition Documentation . . . . .	1097
12.88 cfe/modules/evs/config/default_cfe_evs_extern_typedefs.h File Reference . . . . .	1119
12.88.1 Detailed Description . . . . .	1119
12.88.2 Typedef Documentation . . . . .	1119
12.88.3 Enumeration Type Documentation . . . . .	1120
12.89 cfe/modules/evs/config/default_cfe_evs_fcncodes.h File Reference . . . . .	1122
12.89.1 Detailed Description . . . . .	1122
12.89.2 Macro Definition Documentation . . . . .	1122
12.90 cfe/modules/evs/config/default_cfe_evs_interface_cfg.h File Reference . . . . .	1140
12.90.1 Detailed Description . . . . .	1140
12.90.2 Macro Definition Documentation . . . . .	1140
12.91 cfe/modules/evs/config/default_cfe_evs_internal_cfg.h File Reference . . . . .	1141
12.91.1 Detailed Description . . . . .	1141
12.91.2 Macro Definition Documentation . . . . .	1141
12.92 cfe/modules/evs/config/default_cfe_evs_mission_cfg.h File Reference . . . . .	1145
12.92.1 Detailed Description . . . . .	1145
12.93 cfe/modules/evs/config/default_cfe_evs_msg.h File Reference . . . . .	1145
12.93.1 Detailed Description . . . . .	1145
12.94 cfe/modules/evs/config/default_cfe_evs_msgdefs.h File Reference . . . . .	1145
12.94.1 Detailed Description . . . . .	1145
12.94.2 Macro Definition Documentation . . . . .	1146
12.95 cfe/modules/evs/config/default_cfe_evs_msgids.h File Reference . . . . .	1146
12.95.1 Detailed Description . . . . .	1147
12.95.2 Macro Definition Documentation . . . . .	1147
12.96 cfe/modules/evs/config/default_cfe_evs_msgstruct.h File Reference . . . . .	1147
12.96.1 Detailed Description . . . . .	1150
12.96.2 Typedef Documentation . . . . .	1150
12.97 cfe/modules/evs/config/default_cfe_evs_platform_cfg.h File Reference . . . . .	1154
12.97.1 Detailed Description . . . . .	1154
12.98 cfe/modules/evs/config/default_cfe_evs_topicids.h File Reference . . . . .	1154
12.98.1 Detailed Description . . . . .	1154
12.98.2 Macro Definition Documentation . . . . .	1154
12.99 cfe/modules/evs/fsw/inc/cfe_evs_eventids.h File Reference . . . . .	1155
12.99.1 Detailed Description . . . . .	1157
12.99.2 Macro Definition Documentation . . . . .	1157
12.100 cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h File Reference . . . . .	1167

12.100.1 Detailed Description . . . . .	1167
12.101 cfe/modules/fs/config/default_cfe_fs_filedef.h File Reference . . . . .	1167
12.101.1 Detailed Description . . . . .	1168
12.101.2 Typedef Documentation . . . . .	1168
12.101.3 Enumeration Type Documentation . . . . .	1168
12.102 cfe/modules/fs/config/default_cfe_fs_interface_cfg.h File Reference . . . . .	1169
12.102.1 Detailed Description . . . . .	1169
12.102.2 Macro Definition Documentation . . . . .	1169
12.103 cfe/modules/fs/config/default_cfe_fs_mission_cfg.h File Reference . . . . .	1170
12.103.1 Detailed Description . . . . .	1170
12.104 cfe/modules/msg/fsw/inc/ccsds_hdr.h File Reference . . . . .	1170
12.104.1 Detailed Description . . . . .	1170
12.104.2 Typedef Documentation . . . . .	1170
12.105 cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h File Reference . . . . .	1171
12.105.1 Detailed Description . . . . .	1171
12.106 cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h File Reference . . . . .	1171
12.106.1 Detailed Description . . . . .	1172
12.106.2 Macro Definition Documentation . . . . .	1172
12.107 cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h File Reference . . . . .	1172
12.107.1 Detailed Description . . . . .	1173
12.107.2 Macro Definition Documentation . . . . .	1173
12.107.3 Typedef Documentation . . . . .	1173
12.107.4 Enumeration Type Documentation . . . . .	1174
12.108 cfe/modules/sb/config/default_cfe_sb_fcncodes.h File Reference . . . . .	1174
12.108.1 Detailed Description . . . . .	1175
12.108.2 Macro Definition Documentation . . . . .	1175
12.109 cfe/modules/sb/config/default_cfe_sb_interface_cfg.h File Reference . . . . .	1184
12.109.1 Detailed Description . . . . .	1184
12.109.2 Macro Definition Documentation . . . . .	1184
12.110 cfe/modules/sb/config/default_cfe_sb_internal_cfg.h File Reference . . . . .	1185
12.110.1 Detailed Description . . . . .	1186
12.110.2 Macro Definition Documentation . . . . .	1186
12.111 cfe/modules/sb/config/default_cfe_sb_mission_cfg.h File Reference . . . . .	1193
12.111.1 Detailed Description . . . . .	1193
12.112 cfe/modules/sb/config/default_cfe_sb_msg.h File Reference . . . . .	1194
12.112.1 Detailed Description . . . . .	1194
12.113 cfe/modules/sb/config/default_cfe_sb_msgdefs.h File Reference . . . . .	1194
12.113.1 Detailed Description . . . . .	1194
12.114 cfe/modules/sb/config/default_cfe_sb_msgids.h File Reference . . . . .	1194

---

12.114.1 Detailed Description . . . . .	1194
12.114.2 Macro Definition Documentation . . . . .	1195
12.115 cfe/modules/sb/config/default_cfe_sb_msgstruct.h File Reference . . . . .	1195
12.115.1 Detailed Description . . . . .	1197
12.115.2 Typedef Documentation . . . . .	1197
12.116 cfe/modules/sb/config/default_cfe_sb_platform_cfg.h File Reference . . . . .	1200
12.116.1 Detailed Description . . . . .	1200
12.117 cfe/modules/sb/config/default_cfe_sb_topicids.h File Reference . . . . .	1200
12.117.1 Detailed Description . . . . .	1200
12.117.2 Macro Definition Documentation . . . . .	1200
12.118 cfe/modules/sb/fsw/inc/cfe_sb_eventids.h File Reference . . . . .	1201
12.118.1 Detailed Description . . . . .	1204
12.118.2 Macro Definition Documentation . . . . .	1204
12.119 cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h File Reference . . . . .	1220
12.119.1 Detailed Description . . . . .	1221
12.119.2 Typedef Documentation . . . . .	1221
12.119.3 Enumeration Type Documentation . . . . .	1221
12.120 cfe/modules/tbl/config/default_cfe_tbl_fcncodes.h File Reference . . . . .	1222
12.120.1 Detailed Description . . . . .	1222
12.120.2 Macro Definition Documentation . . . . .	1222
12.121 cfe/modules/tbl/config/default_cfe_tbl_interface_cfg.h File Reference . . . . .	1231
12.121.1 Detailed Description . . . . .	1231
12.121.2 Macro Definition Documentation . . . . .	1231
12.122 cfe/modules/tbl/config/default_cfe_tbl_internal_cfg.h File Reference . . . . .	1232
12.122.1 Detailed Description . . . . .	1233
12.122.2 Macro Definition Documentation . . . . .	1233
12.123 cfe/modules/tbl/config/default_cfe_tbl_mission_cfg.h File Reference . . . . .	1238
12.123.1 Detailed Description . . . . .	1238
12.124 cfe/modules/tbl/config/default_cfe_tbl_msg.h File Reference . . . . .	1238
12.124.1 Detailed Description . . . . .	1238
12.125 cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h File Reference . . . . .	1238
12.125.1 Detailed Description . . . . .	1239
12.126 cfe/modules/tbl/config/default_cfe_tbl_msgids.h File Reference . . . . .	1239
12.126.1 Detailed Description . . . . .	1239
12.126.2 Macro Definition Documentation . . . . .	1239
12.127 cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h File Reference . . . . .	1239
12.127.1 Detailed Description . . . . .	1242
12.127.2 Typedef Documentation . . . . .	1242
12.128 cfe/modules/tbl/config/default_cfe_tbl_platform_cfg.h File Reference . . . . .	1244

12.128.1 Detailed Description . . . . .	1244
12.129 cfe/modules/tbl/config/default_cfe_tbl_topicids.h File Reference . . . . .	1244
12.129.1 Detailed Description . . . . .	1244
12.129.2 Macro Definition Documentation . . . . .	1245
12.130 cfe/modules/tbl/fsw/inc/cfe_tbl_eventids.h File Reference . . . . .	1245
12.130.1 Detailed Description . . . . .	1248
12.130.2 Macro Definition Documentation . . . . .	1248
12.131 cfe/modules/time/config/default_cfe_time_extern_typedefs.h File Reference . . . . .	1265
12.131.1 Detailed Description . . . . .	1266
12.131.2 Typedef Documentation . . . . .	1267
12.131.3 Enumeration Type Documentation . . . . .	1268
12.132 cfe/modules/time/config/default_cfe_time_fcncodes.h File Reference . . . . .	1270
12.132.1 Detailed Description . . . . .	1271
12.132.2 Macro Definition Documentation . . . . .	1271
12.133 cfe/modules/time/config/default_cfe_time_interface_cfg.h File Reference . . . . .	1285
12.133.1 Detailed Description . . . . .	1286
12.133.2 Macro Definition Documentation . . . . .	1286
12.134 cfe/modules/time/config/default_cfe_time_internal_cfg.h File Reference . . . . .	1290
12.134.1 Detailed Description . . . . .	1290
12.134.2 Macro Definition Documentation . . . . .	1291
12.135 cfe/modules/time/config/default_cfe_time_mission_cfg.h File Reference . . . . .	1295
12.135.1 Detailed Description . . . . .	1295
12.136 cfe/modules/time/config/default_cfe_time_msg.h File Reference . . . . .	1295
12.136.1 Detailed Description . . . . .	1296
12.137 cfe/modules/time/config/default_cfe_time_msghDefs.h File Reference . . . . .	1296
12.137.1 Detailed Description . . . . .	1296
12.138 cfe/modules/time/config/default_cfe_time_msghIds.h File Reference . . . . .	1297
12.138.1 Detailed Description . . . . .	1297
12.138.2 Macro Definition Documentation . . . . .	1297
12.139 cfe/modules/time/config/default_cfe_time_msghStruct.h File Reference . . . . .	1298
12.139.1 Detailed Description . . . . .	1300
12.139.2 Typedef Documentation . . . . .	1300
12.140 cfe/modules/time/config/default_cfe_time_platform_cfg.h File Reference . . . . .	1303
12.140.1 Detailed Description . . . . .	1303
12.141 cfe/modules/time/config/default_cfe_time_topicids.h File Reference . . . . .	1303
12.141.1 Detailed Description . . . . .	1303
12.141.2 Macro Definition Documentation . . . . .	1303
12.142 cfe/modules/time/fsw/inc/cfe_time_eventids.h File Reference . . . . .	1305
12.142.1 Detailed Description . . . . .	1306

---

12.142.2 Macro Definition Documentation . . . . .	1306
12.143 osal/docs/src/osal_frontpage.dox File Reference . . . . .	1315
12.144 osal/docs/src/osal_fs.dox File Reference . . . . .	1315
12.145 osal/docs/src/osal_timer.dox File Reference . . . . .	1315
12.146 osal/src/os/inc/common_types.h File Reference . . . . .	1315
12.146.1 Detailed Description . . . . .	1316
12.146.2 Macro Definition Documentation . . . . .	1317
12.146.3 Typedef Documentation . . . . .	1317
12.146.4 Function Documentation . . . . .	1319
12.147 osal/src/os/inc/osapi-binsem.h File Reference . . . . .	1320
12.147.1 Detailed Description . . . . .	1321
12.148 osal/src/os/inc/osapi-bsp.h File Reference . . . . .	1321
12.148.1 Detailed Description . . . . .	1321
12.149 osal/src/os/inc/osapi-clock.h File Reference . . . . .	1321
12.149.1 Detailed Description . . . . .	1322
12.149.2 Enumeration Type Documentation . . . . .	1323
12.150 osal/src/os/inc/osapi-common.h File Reference . . . . .	1323
12.150.1 Detailed Description . . . . .	1324
12.150.2 Typedef Documentation . . . . .	1324
12.150.3 Enumeration Type Documentation . . . . .	1324
12.151 osal/src/os/inc/osapi-condvar.h File Reference . . . . .	1325
12.151.1 Detailed Description . . . . .	1326
12.152 osal/src/os/inc/osapi-constants.h File Reference . . . . .	1326
12.152.1 Detailed Description . . . . .	1326
12.152.2 Macro Definition Documentation . . . . .	1326
12.153 osal/src/os/inc/osapi-countsem.h File Reference . . . . .	1327
12.153.1 Detailed Description . . . . .	1327
12.154 osal/src/os/inc/osapi-dir.h File Reference . . . . .	1327
12.154.1 Detailed Description . . . . .	1328
12.154.2 Macro Definition Documentation . . . . .	1328
12.155 osal/src/os/inc/osapi-error.h File Reference . . . . .	1328
12.155.1 Detailed Description . . . . .	1330
12.155.2 Macro Definition Documentation . . . . .	1331
12.155.3 Typedef Documentation . . . . .	1331
12.156 osal/src/os/inc/osapi-file.h File Reference . . . . .	1331
12.156.1 Detailed Description . . . . .	1333
12.156.2 Macro Definition Documentation . . . . .	1333
12.156.3 Enumeration Type Documentation . . . . .	1334
12.157 osal/src/os/inc/osapi-filesystems.h File Reference . . . . .	1334

12.157.1 Detailed Description . . . . .	1335
12.157.2 Macro Definition Documentation . . . . .	1335
12.158 osal/src/os/inc/osapi-heap.h File Reference . . . . .	1336
12.158.1 Detailed Description . . . . .	1336
12.159 osal/src/os/inc/osapi-idmap.h File Reference . . . . .	1336
12.159.1 Detailed Description . . . . .	1337
12.159.2 Macro Definition Documentation . . . . .	1337
12.160 osal/src/os/inc/osapi-macros.h File Reference . . . . .	1338
12.160.1 Detailed Description . . . . .	1338
12.160.2 Macro Definition Documentation . . . . .	1338
12.161 osal/src/os/inc/osapi-module.h File Reference . . . . .	1339
12.161.1 Detailed Description . . . . .	1340
12.161.2 Macro Definition Documentation . . . . .	1340
12.162 osal/src/os/inc/osapi-mutex.h File Reference . . . . .	1341
12.162.1 Detailed Description . . . . .	1341
12.163 osal/src/os/inc/osapi-network.h File Reference . . . . .	1341
12.163.1 Detailed Description . . . . .	1342
12.164 osal/src/os/inc/osapi-printf.h File Reference . . . . .	1342
12.164.1 Detailed Description . . . . .	1342
12.165 osal/src/os/inc/osapi-queue.h File Reference . . . . .	1342
12.165.1 Detailed Description . . . . .	1343
12.166 osal/src/os/inc/osapi-select.h File Reference . . . . .	1343
12.166.1 Detailed Description . . . . .	1343
12.166.2 Enumeration Type Documentation . . . . .	1343
12.167 osal/src/os/inc/osapi-shell.h File Reference . . . . .	1344
12.167.1 Detailed Description . . . . .	1344
12.168 osal/src/os/inc/osapi-sockets.h File Reference . . . . .	1344
12.168.1 Detailed Description . . . . .	1346
12.168.2 Macro Definition Documentation . . . . .	1346
12.168.3 Enumeration Type Documentation . . . . .	1346
12.169 osal/src/os/inc/osapi-task.h File Reference . . . . .	1347
12.169.1 Detailed Description . . . . .	1348
12.169.2 Macro Definition Documentation . . . . .	1348
12.169.3 Typedef Documentation . . . . .	1348
12.169.4 Function Documentation . . . . .	1349
12.170 osal/src/os/inc/osapi-timebase.h File Reference . . . . .	1349
12.170.1 Detailed Description . . . . .	1349
12.170.2 Typedef Documentation . . . . .	1350
12.171 osal/src/os/inc/osapi-timer.h File Reference . . . . .	1350

12.171.1 Detailed Description . . . . .	1350
12.171.2 Typedef Documentation . . . . .	1350
12.172 osal/src/os/inc/osapi-version.h File Reference . . . . .	1351
12.172.1 Detailed Description . . . . .	1351
12.172.2 Macro Definition Documentation . . . . .	1351
12.172.3 Function Documentation . . . . .	1353
12.173 osal/src/os/inc/osapi.h File Reference . . . . .	1354
12.173.1 Detailed Description . . . . .	1355
12.174 psp/fsw/inc/cfe_psp.h File Reference . . . . .	1355
12.174.1 Macro Definition Documentation . . . . .	1357
12.174.2 Function Documentation . . . . .	1360
12.175 psp/fsw/inc/cfe_psp_error.h File Reference . . . . .	1368
12.175.1 Detailed Description . . . . .	1369
12.175.2 Macro Definition Documentation . . . . .	1369
12.175.3 Typedef Documentation . . . . .	1370
12.175.4 Function Documentation . . . . .	1371
<b>Index</b>	<b>1373</b>

# 1 CFS Checksum (CS) Documentation

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

## 1.1 CFS Checksum Introduction

### Scope

This document provides a complete specification for the commands and telemetry associated with the CFS Checksum (CS) application software. The document is intended primarily for users of the software (operations personnel, test engineers, and maintenance personnel).

The deployment guide section is intended for mission developers when deploying and configuring the application software for a mission flight software build environment.

[CFS Checksum 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
EEPROM	Electrically Erasable Programmable Read-Only Memory
FDS	Flight Data System
FM	File Manager
FSW	Flight Software
GN&C	Guidance Navigation & Control
GSFC	Goddard Space Flight Center
HK	Housekeeping
HW, H/W	Hardware
ICD	Interface Control Document
ISR	Interrupt Service Routine
OS	Operating System
OSAL	Operating System Abstraction Layer
Pkts	Packets
RAM	Random-Access Memory
RTOS	Real Time Operating System
RTP	Relative Time Processor
RTS	Relative Time tagged command Sequence
SB	Software Bus Service
SBC	Single Board Computer
SC	Stored Commands task

SW, S/W	Software
TBD	To Be Determined
TBL	Table
TLM	Telemetry
UTC	Universal time code

## 1.2 CFS Checksum Overview

The Checksum (CS) application of the Core Flight System (CFS) is responsible for ensuring the integrity of onboard memory. CS has the ability to ensure the integrity of Core Flight Executive (cFE) applications, cFE tables, the cFE core, the onboard operating system (OS), onboard EEPROM, as well as any memory regions ("Memory") specified by the users.

All CS checksums regions are configurable allowing users to specify the enable/disable state for each region upon startup.

The cFE core and the OS checksum regions are enabled by default. These are the only two checksum regions that are not table driven. Additionally, the user may select other table driven areas to checksum, including:

- **EEPROM:** A contiguous block of memory specified by a starting address and number of bytes.
- **Memory:** A contiguous block of (non-EEPROM) memory specified by a starting address and number of bytes.
- **Tables:** Any cFE or CFS application table, specified by the table name.
- **Applications:** Any cFE or CFS application, specified by the app name.

All table driven checksum regions are also enabled by default.

The CS application is both table and schedule driven. All of the applications, tables, areas of EEPROM, and other memory areas to be checksummed must be defined by the user in four separate tables that can be controlled independently. For each checksum region that is enabled, CS will checksum up to [CS\\_DEFAULT\\_BYTES\\_PER\\_CYCLE](#) (a configuration parameter) bytes every time CS receives its background cycle MID from a scheduling application/source.

The CS application uses two sets of tables: a set of definition tables which the user populates on the ground with the set of applications, tables, areas of EEPROM and other memory areas, and a set of results tables which are dump only tables. The results tables are populated by CS automatically from the definition tables, but also contain more information. The results tables contain the checksum values for each object in the table as well as other information for CS to use during runtime. For more information see [CFS Checksum Table Definitions](#).

## 1.3 CFS Checksum Operation

### Initialization

Upon initialization CS will initialize its global memory area and proceed to populate its results tables (which are dump only tables) from the default definition tables located in non-volatile memory. Each table has a corresponding setting in the platform configuration of the enable/disabled state of each table, the OS code segments and cFE Core for checksumming. These values are used at initialization for power-on reset or if CS is not configured to preserve states via the CDS they will also be used for processor-reset. If CS is configured to preserve enable state of the tables on a proc reset, states from CDS will be used upon processor reset.

If the default definition tables do not pass table validation, or they are nonexistent, blank definition tables are used from CS's memory and the table is disabled for checksumming. The purpose of loading a blank table from memory is so that another table can be loaded from the ground and be functional. The other purpose of loading a blank table from memory is so that if the table is accidentally enabled for checksumming, the table pointers for the results table point to good data.

Since CS cannot operate without the operating system or the cFE core, CS automatically sets up checksumming for those areas and enables them for checksumming.

### CDS Usage

If the CS application is configured to preserve states across processor resets, the enable/disable state of each individual checksum region will be stored in the CDS. The CDS is updated each time an Enable or Disable ground command is processed for a specific checksum region.

CS also includes a ground command to enable or disable all checksumming. This overall setting is not saved to the CDS.

## 1.4 CFS Checksum Commands

[CFS Checksum Command Message IDs](#)

[CFS Checksum Command Structures](#)

[CFS Checksum Command Codes](#)

## 1.5 CFS Checksum Telemetry

[CFS Checksum Telemetry Message IDs](#)

[CFS Checksum Telemetry](#)

## 1.6 CFS Checksum Events

[CFS Checksum Event IDs](#)

## 1.7 CFS Checksum Deployment Guide

To integrate the CS application with the cFE, follow the CFS App Integration Guide.

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

There are two message IDs that must be included in the CFS Scheduler Table: [CS\\_SEND\\_HK\\_MID](#) is sent out at the housekeeping request interval. [CS\\_BACKGROUND\\_CYCLE\\_MID](#) is sent out at the desired rate for background checksumming.

The Checksum app generates telemetry when it receives the housekeeping request. Its telemetry message ID is [CS\\_HK\\_TLM\\_MID](#).

The ES app uses the CS performance ID, [CS\\_APPMAIN\\_PERF\\_ID](#), to keep track of the performance of the CS app.

The mission configuration file [cs\\_mission\\_cfg.h](#) contains mission-level parameters that can be adjusted across all platforms.

See [CFS Checksum Mission Configuration](#)

The platform configuration file [cs\\_platform\\_cfg.h](#) contains parameters that can be adjusted to specific platforms. The defined parameters (and their default values) are:

See [CFS Checksum Platform Configuration](#)

## 1.8 CFS Checksum Configuration

[CFS Checksum Mission Configuration](#)

[CFS Checksum Platform Configuration](#)

## 1.9 CFS Checksum Table Definitions

The Checksum application uses eight tables (four definition tables and four corresponding results tables). The CS App copies the data from each definition table at initialization and during table updates, and stores the data in the corresponding results tables. The CS app accesses this data from each results table for processing, and saves its state in the same table.

The definitions tables are load-dump tables, and the results tables are dump-only.

### EEPROM Definition Table and Memory Definition Table

There are [CS\\_MAX\\_NUM\\_EEPROM\\_TABLE\\_ENTRIES](#) entries defined for the EEPROM Definition Table, and [CS\\_MAX\\_NUM\\_MEMORY\\_TABLE\\_ENTRIES](#) entries defined for the Memory Definition Table.

The format of a single entry in the EEPROM Definition Table or the Memory Definition Table is defined by [CS\\_Def\\_EepromMemory\\_Table\\_Entry\\_t](#).

### Tables Definition Table

There are [CS\\_MAX\\_NUM\\_TABLES\\_TABLE\\_ENTRIES](#) entries defined for the Tables Definition Table.

The format of a single entry in the Tables Definition Table is defined by [CS\\_DefTablesTableEntry\\_t](#).

### Apps Definition Table

There are [CS\\_MAX\\_NUM\\_APP\\_TABLE\\_ENTRIES](#) entries defined for the Apps Definition Table.

The format of a single entry in the Apps Definition Table is defined by [CS\\_DefAppTableEntry\\_t](#).

### EEPROM Results Table and Memory Results Table

There are [CS\\_MAX\\_NUM\\_EEPROM\\_TABLE\\_ENTRIES](#) entries defined for the EEPROM Results Table, and [CS\\_MAX\\_NUM\\_MEMORY\\_TABLE\\_ENTRIES](#) entries defined for the Memory Results Table.

The format of a single entry in the EEPROM Results Table or the Memory Results Table is defined by [CS\\_ResEepromMemoryTableEntry\\_t](#).

### Tables Results Table

There are [CS\\_MAX\\_NUM\\_TABLES\\_TABLE\\_ENTRIES](#) entries defined for the Tables Results Table.

The format of a single entry in the Tables Results Table is defined by [CS\\_ResTablesTableEntry\\_t](#).

### Apps Results Table

There are [CS\\_MAX\\_NUM\\_APP\\_TABLE\\_ENTRIES](#) entries defined for the Apps Results Table.

The format of a single entry in the Apps Results Table is defined by [CS\\_ResAppTableEntry\\_t](#).

## 1.10 CFS Checksum Constraints

### Child Tasks

The Checksum application only allows one child task to be used at a time. One child task is used during each of the [CS\\_\\*Recompute](#) commands and during the [CS OneShot](#) command. A child task is started when the command returns successfully (with a debug event message), and then ends when the child task finishes the recompute (an info event message is generated). While it is not possible to cancel a Recompute command's child task, the functionality is provided to cancel a One Shot command's child task, because any length can be specified in the command, and the checksum may take longer than expected.

What this means is that when a Recompute command (or One Shot command) is sent, the user has to wait until completion of the checksum to send another Recompute (or One Shot) command can be sent. If another Recompute (or One Shot) command is sent while the child task is in use, the command will get rejected.

### Use On OS X and Linux

Because of the way these two OS's are set up, it is not possible to checksum applications or the OS text segment with these two operating systems. CS will perform the rest of its functions without incident, but will issue an informational event message on startup when it determines the OS cannot be checksummed, and issue a debug event message.

### Checksum application's tables

It is worth a note to say that if the user wants CS's tables to be checksummed, they must be put in CS's table of tables to be checksummed.

### Checksumming tables

When creating CS's Table of tables to be checksummed, the 'Name' field must be the same name that cFE Table Services uses. That is it must be in the form of 'AppName.TableName', otherwise CS will not be able to find the table and it will not be checksummed.

## 1.11 CFS Checksum Frequently Asked Questions

**(Q) Can I set the max number of table entries to zero to not use checksumming of apps (or tables or EEPROM or user defined memory) ?**

*No. The maximum number of entries must be at least one. This is to prevent null pointers from ever becoming an issue in CS. If you don't want to checksum a certain area ever, than the max can be set to 1 and no default table loaded, so CS will just use its default blank table and disable checksumming for that area.*

**(Q) What is the procedure for updating a table that is being checksummed?**

*The recommended procedure for updating a table that is being checksummed is to first disable the checksumming of that table, then load the new table via CFE TBL, then issue the CS Recompute Checksum command on that table, and finally re-enable the checksumming of that table. What this procedure does is ensures that no checksum miscompute errors get generated.*

**(Q) What do I do if I want to see the checksums of my applications/tables/etc?**

*The only way to see what the actual checksum value (Comparison Value) of an object is is to dump the results table that contains that object. For CS, knowing what the Comparison Value is isn't important for routine activities, only knowing when the Comparison Value has changed is important. That being said, having the Comparison Values on hand may be necessary, hence they are available by dumping the table.*

## 2 Core Flight Executive Documentation

- General Information and Concepts
  - [Background](#)
  - [Applicable Documents](#)
  - [Version Numbers](#)
  - [Dependencies](#)
  - [Acronyms](#)
  - [Glossary of Terms](#)
- Executive Services (ES)

- [cFE Executive Services Overview](#)
- [cFE Executive Services Commands](#)
- [cFE Executive Services Telemetry](#)
- [ES Event Message Reference](#)
- [cFE Executive Services Configuration Parameters](#)
- Events Services (EVS)
  - [cFE Event Services Overview](#)
  - [cFE Event Services Commands](#)
  - [cFE Event Services Telemetry](#)
  - [EVS Event Message Reference](#)
  - [cFE Event Services Configuration Parameters](#)
- Software Bus Services (SB)
  - [cFE Software Bus Overview](#)
  - [cFE Software Bus Commands](#)
  - [cFE Software Bus Telemetry](#)
  - [SB Event Message Reference](#)
  - [cFE Software Bus Configuration Parameters](#)
- Table Services (TBL)
  - [cFE Table Services Overview](#)
  - [cFE Table Services Commands](#)
  - [cFE Table Services Telemetry](#)
  - [TBL Event Message Reference](#)
  - [cFE Table Services Configuration Parameters](#)
- Time Services (TIME)
  - [cFE Time Services Overview](#)
  - [cFE Time Services Commands](#)
  - [cFE Time Services Telemetry](#)
  - [TIME Event Message Reference](#)
  - [cFE Time Services Configuration Parameters](#)
- [cFE Event Message Cross Reference](#)
- [cFE Command Mnemonic Cross Reference](#)
- [cFE Telemetry Mnemonic Cross Reference](#)
- [cFE Application Programmer's Interface \(API\) Reference](#)

## 2.1 Background

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

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

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

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

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

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

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

### 2.1.1 Core Flight Executive (cFE) Goals

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

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

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

## 2.2 Applicable Documents

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

## 2.3 Version Numbers

### 2.3.1 Version Number Semantics

The version number is a sequence of four numbers, generally separated by dots when written. These are, in order, the Major number, the Minor number, the Revision number, and the Mission Revision number.

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

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

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

The REVISION number shall be incremented on changes that benefit from unique identification such as bug fixes or major documentation updates. The Revision number may also be updated if there are other changes contained within a release that make it desirable for applications to distinguish one release from another. **WARNING:** The revision number is set to the number 99 in development builds. To distinguish between development builds refer to the BUILD\_NUMBER and BUILD\_BASELINE detailed in the section "Identifying Development Builds".

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

### 2.3.2 How and Where Defined

The version numbers are provided as simple macros defined in the [cfi\\_version.h](#) header file as part of the API definition; these macros must expand to simple integer values, so that they can be used in simple if directives by the macro preprocessor.

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

### 2.3.3 Identifying Development Builds

In order to distinguish between development versions, we also provide a BUILD\_NUMBER.

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

### 2.3.4 Templates for the short and long version string

See [cfi\\_version.h](#) for the standard layout and definition of version information. The apps and repositories follow the same pattern by replacing the CFE\_ prefix with the appropriate name; for example, osal uses OS\_, psp uses CFE\_P←SP\_IMPL, and so on.

Suggested pattern for development:

- CFSCOMPONENT\_SRC\_VERSION: REFERENCE\_GIT\_TAG"+dev"BUILD\_NUMBER
  - Example: "v6.8.0-rc1+dev123"
- CFSCOMPONENT\_VERSION\_STRING: "CFSCOMPONENT DEVELOPMENT BUILD "CFSCOMPONENT\_S←RC\_VERSION" (Codename: CFSCONSTELLATION), Last Official Release: MAJOR.MINOR.REVISION"
  - Example: "cFE DEVELOPMENT BUILD v6.8.0-rc1+dev123 (Codename: Bootes), Last Official Release: cfe v6.7.0"

Suggested pattern for official releases:

- CFSCOMPONENT\_SRC\_VERSION: OFFICIAL\_GIT\_TAG
  - Example: "v7.0.0"
- COMPONENT\_VERSION\_STRING: "CFSCOMPONENT OFFICIAL RELEASE "CFSCOMPONENT\_SRC\_V←RSION" (Codename: CFSCONSTELLATION)"
  - Example: "cFE OFFICIAL RELEASE v7.0.0 (Codename: Caelum)"

## 2.4 Dependencies

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

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

## 2.5 Acronyms

Acronym	Description
AC	Attitude Control
ACE	Attitude Control Electronics
ACS	Attitude Control System
API	Application Programming Interface
APID	CCSDS Application ID
App	Application
CCSDS	Consultative Committee for Space Data Systems
CDH, C&DH	Command and Data Handling
cFE	core Flight Executive
cFS	core Flight System
CM	Configuration Management
CMD	Command
CPU	Central Processing Unit
EDAC	Error Detection and Correction
EEPROM	Electrically Erasable Programmable Read-Only Memory
ES	Executive Services
EVS	Event Services
FC	Function Code
FDC	Failure Detection and Correction
FSW	Flight Software
HW, H/W	Hardware
ICD	Interface Control Document
MET	Mission Elapsed Time
MID	Message ID
OS	Operating System
OSAL	Operating System Abstraction Layer
PID	Pipeline ID
PKT	Packet
PSP	Platform Support Package
RAM	Random-Access Memory
SB	Software Bus
SDO	Solar Dynamics Observatory
ST5	Space Technology Five

Acronym	Description
STCF	Spacecraft Time Correlation Factor
SW, S/W	Software
TAI	International Atomic Time
TBD	To Be Determined
TBL	Table Services
TID	Task ID
TIME	Time Services
TLM	Telemetry
UTC	Coordinated Universal Time

## 2.6 cFE Executive Services Overview

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

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

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

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

- [Terminology](#)
- [Software Reset](#)
  - [Reset Types and Subtypes](#)
  - [Exception and Reset \(ER\) Log](#)
- [Application and Child Task Management](#)
  - [Starting an Application](#)
  - [Stopping an Application](#)
  - [Restarting an Application](#)
  - [Reloading an Application](#)

- Listing Current Applications
- Listing Current Tasks
- Loading Common Libraries
- Basic File System
- Performance Data Collection
- Critical Data Store
- Memory Pool
- System Log
- Version Identification
- Frequently Asked Questions about Executive Services

### 2.6.1 Terminology

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

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

#### 2.6.1.1 "Application" and "cFE Application"

##### Application

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

##### cFE Application

A 'cFE Application' is an application that is external to the cFE and designed to interface to the cFE through the APIs. It is created through an entry in the ["Startup Script"](#) (with the 'Object Type' field set to CFE\_APP) or by way of the [CFE\\_ES\\_START\\_APP\\_CC](#) ground command.

When referring to one of the five applications internal to the cFE (ES, EVS, SB, TIME or TBL), the term 'Service' or 'Core Application' is typically used.

A listing of cFE applications can be acquired by using the [CFE\\_ES\\_QUERY\\_ALL\\_CC](#) ground command. This listing will include the cFE internal applications as well as cFE applications that are loaded and running.

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

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

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

**2.6.1.3 "Startup Script"** The startup script is a text file, written by the user that contains a list of entries (one entry for each application) and is used by the ES application for automating the startup of applications. For a processor reset, ES checks for the CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE first, and if it doesn't exist or for a power on reset ES uses the file passed in to [CFE\\_ES\\_Main](#) (typically CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE but dependent on the PSP).

The fields in a single entry include:

Object Type	CFE_APP for an Application, or CFE_LIB for a library.
Path/Filename	This is a cFE Virtual filename, not a vxWorks device pathname
Entry Point	This is the name of the "main" function for App.
CFE Name	The cFE name for the APP or Library
Priority	This is the Priority of the App, not used for a Library
Stack Size	This is the Stack size for the App, not used for a Library
Load Address	This is the Optional Load Address for the App or Library. It is currently not implemented so it should always be 0x0.
Exception Action	<p>This is the Action the cFE should take if the Application has an exception.</p> <ul style="list-style-type: none"> <li>• 0 = Do a cFE Processor Reset</li> <li>• Non-Zero = Just restart the Application</li> </ul>

Immediately after the cFE completes its initialization, the ES Application first looks for the volatile startup script. The location in the file system is defined by the cFE platform configuration parameter named [CFE\\_PLATFORM\\_ES\\_VOLATILE\\_STARTUP\\_FILE](#). This configuration parameter contains a path as well as a filename. If the file is found, ES begins to startup the applications that are listed in the file. If ES does not find the file, it attempts to open the [CFE\\_PLATFORM\\_ES\\_NONVOL\\_STARTUP\\_FILE](#).

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

Any errors encountered in the startup script processing are written to the [System Log](#). The [System Log](#) may also contain positive acknowledge messages regarding the startup script processing.

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

## 2.6.2 Software Reset

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

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

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

## 2.6.3 Reset Types and Subtypes

The Reset Type is sent to the ground in the ES housekeeping packet and tells how the current running version of the cFE was invoked. The possible Reset Types expected in the telemetry field are [CFE\\_PSP\\_RST\\_TYPE\\_POWERON](#) and [CFE\\_PSP\\_RST\\_TYPE\\_PROCESSOR](#). There is a third Reset Type defined in the ES code as [CFE\\_ES\\_APP\\_RESTART](#) which applies only to restarting an individual application and is covered in more detail in the section titled Application and Child Task.

The Reset Subtype is also sent in the ES housekeeping packet and gives more detail about the type of reset that started the execution of the current running version of the cFE. The possible Reset Subtypes are [CFE\\_PSP\\_RST\\_SUBTYPE\\_POWER\\_CYCLE](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_PUSH\\_BUTTON](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_HW\\_SPECIFIC](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_HW\\_WATCHDOG](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_RESET\\_COMMAND](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_EXCEPTION](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_UNDEFINED\\_RESET](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_HWDEBUG\\_RESET](#), [CFE\\_PSP\\_RST\\_SUBTYPE\\_BAN](#).

## 2.6.4 Exception and Reset (ER) Log

The Exception and Reset Log contains detailed information about past resets and exceptions. To view the information the [CFE\\_ES\\_WRITE\\_ER\\_LOG\\_CC](#) command must be sent. This command will write the log to a binary file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_ER\\_LOG\\_FILE](#) is used to specify the path and filename. Use the ground system to get the file and display the contents. There is also a command to clear the ER log, [CFE\\_ES\\_CLEAR\\_ER\\_LOG\\_CC](#).

The size of the ER log is defined by the platform configuration parameter [CFE\\_PLATFORM\\_ES\\_ER\\_LOG\\_ENTRIES](#). This log is preserved after a processor reset and held in the ES reset area.

A count of the number of entries in the log is present in the ES housekeeping telemetry. This count can be used with the configuration parameter [CFE\\_PLATFORM\\_ES\\_ER\\_LOG\\_ENTRIES](#) to calculate the fullness of the log.

The information contained in a single log entry is defined by the structure [CFE\\_ES\\_ERLog\\_t](#).

### 2.6.5 Application and Child Task Management

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

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

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

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

### 2.6.6 Starting an Application

There are two ways to start an application, through the ground command [CFE\\_ES\\_START\\_APP\\_CC](#) or through the startup script. In either case, the object file must be loaded on board before the command is sent or before the startup script is executed. The startup script contains a list of applications and library routines to load and start immediately after the cFE finishes its startup sequence. The parameters in the command, match the elements of an entry in the startup script.

The format of the Start Application command, is defined in the structure [CFE\\_ES\\_StartAppCmd\\_t](#). The members of the structure include, application name, entry point, filename, stack size, load address, exception action and priority.

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

After starting an application, the ES task sends an informational event message displaying the application name, file-name of the object and the application ID. The new application will then show up in the query list downloaded in response to the [CFE\\_ES\\_QUERY\\_ALL\\_CC](#) command.

### 2.6.7 Stopping an Application

Stopping an application can be done through the ground command [CFE\\_ES\\_STOP\\_APP\\_CC](#). This command will terminate the application execution and all child tasks created by the application, free the system resources that it allocated and delete the corresponding object file.

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

The format of the Stop Application command, is defined in the structure [CFE\\_ES\\_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.

### 2.6.8 Restarting an Application

The [CFE\\_ES\\_RESTART\\_APP\\_CC](#) command is used to restart an application using the same file name as the last start.

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

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

### 2.6.9 Reloading an Application

The [CFE\\_ES\\_RELOAD\\_APP\\_CC](#) command is used to reload an application using a new file name.

This command performs the same actions as [CFE\\_ES\\_RESTART\\_APP\\_CC](#) only using the new file.

### 2.6.10 Listing Current Applications

There are two options for receiving information about applications, the [CFE\\_ES\\_QUERY\\_ONE\\_CC](#) command can be used to get details about a single application. This command takes an application name as its only parameter and the application information is sent as a software bus packet that can be telemetered to the ground.

Or the [CFE\\_ES\\_QUERY\\_ALL\\_CC](#) command can be used to get information about all the applications that are currently registered with ES. This command writes the application data to a file and has a one parameter which specifies the path and filename of the output file.

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

- **Application ID** - The Application ID assigned by the cFE to the Application
- **Type Identifier** - Identifies whether the Application is a CORE App or an EXTERNAL App
- **Name** - The Application Name
- **Entry Point** - The symbolic name for the entry point into the Application
- **Filename** - The name of the file the Application was loaded from
- **Stack Size** - The number of bytes allocated for the Application's stack
- **Load Address** - The starting address of memory where the Application was loaded

- **Load Size** - The size, in bytes, of the Application when loaded into memory
- **Start Address** - The physical address that maps to the Entry Point
- **Exception Action** - A flag that identifies whether the Processor should undergo a Restart or whether just the Application should restart upon an exception condition within the Application
- **Priority** - The assigned priority for the Application
- **Main Task ID** - The Task ID assigned to the main task associated with the Application
- **Main Task Name** - The name of the main task associated with the Application
- **Number of Child Tasks** - The number of child tasks spawned by the main task

For a description of the format in which this data is dumped, see [CFE\\_ES\\_AppInfo\\_t](#).

#### 2.6.11 Listing Current Tasks

The [CFE\\_ES\\_QUERY\\_ALL\\_TASKS\\_CC](#) command is used to get a list of child tasks that are currently registered with ES. The following information is provided for each registered task:

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

#### 2.6.12 Loading Common Libraries

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

#### 2.6.13 Basic File System

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

## 2.6.14 Performance Data Collection

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

API calls are inserted by the development team at key points in the code. The basic operation is to start the data collection, wait some amount of time, then send the command to stop the data collection. When the stop command is received, the ES task writes all the data from the buffer to a file. The file can then be imported to analysis tools for viewing. The size of the buffer is configurable through the [CFE\\_PLATFORM\\_ES\\_PERF\\_DATA\\_BUFFER\\_SIZE](#) platform configuration parameter.

Additional information follows:

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

**2.6.14.1 Performance Data Collection Trigger Masks** The trigger mask is used to control precisely when to start collecting the data. There is a bit in the trigger mask for every marker used in the code. After a start command is received, the trigger mask is read and dictates when to begin storing data in the buffer.

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

**2.6.14.2 Starting to Collect Performance Data** The [CFE\\_ES\\_START\\_PERF\\_DATA\\_CC](#) command is used to start the data collection process. The ES task sends a debug event when the command is received. It is not possible to start a collection if the buffer-to-file write is in process from an earlier collection. There is an ES telemetry point that can be used to ensure there is not a buffer-to-file write in progress. This ES telemetry point is called 'Perf Data to Write' and begins counting down from 'Data Count' to zero. If this counter is zero, it is ok to send the start command. If any errors are encountered when the start command is received, the details will be displayed in an error event message.

**2.6.14.3 Stopping the Collection of Performance Data** The [CFE\\_ES\\_STOP\\_PERF\\_DATA\\_CC](#) command is used to stop the data collection process and write the buffer data to a file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_PERF\\_DUMP\\_FILENAME](#) is used to specify the path and filename. The number of entries written to the file is determined by the 'data count' variable, which is sent in the ES housekeeping telemetry packet. To ensure cpu hogging does not occur during the write process, ES creates a low priority child task to perform the file write operation. This child task will write a number of entries, then sleep for a short time to give tasks of lower priority a chance to run. The number of entries between delays, and the delay time is displayed in the debug event at the time the stop command is received.

**2.6.14.4 Viewing the Collection of Performance Data** To view the performance data, the file created as a result of the stop command must be transferred to the ground and imported into a viewing tool. See <https://github.com/nasa/perfutils-java> as an example.

## 2.6.15 Critical Data Store

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

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

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

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

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

The format of the CDS Registry Dump File is a cFE Standard File header (see [CFE\\_FS\\_Header\\_t](#)) followed by one or more CDS Registry Dump File Records (see [CFE\\_ES\\_CDSRegDumpRec\\_t](#)).

## 2.6.16 Memory Pool

Refer to the cFE Application Developers Guide for additional information.

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

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

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

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

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

This algorithm is recommended when the size of the requests and the peak rate of requests can be pre-determined. It is highly recommended that adequate margin is designed into the pool size. The memory pool should never get close to being fully configured (i.e. not enough memory to create a new block). If the memory does become fully configured, requests for new size blocks will fail, regardless of whether the created blocks are in-use or not. The margin on the memory pool can be monitored by viewing the 'free bytes' member of the memory pool statistics. The memory pool statistics are dumped only when commanded by way of the ES command [CFE\\_ES\\_SEND\\_MEM\\_POOL\\_STATS\\_CC](#).

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

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

If the defaults are not sufficient, the user must define the block sizes and use the [CFE\\_ES\\_PoolCreateEx](#) API.

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

If another request for that size comes in before the first block was released through the [CFE\\_ES\\_PutPoolBuf](#) API, then the memory pool will create a second block of that size and return a pointer to the second block. If both blocks were then released through the [CFE\\_ES\\_PutPoolBuf](#) API and the memory pool statistics were dumped via the [CFE\\_ES\\_SEND\\_MEM\\_POOL\\_STATS\\_CC](#) command, the number of blocks created would be two. The number of 'free bytes' in the pool would be the size of the pool minus the sum of the following items:

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

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

An operator can obtain information about an Application's Memory Pool by using the [Telemeter Memory Pool Statistics Command](#).

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

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

The [Memory Pool Statistics Telemetry Packet](#) contains the following information:

- **Memory Pool Handle** - the handle, as provided by the operator in the [Telemeter Memory Pool Statistics Command](#). This repeating of the handle in telemetry ensures the operator knows which Memory Pool Statistics are being viewed
- **Pool Size** - The total size of the memory pool (in bytes)
- **Number Blocks Requested** - The total number of memory blocks requested for allocation
- **Number of Errors** - The total number of errors encountered when a block was released
- **Number of Free Bytes** - The total number of bytes in the Memory Pool that have never been allocated to a Memory Block
- **Block Statistics** - For each specified size of memory block (of which there are `CFE_MISSION_ES_POOL_MAX_BUCKETS`), the following statistics are kept
  - **Block Size** - The size, in bytes, of all blocks of this type
  - **Number of Blocks Allocated** - The number of this sized block which are currently allocated and in use
  - **Number of Blocks Free** - The number of this size block which have been in use previously but are no longer being used

### 2.6.17 System Log

The System Log is an array of bytes that contains back-to-back printf type messages from applications. The cFE internal applications use this log when errors are encountered during initialization before the Event Manager is fully initialized. To view the information the [CFE\\_ES\\_WRITE\\_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.

### 2.6.18 Version Identification

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

### 2.6.19 Frequently Asked Questions about Executive Services

None submitted

## 2.7 cFE Executive Services Commands

Upon receipt of any command, the Executive Services application will confirm that the message length embedded within the header (from [CFE\\_MSG\\_GetSize\(\)](#)) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, ES will generate the [CFE\\_ES\\_LEN\\_ERR\\_EID](#) event, increment the command error counter ( $\$sc\_scpu\_ES\_CMDEC$ ), 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

Maximum Length of Full CDS Name in messages

### Global `CFE_MISSION_ES_CDS_MAX_NAME_LENGTH`

Maximum Length of CDS Name

Maximum Length of CDS Name

### Global `CFE_MISSION_ES_DEFAULT_CRC`

Mission Default CRC algorithm

Mission Default CRC algorithm

### Global `CFE_MISSION_ES_MAX_APPLICATIONS`

Mission Max Apps in a message

Mission Max Apps in a message

### Global `CFE_MISSION_ES_PERF_MAX_IDS`

Define Max Number of Performance IDs for messages

Define Max Number of Performance IDs for messages

### Global `CFE_MISSION_ES_POOL_MAX_BUCKETS`

Maximum number of block sizes in pool structures

Maximum number of block sizes in pool structures

**Global CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC**

CFE core application startup timeout

**Global CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT**

Define ES Application Kill Timeout

Define ES Application Kill Timeout

**Global CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE**

Define ES Application Control Scan Rate

Define ES Application Control Scan Rate

**Global CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES**

Define Maximum Number of Registered CDS Blocks

Define Maximum Number of Registered CDS Blocks

**Global CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01**

Define ES Critical Data Store Memory Pool Block Sizes

Define ES Critical Data Store Memory Pool Block Sizes

**Global CFE\_PLATFORM\_ES\_CDS\_SIZE**

Define Critical Data Store Size

Define Critical Data Store Size

**Global CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE**

Default Application Information Filename

Default Application Information Filename

**Global CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE**

Default Critical Data Store Registry Filename

Default Critical Data Store Registry Filename

**Global CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE**

Default Exception and Reset (ER) Log Filename

Default Exception and Reset (ER) Log Filename

**Global CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME**

Default Performance Data Filename

Default Performance Data Filename

**Global CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE**

Define Default System Log Mode following Power On Reset

Define Default System Log Mode following Power On Reset

**Global CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE**

Define Default System Log Mode following Processor Reset

Define Default System Log Mode following Processor Reset

**Global CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE**

Define Default Stack Size for an Application

Define Default Stack Size for an Application

**Global CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE**

Default System Log Filename

Default System Log Filename

**Global CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE**

Default Application Information Filename

Default Application Information Filename

**Global CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES**

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

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

**Global CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE**

Maximum size of CPU Context in ES Error Log

Maximum size of CPU Context in ES Error Log

**Global CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS**

Define Max Number of Applications

Define Max Number of Applications

**Global CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS**

Define Max Number of Generic Counters

Define Max Number of Generic Counters

**Global CFE\_PLATFORM\_ES\_MAX\_LIBRARIES**

Define Max Number of Shared libraries

Define Max Number of Shared libraries

**Global CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS**

Maximum number of memory pools

Maximum number of memory pools

**Global CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS**

Define Number of Processor Resets Before a Power On Reset

Define Number of Processor Resets Before a Power On Reset

**Global CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01**

Define Default ES Memory Pool Block Sizes

Define Default ES Memory Pool Block Sizes

**Global CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN**

Define Memory Pool Alignment Size

Define Memory Pool Alignment Size

**Global CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING**

Default virtual path for persistent storage

Default virtual path for persistent storage

**Global CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE**

ES Nonvolatile Startup Filename

ES Nonvolatile Startup Filename

**Global CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE**

Define Number of entries in the ES Object table

Define Number of entries in the ES Object table

**Global CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY**

Define Performance Analyzer Child Task Delay

Define Performance Analyzer Child Task Delay

**Global CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY**

Define Performance Analyzer Child Task Priority

Define Performance Analyzer Child Task Priority

**Global CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE**

Define Performance Analyzer Child Task Stack Size

Define Performance Analyzer Child Task Stack Size

**Global CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE**

Define Max Size of Performance Data Buffer

Define Max Size of Performance Data Buffer

**Global CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS**

Define Performance Analyzer Child Task Number of Entries Between Delay

Define Performance Analyzer Child Task Number of Entries Between Delay

**Global CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL**

Define Filter Mask Setting for Enabling All Performance Entries

Define Filter Mask Setting for Enabling All Performance Entries

**Global CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT**

Define Default Filter Mask Setting for Performance Data Buffer

Define Default Filter Mask Setting for Performance Data Buffer

**Global CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE**

Define Filter Mask Setting for Disabling All Performance Entries

Define Filter Mask Setting for Disabling All Performance Entries

**Global CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL**

Define Filter Trigger Setting for Enabling All Performance Entries

Define Filter Trigger Setting for Enabling All Performance Entries

**Global CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT**

Define Default Filter Trigger Setting for Performance Data Buffer

Define Default Filter Trigger Setting for Performance Data Buffer

**Global CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE**

Define Default Filter Trigger Setting for Disabling All Performance Entries

Define Default Filter Trigger Setting for Disabling All Performance Entries

**Global CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS**

Maximum number of block sizes in pool structures

Maximum number of block sizes in pool structures

**Global CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING**

Default virtual path for volatile storage

Default virtual path for volatile storage

**Global CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS**

ES Ram Disk Number of Sectors

ES Ram Disk Number of Sectors

**Global CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED**

Percentage of Ram Disk Reserved for Decompressing Apps

Percentage of Ram Disk Reserved for Decompressing Apps

**Global CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE**

ES Ram Disk Sector Size

ES Ram Disk Sector Size

**Global CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY**

Define ES Task Priority

Define ES Task Priority

**Global CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE**

Define ES Task Stack Size

Define ES Task Stack Size

**Global CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC**

Startup script timeout

Startup script timeout

**Global CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC**

Poll timer for startup sync delay

Poll timer for startup sync delay

**Global CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE**

Define Size of the cFE System Log.

Define Size of the cFE System Log.

**Global CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE**

Define User Reserved Memory Size

Define User Reserved Memory Size

**Global CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE**

ES Volatile Startup Filename

ES Volatile Startup Filename

**Global CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY**

Define EVS Task Priority

Define EVS Task Priority

**Global CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE**

Define EVS Task Stack Size

Define EVS Task Stack Size

**Global CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01**

Define SB Memory Pool Block Sizes

Define SB Memory Pool Block Sizes

**Global CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY**

Define SB Task Priority

Define SB Task Priority

**Global CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE**

Define SB Task Stack Size

Define SB Task Stack Size

**Global CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY**

Define TBL Task Priority

Define TBL Task Priority

**Global CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE**

Define TBL Task Stack Size

Define TBL Task Stack Size

## 2.10 cFE Event Services Overview

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

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

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

- [Event Message Format](#)
- [Local Event Log](#)
- [Event Message Control](#)
- [Event Message Filtering](#)
- [EVS Registry](#)
- [EVS Counters](#)
- [Resetting EVS Counters](#)
- [Effects of a Processor Reset on EVS](#)
- [EVS squelching of misbehaving apps](#)
- [Frequently Asked Questions about Event Services](#)

### 2.10.1 Event Message Format

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

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

The *Timestamp* corresponds to when the event was generated, in spacecraft time. The *Event Type* is one of the following: DEBUG, INFO, ERROR or CRITICAL. The *Spacecraft ID* and *Processor ID* identify the spacecraft and processor from which the event was generated. Note that the *Spacecraft ID* is defined in the cfe\_mission\_cfg.h file; The *Processor ID* is defined in the appropriate cfe\_platform\_cfg.h file. The *Application Name* refers to the Application that issued the event message as specified on application startup (either startup script or app start command). The *Event ID* is an Application unique number that identifies the event. The *Message* is an ASCII text string describing the event. Event messages may have parameters associated with the event message. EVS formats the parameters such that they are part of the ASCII text string that make up the event message.

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

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

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

### 2.10.2 Local Event Log

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

EVS provides a command in order to [clear the Local Event Log](#).

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

### 2.10.3 Event Message Control

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

**2.10.3.1 Event Message Control - By Type** The highest level of event message control that EVS provides is the ability to enable and disable event message types. As mentioned above, there are four event types. They are:

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

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

**2.10.3.2 Event Message Control - By Application** Commands are available to [enable](#) and [disable](#) the generation of event messages for a particular application. The result is that ALL event messages for the specified Application are affected (i.e. enabled or disabled).

**2.10.3.3 Event Message Control - By Event Type for an Application** EVS also provides the capability to [enable](#) / [disable](#) an event type for a particular application. Note that EVS provides the capability to affect multiple event types within one command using a bit mask.

**2.10.3.4 Event Message Control - Individual Events** There are two ways to control the generation of individual events depending on whether the application's event message has been registered with EVS or not.

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

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

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

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

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

## 2.10.4 Event Message Filtering

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

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

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

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

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

	<b>packet X</b>	<b>packet X+1</b>	<b>packet X+2</b>	<b>packet X+3</b>	<b>packet X+4</b>	...
<b>Event ID counter</b>	x'0000'	x'0001'	x'0002'	x'0003'	x'0004'	
<b>Event Filter mask</b>	x'0001'	x'0001'	x'0001'	x'0001'	x'0001'	
<b>Bitwise AND results</b>	x'0000'	x'0001'	x'0000'	x'0001'	x'0000'	
<b>Send event?</b>	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.

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

See [cfe\\_evs.h](#) for predefined macro values which can be used for masks.

### 2.10.5 EVS Registry

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

The registry contains the following information for each Registered Application:

- Active Flag - If equal to FALSE (0), all events from this Application are Filtered
- Event Count - Total number of events issued by this Application. Note that this value stop incrementing at 65535.

The following information for each Filtered Event (up to [CFE\\_PLATFORM\\_EVS\\_MAX\\_EVENT\\_FILTERS](#)) :

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

### 2.10.6 EVS Counters

There are 2 types of counters in EVS housekeeping telemetry:

- Total events sent counter

- Number of events sent for each Application

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

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

### 2.10.7 Resetting EVS Counters

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

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

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

### 2.10.8 Effects of a Processor Reset on EVS

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

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

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

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

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

### 2.10.9 EVS squelching of misbehaving apps

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

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

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

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

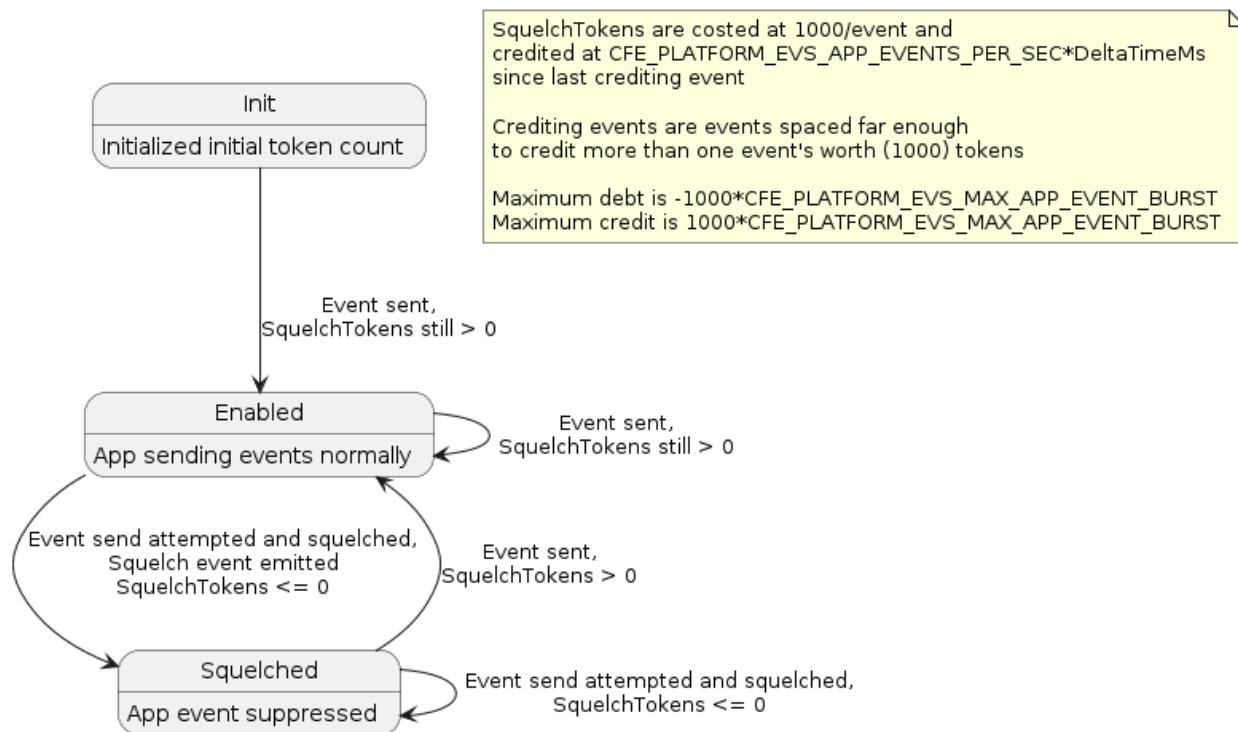


Figure 1 Figure EVS-1: EVS Squelching State Diagram

### 2.10.10 Frequently Asked Questions about Event Services

#### (Q) My telemetry stream is being flooded with the same event message. How do I make it stop?

The most direct way to stop an event message from flooding your downlink stream is to send a command to EVS to filter the offending event (see [Event Message Control](#) or `$sc_$cpu_EVS_SetBinFltrMask`). In order to stop the event

*message from being sent, a bit mask of '0xFFFF' should be used. If the event is not currently registered for filtering, the event message must be added using the command [\\$sc\\_\\$cpu\\_EVS\\_AddEvtFltr](#).*

**(Q) I filtered an event message and would now like to see it again. What do I do in order to see those events again?**

*If the event message that you are interested in is registered with EVS for filtering, then you have 2 options:*

1. *You can use the [\\$sc\\_\\$cpu\\_EVS\\_SetBinFltrMask](#) command using a bit mask of '0x0000' which will result in getting all of the events for that Event Id*
2. *You can remove the registration of that event with EVS (see [\\$sc\\_\\$cpu\\_EVS\\_DelEvtFltr](#)).*

*Note that option (1) is the preferred method.*

**(Q) What is the purpose of DEBUG event messages?**

*Event message of type "DEBUG" are primarily used during flight software development in order to provide information that is most likely not needed on orbit. Some commands send debug event messages as verification that a command request was received. When writing the EVS local event log to a file, for example, an event message of type DEBUG is issued. On orbit, this event message is probably not needed. Instead, the command counter is used for command verification.*

**(Q) How do I find out which events are registered for filtering?**

*EVS provides a command ([\\$sc\\_\\$cpu\\_EVS\\_WriteAppData2File](#)) which generates a file containing all of the applications that have registered with EVS and all of the filters that are registered for each application. Note that EVS merely generates the file. The file must be transferred to the ground in order to view it.*

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

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

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

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

## 2.11 cFE Event Services Commands

Upon receipt of any command, the Event Services application will confirm that the message length embedded within the header (from [CFE\\_MSG\\_GetSize\(\)](#)) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, EVS will generate the [CFE\\_EVS\\_LEN\\_ERR\\_EID](#) event, increment the command error counter ([\\$sc\\_\\$cpu\\_EVS\\_CMDEC](#)), and the command will *not* be accepted for processing.

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

### Global [CFE\\_EVS\\_ADD\\_EVENT\\_FILTER\\_CC](#)

Add Application Event Filter

**Global CFE\_EVS\_CLEAR\_LOG\_CC**

Clear Event Log

**Global CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC**

Delete Application Event Filter

**Global CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC**

Disable Application Event Type

**Global CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC**

Disable Event Services for an Application

**Global CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC**

Disable Event Type

**Global CFE\_EVS\_DISABLE\_PORTS\_CC**

Disable Event Services Output Ports

**Global CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC**

Enable Application Event Type

**Global CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC**

Enable Event Services for an Application

**Global CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC**

Enable Event Type

**Global CFE\_EVS\_ENABLE\_PORTS\_CC**

Enable Event Services Output Ports

**Global CFE\_EVS\_NOOP\_CC**

Event Services No-Op

**Global CFE\_EVS\_RESET\_ALL\_FILTERS\_CC**

Reset All Event Filters for an Application

**Global CFE\_EVS\_RESET\_APP\_COUNTER\_CC**

Reset Application Event Counters

**Global CFE\_EVS\_RESET\_COUNTERS\_CC**

Event Services Reset Counters

**Global CFE\_EVS\_RESET\_FILTER\_CC**

Reset an Event Filter for an Application

**Global CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC**

Set Event Format Mode

**Global CFE\_EVS\_SET\_FILTER\_CC**

Set Application Event Filter

**Global CFE\_EVS\_SET\_LOG\_MODE\_CC**

Set Logging Mode

**Global CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC**

Write Event Services Application Information to File

**Global CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC**

Write Event Log to File

## 2.12 cFE Event Services Telemetry

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

### Global `CFE_EVS_HousekeepingTlm_Payload_t`

Event Services Housekeeping Telemetry Packet

### Global `CFE_EVS_HousekeepingTlm_Payload_t`

Event Services Housekeeping Telemetry Packet

### Global `CFE_EVS_LongEventTlm_Payload_t`

Event Message Telemetry Packet (Long format)

### Global `CFE_EVS_LongEventTlm_Payload_t`

Event Message Telemetry Packet (Long format)

### Global `CFE_EVS_ShortEventTlm_Payload_t`

Event Message Telemetry Packet (Short format)

### Global `CFE_EVS_ShortEventTlm_Payload_t`

Event Message Telemetry Packet (Short format)

## 2.13 cFE Event Services Configuration Parameters

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

### Global `CFE_MISSION_EVS_MAX_MESSAGE_LENGTH`

Maximum Event Message Length

Maximum Event Message Length

### Global `CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC`

Sustained number of event messages per second per app before squelching

Sustained number of event messages per second per app before squelching

### Global `CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE`

Default EVS Application Data Filename

Default EVS Application Data Filename

### Global `CFE_PLATFORM_EVS_DEFAULT_LOG_FILE`

Default Event Log Filename

Default Event Log Filename

### Global `CFE_PLATFORM_EVS_DEFAULT_LOG_MODE`

Default EVS Local Event Log Mode

Default EVS Local Event Log Mode

### Global `CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE`

Default EVS Message Format Mode

Default EVS Message Format Mode

**Global CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG**

Default EVS Event Type Filter Mask

Default EVS Event Type Filter Mask

**Global CFE\_PLATFORM\_EVS\_LOG\_MAX**

Maximum Number of Events in EVS Local Event Log

Maximum Number of Events in EVS Local Event Log

**Global CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST**

Maximum number of event before squelching

Maximum number of event before squelching

**Global CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS**

Define Maximum Number of Event Filters per Application

Define Maximum Number of Event Filters per Application

**Global CFE\_PLATFORM\_EVS\_PORT\_DEFAULT**

Default EVS Output Port State

Default EVS Output Port State

## 2.14 cFE Software Bus Overview

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

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

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

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

### 2.14.1 Software Bus Terminology

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

- [Messages](#)
- [Pipes](#)
- [Subscriptions](#)
- [Memory](#)

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

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

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

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

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

Following the header is the user defined message data.

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

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

**2.14.1.3 Subscriptions** A subscription is a run-time request for a particular message to be sent to a particular pipe. If the caller of the subscribe API is not the owner of the pipe, the request is rejected and an error event is sent. The application that creates the pipe is considered the owner of the pipe. The pipe specified in the subscription is sometimes referred to as the destination of the message. There are a maximum number of destinations for a particular message. This value is specified by the platform configuration parameter [CFE\\_PLATFORM\\_SB\\_MAX\\_DEST\\_PER\\_PKT](#).

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

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

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

**2.14.1.4 Memory** The software bus statically allocates a block of memory for message buffers and subscription blocks. The size of this memory block is defined by the platform configuration parameter [CFE\\_PLATFORM\\_SB\\_BUF\\_MEMORY\\_BYTES](#). The memory is managed by the cFE ES memory pool and is used only by the software bus. The ES memory pool allows an application to define the block sizes for the pool at compile time. These sizes are defined by the platform configuration parameters prefixed with CFE\_SB\_MEM\_BLOCK\_SIZE (for example, [CFE\\_PLATFORM\\_SB\\_MEM\\_BLOCK\\_SIZE\\_01](#)). It is recommended that a project tailor these values for the mission, based on the software bus packet sizes.

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

Also when subscriptions are received through the subscribe API's, the software bus allocates a subscription block (CFE\_SB\_DestinationD\_t) from the pool. The subscription blocks are returned to the pool if and when the subscription is nullified through a [CFE\\_SB\\_Unsubscribe](#) call.

The software bus provides a set of figures regarding memory capacity, current memory utilization and high water marks relevant to the SB memory pool. This information may be requested by sending the command to dump the SB statistics packet. In addition, the current memory utilization value and the 'unmarked memory' value ([CFE\\_PLATFORM\\_SB\\_BUF\\_MEMORY\\_BYTES](#) minus peak memory in use) are sent in software bus housekeeping telemetry. The unmarked memory value should be monitored regularly to ensure that the value (in bytes) does not continue to decline or approach zero. If this value were to approach zero, there is a possibility that memory requests would fail which may inhibit the sending of a message. The current memory utilization value should also be monitored to ensure the system contains no memory leaks. The value (in bytes) should remain stable under nominal conditions. Refer to the ES users guide for more information regarding the ES Memory Pool.

## 2.14.2 Autonomous Actions

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

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

### 2.14.3 Operation of the SB Software

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

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

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

After any reset the following statements are true:

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

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

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

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

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

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

When disabled, the original message will not be altered. This method of message delivery is recommended for situations where the sender did not generate the packet, such as a network interface application passing a packet from a remote system to the local software bus.

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

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

If this error occurs during nominal conditions, it could be an indication that the 'message limit' is not set correctly. The message limit is given at the time of the subscription and given as a parameter in the subscribe API. With the [CFE\\_SB\\_Subscribe](#) API, the SB uses a default message limit value specified by [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_MSG\\_LIMIT](#). This constant is currently set to a value of four. If the default value is insufficient, the message limit value can be specified in the [CFE\\_SB\\_SubscribeEx](#) API.

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

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

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

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

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

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

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

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

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

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

**2.14.3.10 Quality of Service** The software bus has a parameter in the [CFE\\_SB\\_SubscribeEx](#) API named Quality, which means Quality of Service (QOS) for off-board routing and is of the type [CFE\\_SB\\_Qos\\_t](#). This structure has two members named priority and reliability. The Quality parameter is currently unused by the software bus. It is a placeholder to be used with the future software bus capability of inter-processor communication. Although currently the software bus does not implement quality of service.

A default quality of services is provided via the [CFE\\_SB\\_DEFAULT\\_QOS](#) macro.

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

The simplest way to prevent this behavior is to ensure that there is margin when sizing the SB memory pool. To check the margin, monitor the "Peak Memory in Use" vs. the configuration parameter [CFE\\_PLATFORM\\_SB\\_BUF\\_MEMORY\\_BYTES](#) which indicates the amount allocated.

#### 2.14.4 Frequently Asked Questions about Software Bus

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

*The memory pool handle is used to analyze the SB memory pool statistics. The cFE ES command ([CFE\\_ES\\_SEND\\_MEM\\_POOL\\_STATS\\_CC](#)) to dump the memory pool statistics takes the pool handle as a parameter. These statistics tell how the SB memory pool is configured and gives details on margin. An improperly configured SB memory pool may inhibit communication. This may occur if there is not enough margin to create a block of the size needed for a transfer. Refer to the ES memory pool users guide for more details. [Memory Pool](#)*

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

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

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

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

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

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

**(Q) The first parameter in the [CFE\\_SB\\_ReceiveBuffer](#) API is a pointer to a pointer which can get confusing. How can I be sure that the correct address is given for this parameter.**

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

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

*It is possible the events are being filtered by cFE Event Services. The filtering for this event may be specified in the platform configuration file or it may have been commanded after the system initializes.*

*There is a corresponding counter for each of these conditions. First verify that the condition is happening by viewing the counter in SB HK telemetry. If the condition is happening, you can view the SB filter information through the EVS App Data Main page by clicking the 'go to' button for SB. The event Id for these events can be learned through a previous event or from the `cfe_sb_eventids.h` file.*

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

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

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

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

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

*The QOS parameter is currently unused by the software bus. It is a placeholder to be used with the future software bus capability of inter-processor communication. Setting the QOS as `CFE_SB_DEFAULT_QOS` will ensure seamless integration when the software bus is expanded to support inter-processor communication.*

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

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

## 2.15 cFE Software Bus Commands

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

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

### Global `CFE_SB_DISABLE_ROUTE_CC`

Disable Software Bus Route

**Global CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC**  
Disable Subscription Reporting Command

**Global CFE\_SB\_ENABLE\_ROUTE\_CC**  
Enable Software Bus Route

**Global CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC**  
Enable Subscription Reporting Command

**Global CFE\_SB\_NOOP\_CC**  
Software Bus No-Op

**Global CFE\_SB\_RESET\_COUNTERS\_CC**  
Software Bus Reset Counters

**Global CFE\_SB\_SEND\_PREV\_SUBS\_CC**  
Send Previous Subscriptions Command

**Global CFE\_SB\_SEND\_SB\_STATS\_CC**  
Send Software Bus Statistics

**Global CFE\_SB\_WRITE\_MAP\_INFO\_CC**  
Write Map Info to a File

**Global CFE\_SB\_WRITE\_PIPE\_INFO\_CC**  
Write Pipe Info to a File

**Global CFE\_SB\_WRITE\_ROUTING\_INFO\_CC**  
Write Software Bus Routing Info to a File

## 2.16 cFE Software Bus Telemetry

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

**Global CFE\_SB\_AllSubscriptionsTlm\_Payload\_t**  
SB Previous Subscriptions Packet

**Global CFE\_SB\_AllSubscriptionsTlm\_Payload\_t**  
SB Previous Subscriptions Packet

**Global CFE\_SB\_HousekeepingTlm\_Payload\_t**  
Software Bus task housekeeping Packet

**Global CFE\_SB\_HousekeepingTlm\_Payload\_t**  
Software Bus task housekeeping Packet

**Global CFE\_SB\_SingleSubscriptionTlm\_Payload\_t**  
SB Subscription Report Packet

**Global CFE\_SB\_SingleSubscriptionTlm\_Payload\_t**  
SB Subscription Report Packet

**Global CFE\_SB\_StatsTlm\_Payload\_t**  
SB Statistics Telemetry Packet

**Global CFE\_SB\_StatsTlm\_Payload\_t**  
SB Statistics Telemetry Packet

## 2.17 cFE Software Bus Configuration Parameters

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

### Global CFE\_MISSION\_SB\_MAX\_PIPES

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

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

### Global CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE

Maximum SB Message Size

Maximum SB Message Size

### Global CFE\_PLATFORM\_ENDIAN

Platform Endian Indicator

### Global CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES

Size of the SB buffer memory pool

Size of the SB buffer memory pool

### Global CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME

Default Message Map Filename

Default Message Map Filename

### Global CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT

Default Subscription Message Limit

Default Subscription Message Limit

### Global CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME

Default Pipe Information Filename

Default Pipe Information Filename

### Global CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME

Default Routing Information Filename

Default Routing Information Filename

### Global CFE\_PLATFORM\_SB\_FILTERED\_EVENT1

SB Event Filtering

SB Event Filtering

### Global CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID

Highest Valid Message Id

Highest Valid Message Id

### Global CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT

Maximum Number of unique local destinations a single MsgId can have

Maximum Number of unique local destinations a single MsgId can have

### Global CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS

Maximum Number of Unique Message IDs SB Routing Table can hold

Maximum Number of Unique Message IDs SB Routing Table can hold

### Global CFE\_PLATFORM\_SB\_MAX\_PIPES

Maximum Number of Unique Pipes SB Routing Table can hold

Maximum Number of Unique Pipes SB Routing Table can hold

## 2.18 cFE Table Services Overview

Applications often organize sets of their parameters into logical units called tables. These are typically constant parameters that can change the behavior of a flight software algorithm and are only intended to be modified by operations personnel. Examples of this would be attitude control gains, sensor scalefactors, telemetry filter settings, etc.

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

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

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

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

### 2.18.1 Managing Tables

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

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

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

To understand the methods of maintaining a table, it is important that the terminology be clear. A table has two images: "Active" and "Inactive". The Active table is the one that a cFE Application is currently accessing when it executes. The

Inactive table is a copy of the Active table that an operator (or on-board process such as a stored command processor) can manipulate and change to have a newly desired set of data.

To create an Inactive table image on board, the operator would be required to perform a "Load" to the table. Loads are table images stored in on-board files. The Load can contain either a complete table image or just a part of a table image. If the Load contains just a portion, the Inactive image is first initialized with the contents of the Active image and then the portion identified in the Load file is written on top of the Active image. After the initial Load, an operator can continue to manipulate the Inactive table image with additional partial table load images. This allows the operator to reconfigure the contents of multiple portions of the table before deciding to "Validate" and/or "Activate" it.

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

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

### 2.18.2 cFE Table Types and Table Options

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

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

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

- Table Types
  - [Single Buffered Tables](#)
  - [Double Buffered Tables](#)
- Table Options
  - [Tables with Validation Functions](#)
  - [Critical Tables](#)
  - [User Defined Address Tables](#)
  - [Dump Only Tables](#)

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

The number of single buffered tables that can have inactive table images being manipulated at one time is specified by a TBL Services configuration parameter ([CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#)) found in the `cfe_platform_cfg.h` file associated with the processor in question. This parameter identifies the number of shared table buffers that are available.

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

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

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

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

Applications which have unusually large tables may decide to conserve memory resources by making them double buffered. This is because the shared buffers used by single buffered tables must be sized to match the largest table. If there is one table that is unusually large, there is little reason to allocate up to [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) number of buffers that size. A double buffered table will only allocate ONE extra buffer of that size.

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

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

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

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

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

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

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

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

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

### 2.18.3 Table Registry

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

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

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

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

#### 2.18.4 Table Services Telemetry

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

#### 2.18.5 Effects of Processor Reset on Tables

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

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

#### 2.18.6 Frequently Asked Questions about Table Services

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

*Table images that are smaller than the declared size of a table fall into one of two categories.*

*If the starting offset of the table image (as specified in the Table Image secondary file header) is not equal to zero, then the table image is considered to be a "partial" table load. Partial loads are valid as long as a table has been previously loaded with a non-"partial" table image.*

*If the starting offset of the table image is zero and the size is less than the declared size of the table, the image is considered "short" but valid. This feature allows application developers to use variable length tables.*

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

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

##### What happened?

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

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

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

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

**(Q) What causes cFE Table Services to generate the following sys log message:**

```
CFE_TBL:GetAddressInternal-App(%d) attempt to access unowned Tbl Handle=%d
```

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

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

The Abort command should be used whenever a table image has been loaded but the application has not yet activated it and the operator no longer wants the table to be loaded.

The purpose of the Abort command is to free a previously allocated table buffer. It should be noted, however, that multiple table loads to the SAME table without an intervening activation or abort, will simply OVERWRITE the previous table load using the SAME buffer.

Therefore, the most likely scenarios that would lead to a needed abort are as follows:

1. Operator loads a table and realizes immediately that the load is not wanted.
2. Operator loads a table and performs a validation on it. Regardless of whether the table passes or fails the validation, if the operator no longer wants to activate the table, the abort command should be issued.  
*It should be noted that a table image that fails activation is retained in the inactive buffer for diagnosis, if necessary. It is NOT released until it is aborted or overwritten and successfully validated and activated.*
3. A table image was loaded; the image was successfully validated; the command for activation was sent; but the application fails to perform the activation.

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

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

## 2.19 cFE Table Services Commands

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

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

### Global `CFE_TBL_ABORT_LOAD_CC`

Abort Table Load

### Global `CFE_TBL_ACTIVATE_CC`

Activate Table

### Global `CFE_TBL_DELETE_CDS_CC`

Delete Critical Table from Critical Data Store

### Global `CFE_TBL_DUMP_CC`

Dump Table

### Global `CFE_TBL_DUMP_REGISTRY_CC`

Dump Table Registry

### Global `CFE_TBL_LOAD_CC`

Load Table

### Global `CFE_TBL_NOOP_CC`

Table No-Op

### Global `CFE_TBL_RESET_COUNTERS_CC`

Table Reset Counters

### Global `CFE_TBL_SEND_REGISTRY_CC`

Telemeter One Table Registry Entry

### Global `CFE_TBL_VALIDATE_CC`

Validate Table

## 2.20 cFE Table Services Telemetry

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

### Global `CFE_TBL_HousekeepingTlm_Payload_t`

Table Services Housekeeping Packet

### Global `CFE_TBL_HousekeepingTlm_Payload_t`

Table Services Housekeeping Packet

### Global `CFE_TBL_TblRegPacket_Payload_t`

Table Registry Info Packet

### Global `CFE_TBL_TblRegPacket_Payload_t`

Table Registry Info Packet

## 2.21 cFE Table Services Configuration Parameters

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

### Global CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN

Maximum Length of Full Table Name in messages  
Maximum Length of Full Table Name in messages

### Global CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH

Maximum Table Name Length  
Maximum Table Name Length

### Global CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES

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

### Global CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE

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

### Global CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES

Maximum Number of Critical Tables that can be Registered  
Maximum Number of Critical Tables that can be Registered

### Global CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE

Maximum Size Allowed for a Double Buffered Table  
Maximum Size Allowed for a Double Buffered Table

### Global CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES

Maximum Number of Table Handles  
Maximum Number of Table Handles

### Global CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES

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

### Global CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS

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

### Global CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS

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

### Global CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE

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

### Global CFE\_PLATFORM\_TBL\_VALID\_PRID\_1

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

**Global CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT**

Number of Processor ID's specified for validation

Number of Processor ID's specified for validation

**Global CFE\_PLATFORM\_TBL\_VALID\_SCID\_1**

Spacecraft ID values used for table load validation

Spacecraft ID values used for table load validation

**Global CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT**

Number of Spacecraft ID's specified for validation

Number of Spacecraft ID's specified for validation

## 2.22 cFE Time Services Overview

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

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

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

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

- [Time Components](#)
- [Time Structure](#)
- [Time Formats](#)
- [Time Configuration](#)
  - [Time Format Selection](#)
  - [Enabling Fake Tone Signal](#)
  - [Selecting Tone and Data Ordering](#)
  - [Specifying Tone and Data Window](#)
  - [Specifying Time Server/Client](#)

- Specifying Time Tone Byte Order
- Virtual MET
- Specifying Time Source
- Specifying Time Signal
- Time Services Paradigm(s)
- Flywheeling
- Time State
- Initialization
  - Power-On Reset
  - Processor Reset
- Initialization
  - Power-On Reset
  - Processor Reset
- Normal Operation
  - Client
  - Server
    - \* Setting Time
    - \* Adjusting Time
    - \* Setting MET
- Frequently Asked Questions about Time Services

### 2.22.1 Time Components

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

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

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

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

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

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

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

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

### 2.22.2 Time Structure

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

The system time structure is defined as follows:

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

### 2.22.3 Time Formats

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

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

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

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

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

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

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

### 2.22.4 Time Configuration

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

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

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

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

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

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

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

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

#### 2.22.4.1 Time Format Selection

Time format is defined in the mission configuration header file.

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

```
#define CFE_MISSION_TIME_CFG_DEFAULT_TAI  TRUE  
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC  FALSE
```

or

```
#define CFE_MISSION_TIME_CFG_DEFAULT_TAI  FALSE  
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC  TRUE
```

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

See also

[CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_TAI](#), [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_UTC](#)

#### 2.22.4.2 Enabling Fake Tone Signal

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

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

```
#define CFE_MISSION_TIME_CFG_FAKE_TONE    TRUE
```

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

See also

[CFE\\_MISSION\\_TIME\\_CFG\\_FAKE\\_TONE](#)

#### 2.22.4.3 Selecting Tone and Data Ordering

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

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

```
#define CFE_MISSION_TIME_AT_TONE_WAS  
#define CFE_MISSION_TIME_AT_TONE_WILL_BE
```

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

See also

[CFE\\_MISSION\\_TIME\\_AT\\_TONE\\_WAS](#), [CFE\\_MISSION\\_TIME\\_AT\\_TONE\\_WILL\\_BE](#)

#### 2.22.4.4 Specifying Tone and Data Window

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

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

```
#define CFE_MISSION_TIME_MIN_ELAPSED  0  
#define CFE_MISSION_TIME_MAX_ELAPSED 100000
```

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

See also

[CFE\\_MISSION\\_TIME\\_MIN\\_ELAPSED](#), [CFE\\_MISSION\\_TIME\\_MAX\\_ELAPSED](#)

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

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

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

or

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

#### See also

[CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_CLIENT](#)

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

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

```
#define CFE_PLATFORM_TIME_CFG_BIGENDIAN
```

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

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

TIME servers must be configured as using a virtual MET

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

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

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

```
#define CFE_PLATFORM_TIME_CFG_SOURCE   TRUE
```

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

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

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

```
#define CFE_PLATFORM_TIME_CFG_SRC_MET    TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS    FALSE
#define CFE_PLATFORM_TIME_CFG_SRC_TIME   FALSE
```

configuration definitions for the particular source.

If the cfe\_platform\_cfg.h file contains "#define CFE\_PLATFORM\_TIME\_CFG\_SOURCE TRUE" then time is configured to allow switching between internal and external time sources (see [CFE\\_TIME\\_SET\\_SOURCE\\_CC](#)). If this configuration parameter is set to FALSE then the command to set the source will be rejected.

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

```
#define CFE_PLATFORM_TIME_CFG_SRC_MET    TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS    FALSE
#define CFE_PLATFORM_TIME_CFG_SRC_TIME   FALSE
```

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

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

#### See also

[CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_MET](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_GPS](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_TIME](#)

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

```
#define CFE_PLATFORM_TIME_CFG_SIGNAL  TRUE
```

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

#### See also

[CFE\\_PLATFORM\\_TIME\\_CFG\\_SIGNAL](#)

### 2.22.5 Time Format Selection

Time format is defined in the mission configuration header file.

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

```
#define CFE_MISSION_TIME_CFG_DEFAULT_TAI TRUE  
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC FALSE
```

or

```
#define CFE_MISSION_TIME_CFG_DEFAULT_TAI FALSE  
#define CFE_MISSION_TIME_CFG_DEFAULT_UTC TRUE
```

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

See also

[CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_TAI](#), [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_UTC](#)

### 2.22.6 Enabling Fake Tone Signal

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

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

```
#define CFE_MISSION_TIME_CFG_FAKE_TONE TRUE
```

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

See also

[CFE\\_MISSION\\_TIME\\_CFG\\_FAKE\\_TONE](#)

### 2.22.7 Selecting Tone and Data Ordering

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

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

```
#define CFE_MISSION_TIME_AT_TONE_WAS  
#define CFE_MISSION_TIME_AT_TONE_WILL_BE
```

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

See also

[CFE\\_MISSION\\_TIME\\_AT\\_TONE\\_WAS](#), [CFE\\_MISSION\\_TIME\\_AT\\_TONE\\_WILL\\_BE](#)

### 2.22.8 Specifying Tone and Data Window

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

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

```
#define CFE_MISSION_TIME_MIN_ELAPSED 0  
#define CFE_MISSION_TIME_MAX_ELAPSED 100000
```

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

See also

[CFE\\_MISSION\\_TIME\\_MIN\\_ELAPSED](#), [CFE\\_MISSION\\_TIME\\_MAX\\_ELAPSED](#)

### 2.22.9 Specifying Time Server/Client

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

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

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

or

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

See also

[CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_CLIENT](#)

### 2.22.10 Specifying Time Tone Byte Order

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

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

```
#define CFE_PLATFORM_TIME_CFG_BIGENDIAN
```

### 2.22.11 Virtual MET

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

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

TIME servers must be configured as using a virtual MET

### 2.22.12 Specifying Time Source

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

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

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

```
#define CFE_PLATFORM_TIME_CFG_SOURCE TRUE
```

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

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

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

```
#define CFE_PLATFORM_TIME_CFG_SRC_MET TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS FALSE
#define CFE_PLATFORM_TIME_CFG_SRC_TIME FALSE
```

configuration definitions for the particular source.

If the cfe\_platform\_cfg.h file contains "#define CFE\_PLATFORM\_TIME\_CFG\_SOURCE TRUE" then time is configured to allow switching between internal and external time sources (see [CFE\\_TIME\\_SET\\_SOURCE\\_CC](#)). If this configuration parameter is set to FALSE then the command to set the source will be rejected.

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

```
#define CFE_PLATFORM_TIME_CFG_SRC_MET TRUE
#define CFE_PLATFORM_TIME_CFG_SRC_GPS FALSE
#define CFE_PLATFORM_TIME_CFG_SRC_TIME FALSE
```

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

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

#### See also

[CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_MET](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_GPS](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_TIME](#)

### 2.22.13 Specifying Time Signal

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

```
#define CFE_PLATFORM_TIME_CFG_SIGNAL TRUE
```

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

See also

[CFE\\_PLATFORM\\_TIME\\_CFG\\_SIGNAL](#)

### 2.22.14 Time Services Paradigm(s)

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

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

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

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

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

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

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

The mission configuration parameters, [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_TAI](#) and [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_UTC](#) specify the default time format. Applications are encouraged to use the [CFE\\_TIME\\_GetTime](#) API, which returns time in the format specified by this configuration parameter.

### 2.22.15 Flywheeling

Flywheeling occurs when TIME is not getting a valid tone signal or external "time at the tone" message. While this has minimal impact on internal operations, it can result in the drifting apart of times being stored by different spacecraft systems.

Flywheeling occurs when at least one of the following conditions is true:

- loss of tone signal
- loss of "time at the tone" data packet
- signal and packet not within valid window
- commanded into fly-wheel mode

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

### 2.22.16 Time State

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

### 2.22.17 Initialization

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

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

- [Power-On Reset](#)
- [Processor Reset](#)

**2.22.17.1 Power-On Reset** TIME initializes all counters in housekeeping telemetry, sets the Validity state to Invalid, and initializes the STCF, Leap Seconds, and 1 Hz Adjustment to zero.

**2.22.17.2 Processor Reset** In the event of a processor reset, the following time values are preserved:

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

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

#### **2.22.18 Power-On Reset**

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

#### **2.22.19 Processor Reset**

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

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

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

### 2.22.20 Initialization

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

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

- [Power-On Reset](#)
- [Processor Reset](#)

**2.22.20.1 Power-On Reset** TIME initializes all counters in housekeeping telemetry, sets the Validity state to Invalid, and initializes the STCF, Leap Seconds, and 1 Hz Adjustment to zero.

**2.22.20.2 Processor Reset** In the event of a processor reset, the following time values are preserved:

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

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

### 2.22.21 Power-On Reset

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

### 2.22.22 Processor Reset

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

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

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

### 2.22.23 Normal Operation

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

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

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

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

- [Setting Time](#)
- [Adjusting Time](#)
- [Setting MET](#)

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

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

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

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

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

#### See also

[CFE\\_TIME\\_SET\\_TIME\\_CC](#)

**2.22.23.2.2 Adjusting Time** The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set implicitly using the [CFE\\_TIME\\_SET\\_TIME\\_CC](#) or explicitly using [CFE\\_TIME\\_SET\\_STCF\\_CC](#). TIME provides the ability to command a one time adjustment ([CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#) and [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#)) to the current STCF. In addition there is a 1Hz adjustment ([CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#) and [CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#)) that can be made to the STCF to compensate for oscillator drift. Mission specific ground correlation should be used to assist in determining the proper values to use. The Leap Seconds should be set to the current TAI-UTC. Note that the International Earth Rotation and Reference Systems Service Bulletin C, which defines the current difference, reports it as UTC-TAI, and thus that value must be negated. **The Leap Seconds value will always be a positive number.** The Validity state does not have to be set to invalid to change the STCF or Leap Seconds, and should be set to valid at any time that the TIME Server time reference should be synchronized to by the other systems.

#### See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SET\\_STCF\\_CC](#), [CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SET\\_LEAP\\_SECONDS\\_CC](#)

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

#### See also

[CFE\\_TIME\\_SET\\_MET\\_CC](#)

### 2.22.24 Client

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

### 2.22.25 Server

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

- [Setting Time](#)
- [Adjusting Time](#)
- [Setting MET](#)

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

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

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

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

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

See also

[CFE\\_TIME\\_SET\\_TIME\\_CC](#)

**2.22.25.0.2 Adjusting Time** The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set implicitly using the [CFE\\_TIME\\_SET\\_TIME\\_CC](#) or explicitly using [CFE\\_TIME\\_SET\\_STCF\\_CC](#). TIME provides the ability to command a one time adjustment ([CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#) and [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#)) to the current STCF. In addition there is a 1Hz adjustment ([CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#) and [CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#)) that can be made to the STCF to compensate for oscillator drift. Mission specific ground correlation should be used to assist in determining the proper values to use. The Leap Seconds should be set to the current TAI-UTC. Note that the International Earth Rotation and Reference Systems Service Bulletin C, which defines the current difference, reports it as UTC-TAI, and thus that value must be negated. **The Leap Seconds value will always be a positive number.** The Validity state does not have to be set to invalid to change the STCF or Leap Seconds, and should be set to valid at any time that the TIME Server time reference should be synchronized to by the other systems.

See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SET\\_STCF\\_CC](#), [CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SET\\_LEAP\\_SECONDS\\_CC](#)

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

See also

[CFE\\_TIME\\_SET\\_MET\\_CC](#)

## 2.22.26 Setting Time

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

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

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

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

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

See also

[CFE\\_TIME\\_SET\\_TIME\\_CC](#)

### 2.22.27 Adjusting Time

The TIME Server includes commands to set the STCF, Leap Seconds, and Validity state. The STCF should be set implicitly using the [CFE\\_TIME\\_SET\\_TIME\\_CC](#) or explicitly using [CFE\\_TIME\\_SET\\_STCF\\_CC](#). TIME provides the ability to command a one time adjustment ([CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#) and [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#)) to the current STCF. In addition there is a 1Hz adjustment ([CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#) and [CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#)) that can be made to the STCF to compensate for oscillator drift. Mission specific ground correlation should be used to assist in determining the proper values to use. The Leap Seconds should be set to the current TAI-UTC. Note that the International Earth Rotation and Reference Systems Service Bulletin C, which defines the current difference, reports it as UTC-TAI, and thus that value must be negated. **The Leap Seconds value will always be a positive number.** The Validity state does not have to be set to invalid to change the STCF or Leap Seconds, and should be set to valid at any time that the TIME Server time reference should be synchronized to by the other systems.

See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SET\\_STCF\\_CC](#), [CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#),  
[CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SET\\_LEAP\\_SECONDS\\_CC](#)

### 2.22.28 Setting MET

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

See also

[CFE\\_TIME\\_SET\\_MET\\_CC](#)

### 2.22.29 Frequently Asked Questions about Time Services

None submitted

## 2.23 cFE Time Services Commands

Upon receipt of any command, the Time Services application will confirm that the message length embedded within the header (from [CFE\\_MSG\\_GetSize\(\)](#)) matches the expected length of that message, based on the size of the C structure defining that command. If there is any discrepancy between the expected and actual message size, TIME will generate the [CFE\\_TIME\\_LEN\\_ERR\\_EID](#) event, increment the command error counter (\$sc\_\$cpu\_TIME\_CMDEC), and the command will *not* be accepted for processing.

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

#### Global [CFE\\_TIME\\_ADD\\_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

Default Time and Tone Order

### Global CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI

Default Time Format

Default Time Format

### Global CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE

Default Time Format

Default Time Format

### Global CFE\_MISSION\_TIME\_DEF\_MET\_SECS

Default Time Values

Default Time Values

### Global CFE\_MISSION\_TIME\_EPOCH\_YEAR

Default EPOCH Values

Default EPOCH Values

### Global CFE\_MISSION\_TIME\_FS\_FACTOR

Time File System Factor

Time File System Factor

### Global CFE\_MISSION\_TIME\_MIN\_ELAPSED

Min and Max Time Elapsed

Min and Max Time Elapsed

### Global CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY

Define Periodic Time to Update Local Clock Tone Latch

Define Periodic Time to Update Local Clock Tone Latch

### Global CFE\_PLATFORM\_TIME\_CFG\_SERVER

Time Server or Time Client Selection

Time Server or Time Client Selection

### Global CFE\_PLATFORM\_TIME\_CFG\_SIGNAL

Include or Exclude the Primary/Redundant Tone Selection Cmd

Include or Exclude the Primary/Redundant Tone Selection Cmd

### Global CFE\_PLATFORM\_TIME\_CFG\_SOURCE

Include or Exclude the Internal/External Time Source Selection Cmd

Include or Exclude the Internal/External Time Source Selection Cmd

### Global CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET

Choose the External Time Source for Server only

Choose the External Time Source for Server only

**Global CFE\_PLATFORM\_TIME\_CFG\_START\_FLY**

Define Time to Start Flywheel Since Last Tone

Define Time to Start Flywheel Since Last Tone

**Global CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT**

Define Timing Limits From One Tone To The Next

Define Timing Limits From One Tone To The Next

**Global CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL**

Time Tone In Big-Endian Order

Local MET or Virtual MET Selection for Time Servers

Time Tone In Big-Endian Order

Local MET or Virtual MET Selection for Time Servers

**Global CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS**

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

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

**Global CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS**

Define the Local Clock Rollover Value in seconds and subseconds

Define the Local Clock Rollover Value in seconds and subseconds

**Global CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY**

Define TIME Task Priorities

Define TIME Task Priorities

**Global CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE**

Define TIME Task Stack Sizes

Define TIME Task Stack Sizes

## 2.26 cFE Event Message Cross Reference

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

---

## 2.27 cFE Command Mnemonic Cross Reference

The following cross reference maps the cFE command codes to Command Mnemonics. To learn about the details of a particular command, click on its associated command code.

---

**Global CFE\_ES\_CLEAR\_ER\_LOG\_CC**

\$sc\_\$cpu\_ES\_ClearERLog

**Global CFE\_ES\_CLEAR\_SYSLOG\_CC**

\$sc\_\$cpu\_ES\_ClearSysLog

**Global CFE\_ES\_DELETE\_CDS\_CC**

\$sc\_\$cpu\_ES\_DeleteCDS

**Global CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC**

\$sc\_\$cpu\_ES\_WriteCDS2File

**Global CFE\_ES\_NOOP\_CC**  
\$sc\_\$cpu\_ES\_NOOP

**Global CFE\_ES\_OVER\_WRITE\_SYSLOG\_CC**  
\$sc\_\$cpu\_ES\_OverwriteSysLogMode

**Global CFE\_ES\_QUERY\_ALL\_CC**  
\$sc\_\$cpu\_ES\_WriteApplInfo2File

**Global CFE\_ES\_QUERY\_ALL\_TASKS\_CC**  
\$sc\_\$cpu\_ES\_WriteTaskInfo2File

**Global CFE\_ES\_QUERY\_ONE\_CC**  
\$sc\_\$cpu\_ES\_QueryApp

**Global CFE\_ES\_RELOAD\_APP\_CC**  
\$sc\_\$cpu\_ES\_ReloadApp

**Global CFE\_ES\_RESET\_COUNTERS\_CC**  
\$sc\_\$cpu\_ES\_ResetCtrs

**Global CFE\_ES\_RESET\_PR\_COUNT\_CC**  
\$sc\_\$cpu\_ES\_ResetPRCnt

**Global CFE\_ES\_RESTART\_APP\_CC**  
\$sc\_\$cpu\_ES\_ResetApp

**Global CFE\_ES\_RESTART\_CC**  
\$sc\_\$cpu\_ES\_ProcessorReset, \$sc\_\$cpu\_ES\_PowerOnReset

**Global CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC**  
\$sc\_\$cpu\_ES\_PoolStats

**Global CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC**  
\$sc\_\$cpu\_ES\_SetMaxPRCnt

**Global CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC**  
\$sc\_\$cpu\_ES\_LAFilterMask

**Global CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC**  
\$sc\_\$cpu\_ES\_LATriggerMask

**Global CFE\_ES\_START\_APP\_CC**  
\$sc\_\$cpu\_ES\_StartApp

**Global CFE\_ES\_START\_PERF\_DATA\_CC**  
\$sc\_\$cpu\_ES\_StartLAData

**Global CFE\_ES\_STOP\_APP\_CC**  
\$sc\_\$cpu\_ES\_StopApp

**Global CFE\_ES\_STOP\_PERF\_DATA\_CC**  
\$sc\_\$cpu\_ES\_StopLAData

**Global CFE\_ES\_WRITE\_ER\_LOG\_CC**  
\$sc\_\$cpu\_ES\_WriteERLog2File

**Global CFE\_ES\_WRITE\_SYSLOG\_CC**  
\$sc\_\$cpu\_ES\_WriteSysLog2File

**Global CFE\_EVS\_ADD\_EVENT\_FILTER\_CC**  
\$sc\_\$cpu\_EVS\_AddEvtFltr

**Global CFE\_EVS\_CLEAR\_LOG\_CC**

\$sc\_\$cpu\_EVS\_ClrLog

**Global CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC**

\$sc\_\$cpu\_EVS\_DelEvtFltr

**Global CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC**

\$sc\_\$cpu\_EVS\_DisAppEvtType, \$sc\_\$cpu\_EVS\_DisAppEvtTypeMask

**Global CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC**

\$sc\_\$cpu\_EVS\_DisAppEvGen

**Global CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC**

\$sc\_\$cpu\_EVS\_DisEventType, \$sc\_\$cpu\_EVS\_DisEventTypeMask

**Global CFE\_EVS\_DISABLE\_PORTS\_CC**

\$sc\_\$cpu\_EVS\_DisPort, \$sc\_\$cpu\_EVS\_DisPortMask

**Global CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC**

\$sc\_\$cpu\_EVS\_EnaAppEvtType, \$sc\_\$cpu\_EVS\_EnaAppEvtTypeMask

**Global CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC**

\$sc\_\$cpu\_EVS\_EnaAppEvGen

**Global CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC**

\$sc\_\$cpu\_EVS\_EnaEventType, \$sc\_\$cpu\_EVS\_EnaEventTypeMask

**Global CFE\_EVS\_ENABLE\_PORTS\_CC**

\$sc\_\$cpu\_EVS\_EnaPort, \$sc\_\$cpu\_EVS\_EnaPortMask

**Global CFE\_EVS\_NOOP\_CC**

\$sc\_\$cpu\_EVS\_NOOP

**Global CFE\_EVS\_RESET\_ALL\_FILTERS\_CC**

\$sc\_\$cpu\_EVS\_RstAllFltrs

**Global CFE\_EVS\_RESET\_APP\_COUNTER\_CC**

\$sc\_\$cpu\_EVS\_RstAppCtrs

**Global CFE\_EVS\_RESET\_COUNTERS\_CC**

\$sc\_\$cpu\_EVS\_ResetCtrs

**Global CFE\_EVS\_RESET\_FILTER\_CC**

\$sc\_\$cpu\_EVS\_RstBinFltrCtr

**Global CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC**

\$sc\_\$cpu\_EVS\_SetEvtFmt

**Global CFE\_EVS\_SET\_FILTER\_CC**

\$sc\_\$cpu\_EVS\_SetBinFltrMask

**Global CFE\_EVS\_SET\_LOG\_MODE\_CC**

\$sc\_\$cpu\_EVS\_SetLogMode

**Global CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC**

\$sc\_\$cpu\_EVS\_WriteAppData2File

**Global CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC**

\$sc\_\$cpu\_EVS\_WriteLog2File

**Global CFE\_SB\_DISABLE\_ROUTE\_CC**

\$sc\_\$cpu\_SB\_DisRoute

**Global CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC**  
\$sc\_\$cpu\_SB\_DisSubRptg

**Global CFE\_SB\_ENABLE\_ROUTE\_CC**  
\$sc\_\$cpu\_SB\_EnaRoute

**Global CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC**  
\$sc\_\$cpu\_SB\_EnaSubRptg

**Global CFE\_SB\_NOOP\_CC**  
\$sc\_\$cpu\_SB\_NOOP

**Global CFE\_SB\_RESET\_COUNTERS\_CC**  
\$sc\_\$cpu\_SB\_ResetCtrs

**Global CFE\_SB\_SEND\_PREV\_SUBS\_CC**  
\$sc\_\$cpu\_SB\_SendPrevSubs

**Global CFE\_SB\_SEND\_SB\_STATS\_CC**  
\$sc\_\$cpu\_SB\_DumpStats

**Global CFE\_SB\_WRITE\_MAP\_INFO\_CC**  
\$sc\_\$cpu\_SB\_WriteMap2File

**Global CFE\_SB\_WRITE\_PIPE\_INFO\_CC**  
\$sc\_\$cpu\_SB\_WritePipe2File

**Global CFE\_SB\_WRITE\_ROUTING\_INFO\_CC**  
\$sc\_\$cpu\_SB\_WriteRouting2File

**Global CFE\_TBL\_ABORT\_LOAD\_CC**  
\$sc\_\$cpu\_TBL\_LoadAbort

**Global CFE\_TBL\_ACTIVATE\_CC**  
\$sc\_\$cpu\_TBL\_Activate

**Global CFE\_TBL\_DELETE\_CDS\_CC**  
\$sc\_\$cpu\_TBL\_DeleteCDS

**Global CFE\_TBL\_DUMP\_CC**  
\$sc\_\$cpu\_TBL\_Dump

**Global CFE\_TBL\_DUMP\_REGISTRY\_CC**  
\$sc\_\$cpu\_TBL\_WriteReg2File

**Global CFE\_TBL\_LOAD\_CC**  
\$sc\_\$cpu\_TBL\_Load

**Global CFE\_TBL\_NOOP\_CC**  
\$sc\_\$cpu\_TBL\_Noop

**Global CFE\_TBL\_RESET\_COUNTERS\_CC**  
\$sc\_\$cpu\_TBL\_ResetCtrs

**Global CFE\_TBL\_SEND\_REGISTRY\_CC**  
\$sc\_\$cpu\_TBL\_TLMReg

**Global CFE\_TBL\_VALIDATE\_CC**  
\$sc\_\$cpu\_TBL\_Validate

**Global CFE\_TIME\_ADD\_1HZ\_ADJUSTMENT\_CC**  
\$sc\_\$cpu\_TIME\_Add1HzSTCF

**Global CFE\_TIME\_ADD\_ADJUST\_CC**  
\$sc\_\$cpu\_TIME\_AddSTCFAadj

**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\_SubSTCFAadj

**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_CFE_MINORVER

Global CFE_ES_HousekeepingTlm_Payload::CFEMissionRevision
$sc_$cpu_ES_CFEMISSIONREV
```

**Global CFE\_ES\_HousekeepingTlm\_Payload::CFERevision**  
\$sc\_\$cpu\_ES\_CFEREVISION

**Global CFE\_ES\_HousekeepingTlm\_Payload::CommandCounter**  
\$sc\_\$cpu\_ES\_CMDPC

**Global CFE\_ES\_HousekeepingTlm\_Payload::CommandErrorCounter**  
\$sc\_\$cpu\_ES\_CMDEC

**Global CFE\_ES\_HousekeepingTlm\_Payload::ERLogEntries**  
\$sc\_\$cpu\_ES\_ERLOGENTRIES

**Global CFE\_ES\_HousekeepingTlm\_Payload::ERLogIndex**  
\$sc\_\$cpu\_ES\_ERLOGINDEX

**Global CFE\_ES\_HousekeepingTlm\_Payload::HeapBlocksFree**  
\$sc\_\$cpu\_ES\_HeapBlocksFree

**Global CFE\_ES\_HousekeepingTlm\_Payload::HeapBytesFree**  
\$sc\_\$cpu\_ES\_HeapBytesFree

**Global CFE\_ES\_HousekeepingTlm\_Payload::HeapMaxBlockSize**  
\$sc\_\$cpu\_ES\_HeapMaxBlkSize

**Global CFE\_ES\_HousekeepingTlm\_Payload::MaxProcessorResets**  
\$sc\_\$cpu\_ES\_MaxProcResets

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALMajorVersion**  
\$sc\_\$cpu\_ES\_OSMAJORVER

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALMinorVersion**  
\$sc\_\$cpu\_ES\_OSMINORVER

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALMissionRevision**  
\$sc\_\$cpu\_ES\_OSMISSIONREV

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALRevision**  
\$sc\_\$cpu\_ES\_OSREVISION

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfDataCount**  
\$sc\_\$cpu\_ES\_PerfDataCnt

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfDataEnd**  
\$sc\_\$cpu\_ES\_PerfDataEnd

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfDataStart**  
\$sc\_\$cpu\_ES\_PerfDataStart

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfData2Write**  
\$sc\_\$cpu\_ES\_PerfData2Write

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfFilterMask [CFE\_MISSION\_ES\_PERF\_MAX\_IDS/32]**  
\$sc\_\$cpu\_ES\_PerfFltrMask[MaskCnt]

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfMode**  
\$sc\_\$cpu\_ES\_PerfMode

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfState**  
\$sc\_\$cpu\_ES\_PerfState

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfTriggerCount**  
\$sc\_\$cpu\_ES\_PerfTrigCnt

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

Global CFE_ES_HousekeepingTlm_Payload::ProcessorResets
$sc_$cpu_ES_ProcResetCnt

Global CFE_ES_HousekeepingTlm_Payload::PSPMajorVersion
$sc_$cpu_ES_PSPMAJORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMinorVersion
$sc_$cpu_ES_PSPMINORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMissionRevision
$sc_$cpu_ES_PSPMISSIONREV

Global CFE_ES_HousekeepingTlm_Payload::PSPRevision
$sc_$cpu_ES_PSPREVISION

Global CFE_ES_HousekeepingTlm_Payload::RegisteredCoreApps
$sc_$cpu_ES_RegCoreApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredExternalApps
$sc_$cpu_ES_RegExtApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredLibs
$sc_$cpu_ES_RegLibs

Global CFE_ES_HousekeepingTlm_Payload::RegisteredTasks
$sc_$cpu_ES_RegTasks

Global CFE_ES_HousekeepingTlm_Payload::ResetSubtype
$sc_$cpu_ES_ResetSubtype

Global CFE_ES_HousekeepingTlm_Payload::ResetType
$sc_$cpu_ES_ResetType

Global CFE_ES_HousekeepingTlm_Payload::SysLogBytesUsed
$sc_$cpu_ES_SYSLOGBYTEUSED

Global CFE_ES_HousekeepingTlm_Payload::SysLogEntries
$sc_$cpu_ES_SYSLOGENTRIES

Global CFE_ES_HousekeepingTlm_Payload::SysLogMode
$sc_$cpu_ES_SYSLOGMODE

Global CFE_ES_HousekeepingTlm_Payload::SysLogSize
$sc_$cpu_ES_SYSLOGSIZE

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

Global CFE_ES_MemPoolStats::CheckErrCtr
$sc_$cpu_ES_BlkErrCTR

Global CFE_ES_MemPoolStats::NumBlocksRequested
$sc_$cpu_ES_BlkREQ

Global CFE_ES_MemPoolStats::NumFreeBytes
$sc_$cpu_ES_FreeBytes

Global CFE_ES_MemPoolStats::PoolSize
$sc_$cpu_ES_PoolSize
```

**Global CFE\_ES\_PoolStatsTlm\_Payload::PoolHandle**  
\$sc\_\$cpu\_ES\_PoolHandle

**Global CFE\_EVS\_AppTlmData::AppEnableStatus**  
\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS].APPENABLESTAT

**Global CFE\_EVS\_AppTlmData::AppID**  
\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS].APPID

**Global CFE\_EVS\_AppTlmData::AppMessageSentCounter**  
\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS].APPMGSENTC

**Global CFE\_EVS\_AppTlmData::AppMessageSquelchedCounter**  
\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS].SQUELCHEDC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::AppData [CFE\_MISSION\_ES\_MAX\_APPLICATIONS]**  
\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS]

**Global CFE\_EVS\_HousekeepingTlm\_Payload::CommandCounter**  
\$sc\_\$cpu\_EVS\_CMDPC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::CommandErrorCounter**  
\$sc\_\$cpu\_EVS\_CMDEC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::LogEnabled**  
\$sc\_\$cpu\_EVS\_LOGENABLED

**Global CFE\_EVS\_HousekeepingTlm\_Payload::LogFullFlag**  
\$sc\_\$cpu\_EVS\_LOGFULL

**Global CFE\_EVS\_HousekeepingTlm\_Payload::LogMode**  
\$sc\_\$cpu\_EVS\_LOGMODE

**Global CFE\_EVS\_HousekeepingTlm\_Payload::LogOverflowCounter**  
\$sc\_\$cpu\_EVS\_LOGOVERFLOWC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::MessageFormatMode**  
\$sc\_\$cpu\_EVS\_MSGFMTMODE

**Global CFE\_EVS\_HousekeepingTlm\_Payload::MessageSendCounter**  
\$sc\_\$cpu\_EVS\_MSGSENTC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::MessageTruncCounter**  
\$sc\_\$cpu\_EVS\_MSGTRUNC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::OutputPort**  
\$sc\_\$cpu\_EVS\_OUTPUTPORT

**Global CFE\_EVS\_HousekeepingTlm\_Payload::Spare1**  
\$sc\_\$cpu\_EVS\_HK\_SPARE1

**Global CFE\_EVS\_HousekeepingTlm\_Payload::Spare2**  
\$sc\_\$cpu\_EVS\_HK\_SPARE2

**Global CFE\_EVS\_HousekeepingTlm\_Payload::Spare3**  
\$sc\_\$cpu\_EVS\_HK\_SPARE3

**Global CFE\_EVS\_HousekeepingTlm\_Payload::UnregisteredAppCounter**  
\$sc\_\$cpu\_EVS\_UNREGAPPC

**Global CFE\_EVS\_LongEventTlm\_Payload::Message [CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH]**  
\$sc\_\$cpu\_EVENT[CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH]

```
Global CFE_EVS_LongEventTlm_Payload::Spare1
$sc_$cpu_EVS_SPARE1

Global CFE_EVS_LongEventTlm_Payload::Spare2
$sc_$cpu_EVS_SPARE2

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

Global CFE_EVS_PacketID::EventID
$sc_$cpu_EVS_EVENTID

Global CFE_EVS_PacketID::EventType
$sc_$cpu_EVS_EVENTTYPE

Global CFE_EVS_PacketID::ProcessorID
$sc_$cpu_EVS_PROCESSORID

Global CFE_EVS_PacketID::SpacecraftID
$sc_$cpu_EVS_SCID

Global CFE_SB_HousekeepingTlm_Payload::CommandCounter
$sc_$cpu_SB_CMDPC

Global CFE_SB_HousekeepingTlm_Payload::CommandErrorCounter
$sc_$cpu_SB_CMDEC

Global CFE_SB_HousekeepingTlm_Payload::CreatePipeErrorCounter
$sc_$cpu_SB_NewPipeEC

Global CFE_SB_HousekeepingTlm_Payload::DuplicateSubscriptionsCounter
$sc_$cpu_SB_DupSubCnt

Global CFE_SB_HousekeepingTlm_Payload::GetPipeldByNameErrorCounter
$sc_$cpu_SB_GetPipeIDByNameEC

Global CFE_SB_HousekeepingTlm_Payload::InternalErrorCounter
$sc_$cpu_SB_InternalEC

Global CFE_SB_HousekeepingTlm_Payload::MemInUse
$sc_$cpu_SB_MemInUse

Global CFE_SB_HousekeepingTlm_Payload::MemPoolHandle
$sc_$cpu_SB_MemPoolHdl

Global CFE_SB_HousekeepingTlm_Payload::MsgLimitErrorCounter
$sc_$cpu_SB_MsgLimEC

Global CFE_SB_HousekeepingTlm_Payload::MsgReceiveErrorCounter
$sc_$cpu_SB_MsgRecEC

Global CFE_SB_HousekeepingTlm_Payload::MsgSendErrorCounter
$sc_$cpu_SB_MsgSndEC

Global CFE_SB_HousekeepingTlm_Payload::NoSubscribersCounter
$sc_$cpu_SB_NoSubEC

Global CFE_SB_HousekeepingTlm_Payload::PipeOptsErrorCounter
$sc_$cpu_SB_PipeOptsEC

Global CFE_SB_HousekeepingTlm_Payload::PipeOverflowErrorCounter
$sc_$cpu_SB_PipeOvrEC
```

**Global CFE\_SB\_HousekeepingTlm\_Payload::Spare2Align [1]**  
\$sc\_\$cpu\_SB\_Spare2Align[2]

**Global CFE\_SB\_HousekeepingTlm\_Payload::SubscribeErrorCounter**  
\$sc\_\$cpu\_SB\_SubscrEC

**Global CFE\_SB\_HousekeepingTlm\_Payload::UnmarkedMem**  
\$sc\_\$cpu\_SB\_UnMarkedMem

**Global CFE\_SB\_PipeDepthStats::CurrentQueueDepth**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDINUSE

**Global CFE\_SB\_PipeDepthStats::MaxQueueDepth**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDDEPTH

**Global CFE\_SB\_PipeDepthStats::PeakQueueDepth**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDPKINUSE

**Global CFE\_SB\_PipeDepthStats::PipeId**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDPipeID

**Global CFE\_SB\_PipeDepthStats::Spare**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDSpare

**Global CFE\_SB\_StatsTlm\_Payload::MaxMemAllowed**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMMBMALW

**Global CFE\_SB\_StatsTlm\_Payload::MaxMsgIdsAllowed**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMMMDALW

**Global CFE\_SB\_StatsTlm\_Payload::MaxPipeDepthAllowed**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMMPDALW

**Global CFE\_SB\_StatsTlm\_Payload::MaxPipesAllowed**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMMPALW

**Global CFE\_SB\_StatsTlm\_Payload::MaxSubscriptionsAllowed**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMMSALW

**Global CFE\_SB\_StatsTlm\_Payload::MemInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMBMIU

**Global CFE\_SB\_StatsTlm\_Payload::MsgIdsInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMMIDIU

**Global CFE\_SB\_StatsTlm\_Payload::PeakMemInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPBMIU

**Global CFE\_SB\_StatsTlm\_Payload::PeakMsgIdsInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPMIDIU

**Global CFE\_SB\_StatsTlm\_Payload::PeakPipesInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPPIU

**Global CFE\_SB\_StatsTlm\_Payload::PeakSBBuffersInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPSSBIU

**Global CFE\_SB\_StatsTlm\_Payload::PeakSubscriptionsInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPSIU

**Global CFE\_SB\_StatsTlm\_Payload::PipeDepthStats [CFE\_MISSION\_SB\_MAX\_PIPES]**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES]

**Global CFE\_SB\_StatsTlm\_Payload::PipesInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPIU

**Global CFE\_SB\_StatsTlm\_Payload::SBBuffersInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMSBBIU

**Global CFE\_SB\_StatsTlm\_Payload::SubscriptionsInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMSIU

**Global CFE\_TBL\_HousekeepingTlm\_Payload::ActiveBuffer**  
\$sc\_\$cpu\_TBL\_LastValBuf

**Global CFE\_TBL\_HousekeepingTlm\_Payload::ByteAlignPad1**  
\$sc\_\$cpu\_TBL\_BytAlignPad1

**Global CFE\_TBL\_HousekeepingTlm\_Payload::CommandCounter**  
\$sc\_\$cpu\_TBL\_CMDPC

**Global CFE\_TBL\_HousekeepingTlm\_Payload::CommandErrorCounter**  
\$sc\_\$cpu\_TBL\_CMDEC

**Global CFE\_TBL\_HousekeepingTlm\_Payload::FailedValCounter**  
\$sc\_\$cpu\_TBL\_ValFailedCtr

**Global CFE\_TBL\_HousekeepingTlm\_Payload::LastFileDumped [CFE\_MISSION\_MAX\_PATH\_LEN]**  
\$sc\_\$cpu\_TBL\_LastFileDumped[OS\_MAX\_PATH\_LEN]

**Global CFE\_TBL\_HousekeepingTlm\_Payload::LastFileLoaded [CFE\_MISSION\_MAX\_PATH\_LEN]**  
\$sc\_\$cpu\_TBL\_LastFileLoaded[OS\_MAX\_PATH\_LEN]

**Global CFE\_TBL\_HousekeepingTlm\_Payload::LastTableLoaded [CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN]**  
\$sc\_\$cpu\_TBL\_LastTableLoaded[CFE\_TBL\_MAX\_FULL\_NAME\_LEN]

**Global CFE\_TBL\_HousekeepingTlm\_Payload::LastUpdatedTable [CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_L←EN]**  
\$sc\_\$cpu\_TBL\_LastUpdTblName[CFE\_TB\_MAX\_FULL\_NAME\_LEN]

**Global CFE\_TBL\_HousekeepingTlm\_Payload::LastUpdateTime**  
\$sc\_\$cpu\_TBL\_LastUpdTTime, \$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\_TLUUSUBSECONDS

**Global CFE\_TBL\_TblRegPacket\_Payload::ValidationFuncPtr**  
\$sc\_\$cpu\_TBL\_ValFuncPtr

**Global CFE\_TIME\_DiagnosticTlm\_Payload::AtToneDelay**  
\$sc\_\$cpu\_TIME\_DLlatentS, \$sc\_\$cpu\_TIME\_DLlatentSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::AtToneLatch**  
\$sc\_\$cpu\_TIME\_DTVValidS, \$sc\_\$cpu\_TIME\_DTVValidSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::AtToneLeapSeconds**  
\$sc\_\$cpu\_TIME\_DLearpS

**Global CFE\_TIME\_DiagnosticTlm\_Payload::AtToneMET**  
\$sc\_\$cpu\_TIME\_DMETS, \$sc\_\$cpu\_TIME\_DMETSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::AtToneSTCF**  
\$sc\_\$cpu\_TIME\_DSTCFS, \$sc\_\$cpu\_TIME\_DSTCFSS

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ClockFlyState**  
\$sc\_\$cpu\_TIME\_DFlywheel

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ClockSetState**  
\$sc\_\$cpu\_TIME\_DValid

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ClockSignal**  
\$sc\_\$cpu\_TIME\_DSignal

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ClockSource**  
\$sc\_\$cpu\_TIME\_DSource

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ClockStateAPI**  
\$sc\_\$cpu\_TIME\_DAPISate

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ClockStateFlags**  
\$sc\_\$cpu\_TIME\_DStateFlags, \$sc\_\$cpu\_TIME\_DFlagSet, \$sc\_\$cpu\_TIME\_DFlagFly, \$sc\_\$cpu\_TIME\_DFlagSrc,  
\$sc\_\$cpu\_TIME\_DFlagPri, \$sc\_\$cpu\_TIME\_DFlagSfly, \$sc\_\$cpu\_TIME\_DFlagCfly, \$sc\_\$cpu\_TIME\_DFlagAdj,  
\$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\_DLcalS, \$sc\_\$cpu\_TIME\_DLcalSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::CurrentMET**  
\$sc\_\$cpu\_TIME\_DMETS, \$sc\_\$cpu\_TIME\_DMETSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::CurrentTAI**  
\$sc\_\$cpu\_TIME\_DTAIS, \$sc\_\$cpu\_TIME\_DTAISS

**Global CFE\_TIME\_DiagnosticTlm\_Payload::CurrentUTC**  
\$sc\_\$cpu\_TIME\_DUTCS, \$sc\_\$cpu\_TIME\_DUTCSS

**Global CFE\_TIME\_DiagnosticTlm\_Payload::DataStoreStatus**  
\$sc\_\$cpu\_TIME\_DataStStat

**Global CFE\_TIME\_DiagnosticTlm\_Payload::DelayDirection**  
\$sc\_\$cpu\_TIME\_DLlatentDir

**Global CFE\_TIME\_DiagnosticTlm\_Payload::Forced2Fly**  
\$sc\_\$cpu\_TIME\_DCMD2Fly

**Global CFE\_TIME\_DiagnosticTlm\_Payload::LocalIntCounter**  
\$sc\_\$cpu\_TIME\_D1HzSRCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::LocalTaskCounter**  
\$sc\_\$cpu\_TIME\_D1HzTaskCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::MaxElapsed**  
\$sc\_\$cpu\_TIME\_DMaxWindow

**Global CFE\_TIME\_DiagnosticTlm\_Payload::MaxLocalClock**  
\$sc\_\$cpu\_TIME\_DWrapS, \$sc\_\$cpu\_TIME\_DWrapSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::MinElapsed**  
\$sc\_\$cpu\_TIME\_DMinWindow

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneHzAdjust**  
\$sc\_\$cpu\_TIME\_D1HzAdjS, \$sc\_\$cpu\_TIME\_D1HzAdjSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneHzDirection**  
\$sc\_\$cpu\_TIME\_D1HzAdjDir

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneTimeAdjust**  
\$sc\_\$cpu\_TIME\_DAdjustS, \$sc\_\$cpu\_TIME\_DAdjustSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneTimeDirection**  
\$sc\_\$cpu\_TIME\_DAdjustDir

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ServerFlyState**  
\$sc\_\$cpu\_TIME\_DSrvFly

**Global CFE\_TIME\_DiagnosticTlm\_Payload::TimeSinceTone**  
\$sc\_\$cpu\_TIME\_DElapsedS, \$sc\_\$cpu\_TIME\_DElapsedSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ToneDataCounter**  
\$sc\_\$cpu\_TIME\_DTatTCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ToneDataLatch**  
\$sc\_\$cpu\_TIME\_DTDS, \$sc\_\$cpu\_TIME\_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\_DTTSSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ToneTaskCounter**  
\$sc\_\$cpu\_TIME\_DTsTaskCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ToneUnderLimit**  
\$sc\_\$cpu\_TIME\_DMinSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::VersionCounter**  
\$sc\_\$cpu\_TIME\_DVersionCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::VirtualMET**

\$sc\_\$cpu\_TIME\_DLLogicalMET

**Global CFE\_TIME\_HousekeepingTlm\_Payload::ClockStateAPI**

\$sc\_\$cpu\_TIME\_DAPISate

**Global CFE\_TIME\_HousekeepingTlm\_Payload::ClockStateFlags**

\$sc\_\$cpu\_TIME\_StateFlg, \$sc\_\$cpu\_TIME\_FlagSet, \$sc\_\$cpu\_TIME\_FlagFly, \$sc\_\$cpu\_TIME\_FlagSrc, \$sc\_\$cpu\_TIME\_FlagPri, \$sc\_\$cpu\_TIME\_FlagSfly, \$sc\_\$cpu\_TIME\_FlagCfly, \$sc\_\$cpu\_TIME\_FlagAdj, \$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. <b>NOTE: This is different from a CCSDS Application ID which is referred to as an "APID."</b>
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.

Term	Definition
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 <a href="#">Event Message</a> .
Event Filter	A numeric value (bit mask) used to determine how frequently to output an application Event Message defined by its <a href="#">Event ID</a> .
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 <a href="#">CFE_EVS_EventType_DEBUG</a> , <a href="#">CFE_EVS_EventType_INFORMATION</a> , <a href="#">CFE_EVS_EventType_ERROR</a> , or <a href="#">CFE_EVS_EventType_CRITICAL</a> .
Event Message	A data item used to notify the user and/or an external <a href="#">Application</a> of a significant event. Event Messages include a time-stamp of when the message was generated, a processor unique identifier, an <a href="#">Application ID</a> , the <a href="#">Event Type</a> (DEBUG,INFO,ERROR or CRITICAL), and <a href="#">Event Data</a> . An Event Message can either be real-time or playback from a Local Event Log.

## 4 cFE Application Programmer's Interface (API) Reference

### 4.1 Executive Services API

- [cFE Entry/Exit APIs](#)
  - [CFE\\_ES\\_Main](#) - cFE Main Entry Point used by Board Support Package to start cFE
  - [CFE\\_ES\\_ResetCFE](#) - Reset the cFE Core and all cFE Applications.
- [cFE Application Control APIs](#)
  - [CFE\\_ES\\_RestartApp](#) - Restart a single cFE Application.
  - [CFE\\_ES\\_ReloadApp](#) - Reload a single cFE Application.
  - [CFE\\_ES\\_DeleteApp](#) - Delete a cFE Application.
- [cFE Application Behavior APIs](#)
  - [CFE\\_ES\\_RunLoop](#) - Check for Exit, Restart, or Reload commands.
  - [CFE\\_ES\\_WaitForStartupSync](#) - Allow an Application to Wait for the "OPERATIONAL" global system state.

- [CFE\\_ES\\_WaitForSystemState](#) - Allow an Application to Wait for a minimum global system state.
  - [CFE\\_ES\\_IncrementTaskCounter](#) - Increments the execution counter for the calling task.
  - [CFE\\_ES\\_ExitApp](#) - Exit a cFE Application.
- [cFE Information APIs](#)
    - [CFE\\_ES\\_GetResetType](#) - Return the most recent Reset Type.
    - [CFE\\_ES\\_GetAppID](#) - Get an Application ID for the calling Application.
    - [CFE\\_ES\\_GetTaskID](#) - Get the task ID of the calling context.
    - [CFE\\_ES\\_GetAppIDByName](#) - Get an Application ID associated with a specified Application name.
    - [CFE\\_ES\\_GetLibIDByName](#) - Get a Library ID associated with a specified Library name.
    - [CFE\\_ES\\_GetAppName](#) - Get an Application name for a specified Application ID.
    - [CFE\\_ES\\_GetLibName](#) - Get a Library name for a specified Library ID.
    - [CFE\\_ES\\_GetAppInfo](#) - Get Application Information given a specified App ID.
    - [CFE\\_ES\\_GetTaskInfo](#) - Get Task Information given a specified Task ID.
    - [CFE\\_ES\\_GetLibInfo](#) - Get Library Information given a specified Resource ID.
    - [CFE\\_ES\\_GetModuleInfo](#) - Get Information given a specified Resource ID.
  - [cFE Child Task APIs](#)
    - [CFE\\_ES\\_CreateChildTask](#) - Creates a new task under an existing Application.
    - [CFE\\_ES\\_GetTaskIDByName](#) - Get a Task ID associated with a specified Task name.
    - [CFE\\_ES\\_GetTaskName](#) - Get a Task name for a specified Task ID.
    - [CFE\\_ES\\_DeleteChildTask](#) - Deletes a task under an existing Application.
    - [CFE\\_ES\\_ExitChildTask](#) - Exits a child task.
  - [cFE Critical Data Store APIs](#)
    - [CFE\\_ES\\_RegisterCDS](#) - Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)
    - [CFE\\_ES\\_GetCDSBlockIDByName](#) - Get a CDS Block ID associated with a specified CDS Block name.
    - [CFE\\_ES\\_GetCDSBlockName](#) - Get a Block name for a specified Block ID.
    - [CFE\\_ES\\_CopyToCDS](#) - Save a block of data in the Critical Data Store (CDS)
    - [CFE\\_ES\\_RestoreFromCDS](#) - Recover a block of data from the Critical Data Store (CDS)
  - [cFE Memory Manager APIs](#)
    - [CFE\\_ES\\_PoolCreate](#) - Initializes a memory pool created by an application while using a semaphore during processing.
    - [CFE\\_ES\\_PoolCreateEx](#) - Initializes a memory pool created by an application with application specified block sizes.
    - [CFE\\_ES\\_PoolCreateNoSem](#) - Initializes a memory pool created by an application without using a semaphore during processing.
    - [CFE\\_ES\\_PoolDelete](#) - Deletes a memory pool that was previously created.
    - [CFE\\_ES\\_GetPoolBuf](#) - Gets a buffer from the memory pool created by [CFE\\_ES\\_PoolCreate](#) or [CFE\\_ES\\_PoolCreateNoSem](#).
    - [CFE\\_ES\\_PutPoolBuf](#) - Releases a buffer from the memory pool that was previously allocated via [CFE\\_ES\\_GetPoolBuf](#).
    - [CFE\\_ES\\_GetMemPoolStats](#) - Extracts the statistics maintained by the memory pool software.

- [CFE\\_ES\\_GetPoolBufInfo](#) - Gets info on a buffer previously allocated via [CFE\\_ES\\_GetPoolBuf](#).
- cFE Performance Monitor APIs
  - [CFE\\_ES\\_PerfLogEntry](#) - Entry marker for use with Software Performance Analysis Tool.
  - [CFE\\_ES\\_PerfLogExit](#) - Exit marker for use with Software Performance Analysis Tool.
  - [CFE\\_ES\\_PerfLogAdd](#) - Adds a new entry to the data buffer.
- cFE Generic Counter APIs
  - [CFE\\_ES\\_RegisterGenCounter](#) - Register a generic counter.
  - [CFE\\_ES\\_DeleteGenCounter](#) - Delete a generic counter.
  - [CFE\\_ES\\_IncrementGenCounter](#) - Increments the specified generic counter.
  - [CFE\\_ES\\_SetGenCount](#) - Set the specified generic counter.
  - [CFE\\_ES\\_GetGenCount](#) - Get the specified generic counter count.
  - [CFE\\_ES\\_GetGenCounterIDByName](#) - Get the Id associated with a generic counter name.
  - [CFE\\_ES\\_GetGenCounterName](#) - Get a Counter name for a specified Counter ID.
- cFE Miscellaneous APIs
  - [CFE\\_ES\\_BackgroundWakeUp](#) - Wakes up the CFE background task.
  - [CFE\\_ES\\_CalculateCRC](#) - Calculate a CRC on a block of memory.
  - [CFE\\_ES\\_WriteToSysLog](#) - Write a string to the cFE System Log.
  - [CFE\\_ES\\_ProcessAsyncEvent](#) - Notification that an asynchronous event was detected by the underlying OS/PSP.
  - [CFE\\_ES\\_StatusToString](#) - Convert status to a string.
- cFE Resource ID APIs
  - [CFE\\_ES\\_AppID\\_ToIndex](#) - Obtain an index value correlating to an ES Application ID.
  - [CFE\\_ES\\_LibID\\_ToIndex](#) - Obtain an index value correlating to an ES Library ID.
  - [CFE\\_ES\\_TaskID\\_ToIndex](#) - Obtain an index value correlating to an ES Task ID.
  - [CFE\\_ES\\_CounterID\\_ToIndex](#) - Obtain an index value correlating to an ES Counter ID.

## 4.2 Events Services API

- cFE Registration APIs
  - [CFE\\_EVS\\_Register](#) - Register an application for receiving event services.
- cFE Send Event APIs
  - [CFE\\_EVS\\_SendEvent](#) - Generate a software event.
  - [CFE\\_EVS\\_SendEventWithAppID](#) - Generate a software event given the specified Application ID.
  - [CFE\\_EVS\\_SendTimedEvent](#) - Generate a software event with a specific time tag.
- cFE Reset Event Filter APIs
  - [CFE\\_EVS\\_ResetFilter](#) - Resets the calling application's event filter for a single event ID.
  - [CFE\\_EVS\\_ResetAllFilters](#) - Resets all of the calling application's event filters.

### 4.3 File Services API

- cFE File Header Management APIs
  - [CFE\\_FS\\_ReadHeader](#) - Read the contents of the Standard cFE File Header.
  - [CFE\\_FS\\_InitHeader](#) - Initializes the contents of the Standard cFE File Header.
  - [CFE\\_FS\\_WriteHeader](#) - Write the specified Standard cFE File Header to the specified file.
  - [CFE\\_FS\\_SetTimestamp](#) - Modifies the Time Stamp field in the Standard cFE File Header for the specified file.
- cFE File Utility APIs
  - [CFE\\_FS\\_GetDefaultMountPoint](#) - Get the default virtual mount point for a file category.
  - [CFE\\_FS\\_GetDefaultExtension](#) - Get the default filename extension for a file category.
  - [CFE\\_FS\\_ParseInputFileNameEx](#) - Parse a filename input from an input buffer into a local buffer.
  - [CFE\\_FS\\_ParseInputFileName](#) - Parse a filename string from the user into a local buffer.
  - [CFE\\_FS\\_ExtractFilenameFromPath](#) - Extracts the filename from a unix style path and filename string.
  - [CFE\\_FS\\_BackgroundFileDumpRequest](#) - Register a background file dump request.
  - [CFE\\_FS\\_BackgroundFileDumpsPending](#) - Query if a background file write request is currently pending.

### 4.4 Message API

- cFE Generic Message APIs
  - [CFE\\_MSG\\_Init](#) - Initialize a message.
- cFE Message Primary Header APIs
  - [CFE\\_MSG.GetSize](#) - Gets the total size of a message.
  - [CFE\\_MSG\\_SetSize](#) - Sets the total size of a message.
  - [CFE\\_MSG.GetType](#) - Gets the message type.
  - [CFE\\_MSG\\_SetType](#) - Sets the message type.
  - [CFE\\_MSG\\_GetHeaderVersion](#) - Gets the message header version.
  - [CFE\\_MSG\\_SetHeaderVersion](#) - Sets the message header version.
  - [CFE\\_MSG\\_GetHasSecondaryHeader](#) - Gets the message secondary header boolean.
  - [CFE\\_MSG\\_SetHasSecondaryHeader](#) - Sets the message secondary header boolean.
  - [CFE\\_MSG\\_GetApId](#) - Gets the message application ID.
  - [CFE\\_MSG\\_SetApId](#) - Sets the message application ID.
  - [CFE\\_MSG\\_GetSegmentationFlag](#) - Gets the message segmentation flag.
  - [CFE\\_MSG\\_SetSegmentationFlag](#) - Sets the message segmentation flag.
  - [CFE\\_MSG\\_GetSequenceCount](#) - Gets the message sequence count.
  - [CFE\\_MSG\\_SetSequenceCount](#) - Sets the message sequence count.
  - [CFE\\_MSG\\_GetNextSequenceCount](#) - Gets the next sequence count value (rolls over if appropriate)
- cFE Message Extended Header APIs
  - [CFE\\_MSG\\_GetEDSVersion](#) - Gets the message EDS version.
  - [CFE\\_MSG\\_SetEDSVersion](#) - Sets the message EDS version.
  - [CFE\\_MSG\\_GetEndian](#) - Gets the message endian.

- [CFE\\_MSG\\_SetEndian](#) - Sets the message endian.
- [CFE\\_MSG\\_GetPlaybackFlag](#) - Gets the message playback flag.
- [CFE\\_MSG\\_SetPlaybackFlag](#) - Sets the message playback flag.
- [CFE\\_MSG\\_GetSubsystem](#) - Gets the message subsystem.
- [CFE\\_MSG\\_SetSubsystem](#) - Sets the message subsystem.
- [CFE\\_MSG\\_GetSystem](#) - Gets the message system.
- [CFE\\_MSG\\_SetSystem](#) - Sets the message system.
- cFE Message Secondary Header APIs
  - [CFE\\_MSG\\_GenerateChecksum](#) - Calculates and sets the checksum of a message.
  - [CFE\\_MSG\\_ValidateChecksum](#) - Validates the checksum of a message.
  - [CFE\\_MSG\\_SetFcnCode](#) - Sets the function code field in a message.
  - [CFE\\_MSG\\_GetFcnCode](#) - Gets the function code field from a message.
  - [CFE\\_MSG\\_GetMsgTime](#) - Gets the time field from a message.
  - [CFE\\_MSG\\_SetMsgTime](#) - Sets the time field in a message.
- cFE Message Id APIs
  - [CFE\\_MSG\\_GetMsgId](#) - Gets the message id from a message.
  - [CFE\\_MSG\\_SetMsgId](#) - Sets the message id bits in a message.
  - [CFE\\_MSG.GetTypeFromMsgId](#) - Gets message type using message ID.

## 4.5 Resource ID API

- cFE Resource Misc APIs
  - [CFE\\_Resourceld\\_ToInteger](#) - Convert a resource ID to an integer.
  - [CFE\\_Resourceld\\_FromInteger](#) - Convert an integer to a resource ID.
  - [CFE\\_Resourceld\\_Equal](#) - Compare two Resource ID values for equality.
  - [CFE\\_Resourceld\\_IsDefined](#) - Check if a resource ID value is defined.
  - [CFE\\_Resourceld\\_GetBase](#) - Get the Base value (type/category) from a resource ID value.
  - [CFE\\_Resourceld\\_GetSerial](#) - Get the Serial Number (sequential ID) from a resource ID value.
  - [CFE\\_Resourceld\\_FindNext](#) - Locate the next resource ID which does not map to an in-use table entry.
  - [CFE\\_Resourceld\\_ToIndex](#) - Internal routine to aid in converting an ES resource ID to an array index.

## 4.6 Software Bus Services API

- cFE Pipe Management APIs
  - [CFE\\_SB\\_CreatePipe](#) - Creates a new software bus pipe.
  - [CFE\\_SB\\_DeletePipe](#) - Delete a software bus pipe.
  - [CFE\\_SB\\_Pipeld\\_ToIndex](#) - Obtain an index value correlating to an SB Pipe ID.
  - [CFE\\_SB\\_SetPipeOpts](#) - Set options on a pipe.
  - [CFE\\_SB\\_GetPipeOpts](#) - Get options on a pipe.
  - [CFE\\_SB\\_GetPipeName](#) - Get the pipe name for a given id.
  - [CFE\\_SB\\_GetPipeldByName](#) - Get pipe id by pipe name.
- cFE Message Subscription Control APIs

- [CFE\\_SB\\_Subscribe](#) - Subscribe to a message on the software bus with default parameters.
- [CFE\\_SB\\_SubscribeEx](#) - Subscribe to a message on the software bus.
- [CFE\\_SB\\_SubscribeLocal](#) - Subscribe to a message while keeping the request local to a cpu.
- [CFE\\_SB\\_Unsubscribe](#) - Remove a subscription to a message on the software bus.
- [CFE\\_SB\\_UnsubscribeLocal](#) - Remove a subscription to a message on the software bus on the current CPU.
- [cFE Send/Receive Message APIs](#)
  - [CFE\\_SB\\_TransmitMsg](#) - Transmit a message.
  - [CFE\\_SB\\_ReceiveBuffer](#) - Receive a message from a software bus pipe.
- [cFE Zero Copy APIs](#)
  - [CFE\\_SB\\_AllocateMessageBuffer](#) - Get a buffer pointer to use for "zero copy" SB sends.
  - [CFE\\_SB\\_ReleaseMessageBuffer](#) - Release an unused "zero copy" buffer pointer.
  - [CFE\\_SB\\_TransmitBuffer](#) - Transmit a buffer.
- [cFE Message Characteristics APIs](#)
  - [CFE\\_SB\\_SetUserDataLength](#) - Sets the length of user data in a software bus message.
  - [CFE\\_SB\\_TimeStampMsg](#) - Sets the time field in a software bus message with the current spacecraft time.
  - [CFE\\_SB\\_MessageStringSet](#) - Copies a string into a software bus message.
  - [CFE\\_SB\\_GetUserData](#) - Get a pointer to the user data portion of a software bus message.
  - [CFE\\_SB\\_GetUserDataLength](#) - Gets the length of user data in a software bus message.
  - [CFE\\_SB\\_MessageStringGet](#) - Copies a string out of a software bus message.
- [cFE Message ID APIs](#)
  - [CFE\\_SB\\_IsValidMsgId](#) - Identifies whether a given [CFE\\_SB\\_MsgId\\_t](#) is valid.
  - [CFE\\_SB\\_MsgId\\_Equal](#) - Identifies whether two [CFE\\_SB\\_MsgId\\_t](#) values are equal.
  - [CFE\\_SB\\_MsgIdToValue](#) - Converts a [CFE\\_SB\\_MsgId\\_t](#) to a normal integer.
  - [CFE\\_SB\\_ValueToMsgId](#) - Converts a normal integer into a [CFE\\_SB\\_MsgId\\_t](#).

## 4.7 Table Services API

- [cFE Registration APIs](#)
  - [CFE\\_TBL\\_Register](#) - Register a table with cFE to obtain Table Management Services.
  - [CFE\\_TBL\\_Share](#) - Obtain handle of table registered by another application.
  - [CFE\\_TBL\\_Unregister](#) - Unregister a table.
- [cFE Manage Table Content APIs](#)
  - [CFE\\_TBL\\_Load](#) - Load a specified table with data from specified source.
  - [CFE\\_TBL\\_Update](#) - Update contents of a specified table, if an update is pending.
  - [CFE\\_TBL\\_Validate](#) - Perform steps to validate the contents of a table image.
  - [CFE\\_TBL\\_Manage](#) - Perform standard operations to maintain a table.
  - [CFE\\_TBL\\_DumpToBuffer](#) - Copies the contents of a Dump Only Table to a shared buffer.
  - [CFE\\_TBL\\_Modified](#) - Notify cFE Table Services that table contents have been modified by the Application.
- [cFE Access Table Content APIs](#)

- [CFE\\_TBL\\_GetAddress](#) - Obtain the current address of the contents of the specified table.
- [CFE\\_TBL\\_GetAddresses](#) - Obtain the current addresses of an array of specified tables.
- [CFE\\_TBL\\_ReleaseAddress](#) - Release previously obtained pointer to the contents of the specified table.
- [CFE\\_TBL\\_ReleaseAddresses](#) - Release the addresses of an array of specified tables.
- cFE Get Table Information APIs
  - [CFE\\_TBL\\_GetStatus](#) - Obtain current status of pending actions for a table.
  - [CFE\\_TBL\\_GetInfo](#) - Obtain characteristics/information of/about a specified table.
  - [CFE\\_TBL\\_NotifyByMessage](#) - Instruct cFE Table Services to notify Application via message when table requires management.

## 4.8 Time Services API

- cFE Get Current Time APIs
  - [CFE\\_TIME\\_GetTime](#) - Get the current spacecraft time.
  - [CFE\\_TIME\\_GetTAI](#) - Get the current TAI (MET + SCTF) time.
  - [CFE\\_TIME\\_GetUTC](#) - Get the current UTC (MET + SCTF - Leap Seconds) time.
  - [CFE\\_TIME\\_GetMET](#) - Get the current value of the Mission Elapsed Time (MET).
  - [CFE\\_TIME\\_GetMETseconds](#) - Get the current seconds count of the mission-elapsed time.
  - [CFE\\_TIME\\_GetMETsubsecs](#) - Get the current sub-seconds count of the mission-elapsed time.
- cFE Get Time Information APIs
  - [CFE\\_TIME\\_GetSTCF](#) - Get the current value of the spacecraft time correction factor (STCF).
  - [CFE\\_TIME\\_GetLeapSeconds](#) - Get the current value of the leap seconds counter.
  - [CFE\\_TIME\\_GetClockState](#) - Get the current state of the spacecraft clock.
  - [CFE\\_TIME\\_GetClockInfo](#) - Provides information about the spacecraft clock.
- cFE Time Arithmetic APIs
  - [CFE\\_TIME\\_Add](#) - Adds two time values.
  - [CFE\\_TIME\\_Subtract](#) - Subtracts two time values.
  - [CFE\\_TIME\\_Compare](#) - Compares two time values.
- cFE Time Conversion APIs
  - [CFE\\_TIME\\_MET2SCTime](#) - Convert specified MET into Spacecraft Time.
  - [CFE\\_TIME\\_Sub2MicroSecs](#) - Converts a sub-seconds count to an equivalent number of microseconds.
  - [CFE\\_TIME\\_Micro2SubSecs](#) - Converts a number of microseconds to an equivalent sub-seconds count.
- cFE External Time Source APIs
  - [CFE\\_TIME\\_ExternalTone](#) - Provides the 1 Hz signal from an external source.
  - [CFE\\_TIME\\_ExternalMET](#) - Provides the Mission Elapsed Time from an external source.
  - [CFE\\_TIME\\_ExternalGPS](#) - Provide the time from an external source that has data common to GPS receivers.
  - [CFE\\_TIME\\_ExternalTime](#) - Provide the time from an external source that measures time relative to a known epoch.
  - [CFE\\_TIME\\_RegisterSyncCallback](#) - Registers a callback function that is called whenever time synchronization occurs.

- [CFE\\_TIME\\_UnregisterSynchCallback](#) - Unregisters a callback function that is called whenever time synchronization occurs.
- cFE Miscellaneous Time APIs
  - [CFE\\_TIME\\_Print](#) - Print a time value as a string.
  - [CFE\\_TIME\\_Local1HzISR](#) - This function is called via a timer callback set up at initialization of the TIME service.

## 5 Osal API Documentation

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

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

## 5.1 OSAL Introduction

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

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

### Note

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

## 5.2 File System Overview

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

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

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

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

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

## 5.3 File Descriptors In Osal

The OSAL uses abstracted file descriptors. This means that the file descriptors passed back from the OS\_open and OS\_creat calls will only work with other OSAL OS\_\* calls. The reasoning for this is as follows:

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

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

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

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

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

## 5.4 Timer Overview

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

# 6 cFE Mission Configuration Parameters

## Global [CFE\\_MISSION\\_ES\\_CMD\\_MSG](#)

cFE Portable Message Numbers for Commands

## 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\\_EVS\\_HK\\_TLM\\_MSG](#)

cFE Portable Message Numbers for Telemetry

## Global [CFE\\_MISSION\\_MAX\\_API\\_LEN](#)

cFE Maximum length for API names within data exchange structures

cFE Maximum length for API names within data exchange structures

## Global [CFE\\_MISSION\\_MAX\\_FILE\\_LEN](#)

cFE Maximum length for filenames within data exchange structures

cFE Maximum length for filenames within data exchange structures

## Global [CFE\\_MISSION\\_MAX\\_NUM\\_FILES](#)

cFE Maximum number of files in a message/data exchange

cFE Maximum number of files in a message/data exchange

## Global [CFE\\_MISSION\\_MAX\\_PATH\\_LEN](#)

cFE Maximum length for pathnames within data exchange structures

cFE Maximum length for pathnames within data exchange structures

## Global [CFE\\_MISSION\\_SB\\_CMD\\_MSG](#)

cFE Portable Message Numbers for Commands

## Global [CFE\\_MISSION\\_SB\\_HK\\_TLM\\_MSG](#)

cFE Portable Message Numbers for Telemetry

## Global [CFE\\_MISSION\\_TBL\\_CMD\\_MSG](#)

cFE Portable Message Numbers for Commands

**Global CFE\_MISSION\_TBL\_HK\_TLM\_MSG**

cFE Portable Message Numbers for Telemetry

**Global CFE\_MISSION\_TIME\_CMD\_MSG**

cFE Portable Message Numbers for Commands

**Global CFE\_MISSION\_TIME\_DATA\_CMD\_MSG**

cFE Portable Message Numbers for Global Messages

**Global CFE\_MISSION\_TIME\_HK\_TLM\_MSG**

cFE Portable Message Numbers for Telemetry

## 7 Module Index

### 7.1 Modules

Here is a list of all modules:

<b>CFS Checksum Event IDs</b>	<b>125</b>
<b>CFS Checksum Mission Configuration</b>	<b>172</b>
<b>CFS Checksum Telemetry</b>	<b>173</b>
<b>CFS Checksum Command Structures</b>	<b>174</b>
<b>CFS Checksum Command Codes</b>	<b>175</b>
<b>CFS Checksum Command Message IDs</b>	<b>208</b>
<b>CFS Checksum Telemetry Message IDs</b>	<b>209</b>
<b>CFS Checksum Platform Configuration</b>	<b>210</b>
<b>CFS Checksum Version</b>	<b>217</b>
<b>cFE Return Code Defines</b>	<b>218</b>
<b>cFE Resource ID APIs</b>	<b>241</b>
<b>cFE Entry/Exit APIs</b>	<b>244</b>
<b>cFE Application Control APIs</b>	<b>246</b>
<b>cFE Application Behavior APIs</b>	<b>249</b>
<b>cFE Information APIs</b>	<b>253</b>
<b>cFE Child Task APIs</b>	<b>262</b>
<b>cFE Miscellaneous APIs</b>	<b>267</b>
<b>cFE Critical Data Store APIs</b>	<b>270</b>
<b>cFE Memory Manager APIs</b>	<b>275</b>
<b>cFE Performance Monitor APIs</b>	<b>282</b>

cFE Generic Counter APIs	284
cFE Registration APIs	290
cFE Send Event APIs	292
cFE Reset Event Filter APIs	297
cFE File Header Management APIs	299
cFE File Utility APIs	303
cFE Generic Message APIs	308
cFE Message Primary Header APIs	310
cFE Message Extended Header APIs	319
cFE Message Secondary Header APIs	325
cFE Message Id APIs	330
cFE Message Checking APIs	332
cFE Pipe Management APIs	333
cFE Message Subscription Control APIs	338
cFE Send/Receive Message APIs	343
cFE Zero Copy APIs	346
cFE Message Characteristics APIs	349
cFE Message ID APIs	354
cFE SB Pipe options	356
cFE Registration APIs	357
cFE Manage Table Content APIs	362
cFE Access Table Content APIs	368
cFE Get Table Information APIs	373
cFE Table Type Defines	376
cFE Get Current Time APIs	378
cFE Get Time Information APIs	381
cFE Time Arithmetic APIs	384
cFE Time Conversion APIs	387
cFE External Time Source APIs	389
cFE Miscellaneous Time APIs	394

cFE Resource ID base values	397
cFE Clock State Flag Defines	399
OSAL Semaphore State Defines	401
OSAL Binary Semaphore APIs	402
OSAL BSP low level access APIs	407
OSAL Real Time Clock APIs	408
OSAL Core Operation APIs	419
OSAL Condition Variable APIs	422
OSAL Counting Semaphore APIs	428
OSAL Directory APIs	433
OSAL Return Code Defines	437
OSAL Error Info APIs	444
OSAL File Access Option Defines	446
OSAL Reference Point For Seek Offset Defines	447
OSAL Standard File APIs	448
OSAL File System Level APIs	459
OSAL Heap APIs	467
OSAL Object Type Defines	468
OSAL Object ID Utility APIs	471
OSAL Dynamic Loader and Symbol APIs	476
OSAL Mutex APIs	480
OSAL Network ID APIs	484
OSAL Printf APIs	486
OSAL Message Queue APIs	487
OSAL Select APIs	491
OSAL Shell APIs	495
OSAL Socket Address APIs	496
OSAL Socket Management APIs	500
OSAL Task APIs	508
OSAL Time Base APIs	514

OSAL Timer APIs	519
-----------------	-----

## 8 Data Structure Index

### 8.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#"><b>CCSDS_ExtendedHeader</b></a> CCSDS packet extended header	525
<a href="#"><b>CCSDS_PrimaryHeader</b></a> CCSDS packet primary header	525
<a href="#"><b>CFE_ES_AppInfo</b></a> Application Information	526
<a href="#"><b>CFE_ES_AppNameCmd</b></a> Generic application name command	530
<a href="#"><b>CFE_ES_AppNameCmd_Payload</b></a> Generic application name command payload	531
<a href="#"><b>CFE_ES_AppReloadCmd_Payload</b></a> Reload Application Command Payload	531
<a href="#"><b>CFE_ES_BlockStats</b></a> Block statistics	532
<a href="#"><b>CFE_ES_CDSRegDumpRec</b></a> CDS Register Dump Record	533
<a href="#"><b>CFE_ES_DeleteCDSCmd</b></a> Delete Critical Data Store Command	534
<a href="#"><b>CFE_ES_DeleteCDSCmd_Payload</b></a> Delete Critical Data Store Command Payload	534
<a href="#"><b>CFE_ES_DumpCDSRegistryCmd</b></a> Dump CDS Registry Command	535
<a href="#"><b>CFE_ES_DumpCDSRegistryCmd_Payload</b></a> Dump CDS Registry Command Payload	535
<a href="#"><b>CFE_ES_FileNameCmd</b></a> Generic file name command	536
<a href="#"><b>CFE_ES_FileNameCmd_Payload</b></a> Generic file name command payload	536
<a href="#"><b>CFE_ES_HousekeepingTIm</b></a>	537
<a href="#"><b>CFE_ES_HousekeepingTIm_Payload</b></a>	538
<a href="#"><b>CFE_ES_MemPoolStats</b></a> Memory Pool Statistics	546

<b>CFE_ES_MemStatsTlm</b>	547
<b>CFE_ES_NoArgsCmd</b>	
Generic "no arguments" command	547
<b>CFE_ES_OneAppTlm</b>	548
<b>CFE_ES_OneAppTlm_Payload</b>	549
<b>CFE_ES_OverWriteSysLogCmd</b>	
Overwrite/Discard System Log Configuration Command Payload	549
<b>CFE_ES_OverWriteSysLogCmd_Payload</b>	
Overwrite/Discard System Log Configuration Command Payload	550
<b>CFE_ES_PoolAlign</b>	
Pool Alignment	550
<b>CFE_ES_PoolStatsTlm_Payload</b>	551
<b>CFE_ES_ReloadAppCmd</b>	
Reload Application Command	552
<b>CFE_ES_RestartCmd</b>	
Restart cFE Command	552
<b>CFE_ES_RestartCmd_Payload</b>	
Restart cFE Command Payload	553
<b>CFE_ES_SendMemPoolStatsCmd</b>	
Send Memory Pool Statistics Command	553
<b>CFE_ES_SendMemPoolStatsCmd_Payload</b>	
Send Memory Pool Statistics Command Payload	554
<b>CFE_ES_SetMaxPRCountCmd</b>	
Set Maximum Processor Reset Count Command	555
<b>CFE_ES_SetMaxPRCountCmd_Payload</b>	
Set Maximum Processor Reset Count Command Payload	555
<b>CFE_ES_SetPerfFilterMaskCmd</b>	
Set Performance Analyzer Filter Mask Command	556
<b>CFE_ES_SetPerfFilterMaskCmd_Payload</b>	
Set Performance Analyzer Filter Mask Command Payload	556
<b>CFE_ES_SetPerfTriggerMaskCmd</b>	
Set Performance Analyzer Trigger Mask Command	557
<b>CFE_ES_SetPerfTrigMaskCmd_Payload</b>	
Set Performance Analyzer Trigger Mask Command Payload	558
<b>CFE_ES_StartApp</b>	
Start Application Command	558

<b>CFE_ES_StartAppCmd_Payload</b> Start Application Command Payload	559
<b>CFE_ES_StartPerfCmd_Payload</b> Start Performance Analyzer Command Payload	560
<b>CFE_ES_StartPerfDataCmd</b> Start Performance Analyzer Command	560
<b>CFE_ES_StopPerfCmd_Payload</b> Stop Performance Analyzer Command Payload	561
<b>CFE_ES_StopPerfDataCmd</b> Stop Performance Analyzer Command	562
<b>CFE_ES_TaskInfo</b> Task Information	562
<b>CFE_EVS_AppDataCmd_Payload</b> Write Event Services Application Information to File Command Payload	564
<b>CFE_EVS_AppNameBitMaskCmd</b> Generic App Name and Bitmask Command	564
<b>CFE_EVS_AppNameBitMaskCmd_Payload</b> Generic App Name and Bitmask Command Payload	565
<b>CFE_EVS_AppNameCmd</b> Generic App Name Command	566
<b>CFE_EVS_AppNameCmd_Payload</b> Generic App Name Command Payload	566
<b>CFE_EVS_AppNameEventIDCmd</b> Generic App Name and Event ID Command	567
<b>CFE_EVS_AppNameEventIDCmd_Payload</b> Generic App Name and Event ID Command Payload	567
<b>CFE_EVS_AppNameEventIDMaskCmd</b> Generic App Name, Event ID, Mask Command	568
<b>CFE_EVS_AppNameEventIDMaskCmd_Payload</b> Generic App Name, Event ID, Mask Command Payload	569
<b>CFE_EVS_AppTlmData</b>	569
<b>CFE_EVS_BinFilter</b> Event message filter definition structure	570
<b>CFE_EVS_BitMaskCmd</b> Generic Bitmask Command	571
<b>CFE_EVS_BitMaskCmd_Payload</b> Generic Bitmask Command Payload	572
<b>CFE_EVS_HousekeepingTlm</b>	572

<b>CFE_EVS_HousekeepingTlm_Payload</b>	573
<b>CFE_EVS_LogFileCmd_Payload</b>	
Write Event Log to File Command Payload	576
<b>CFE_EVS_LongEventTlm</b>	576
<b>CFE_EVS_LongEventTlm_Payload</b>	577
<b>CFE_EVS_NoArgsCmd</b>	
Command with no additional arguments	578
<b>CFE_EVS_PacketID</b>	578
<b>CFE_EVS_SetEventFormatCode_Payload</b>	
Set Event Format Mode Command Payload	580
<b>CFE_EVS_SetEventFormatModeCmd</b>	
Set Event Format Mode Command	580
<b>CFE_EVS_SetLogMode_Payload</b>	
Set Log Mode Command Payload	581
<b>CFE_EVS_SetLogModeCmd</b>	
Set Log Mode Command	582
<b>CFE_EVS_ShortEventTlm</b>	582
<b>CFE_EVS_ShortEventTlm_Payload</b>	583
<b>CFE_EVS_WriteAppDataFileCmd</b>	
Write Event Services Application Information to File Command	583
<b>CFE_EVS_WriteLogFileCmd</b>	
Write Event Log to File Command	584
<b>CFE_FS_FileWriteMetaData</b>	
External Metadata/State object associated with background file writes	584
<b>CFE_FS_Header</b>	
Standard cFE File header structure definition	586
<b>CFE_SB_AllSubscriptionsTlm</b>	587
<b>CFE_SB_AllSubscriptionsTlm_Payload</b>	588
<b>CFE_SB_HousekeepingTlm</b>	589
<b>CFE_SB_HousekeepingTlm_Payload</b>	589
<b>CFE_SB_Msg</b>	
Software Bus generic message	593
<b>CFE_SB_MsgId_t</b>	
CFE_SB_MsgId_t type definition	594
<b>CFE_SB_MsgMapFileEntry</b>	
SB Map File Entry	594

<b>CFE_SB_PipeDepthStats</b>	SB Pipe Depth Statistics	595
<b>CFE_SB_PipeInfoEntry</b>	SB Pipe Information File Entry	596
<b>CFE_SB_Qos_t</b>	Quality Of Service Type Definition	598
<b>CFE_SB_RouteCmd</b>	Enable/Disable Route Command	598
<b>CFE_SB_RouteCmd_Payload</b>	Enable/Disable Route Command Payload	599
<b>CFE_SB_RoutingFileEntry</b>	SB Routing File Entry	600
<b>CFE_SB_SingleSubscriptionTlm</b>		601
<b>CFE_SB_SingleSubscriptionTlm_Payload</b>		602
<b>CFE_SB_StatsTlm</b>		603
<b>CFE_SB_StatsTlm_Payload</b>		603
<b>CFE_SB_SubEntries</b>	SB Previous Subscriptions Entry	607
<b>CFE_SB_WriteFileInfoCmd</b>	Write File Info Command	607
<b>CFE_SB_WriteFileInfoCmd_Payload</b>	Write File Info Command Payload	608
<b>CFE_TBL_AbortLoadCmd</b>	Abort Load Command	608
<b>CFE_TBL_AbortLoadCmd_Payload</b>	Abort Load Command Payload	609
<b>CFE_TBL_ActivateCmd</b>	Activate Table Command	610
<b>CFE_TBL_ActivateCmd_Payload</b>	Activate Table Command Payload	610
<b>CFE_TBL_DeleteCDSCmd_Payload</b>	Delete Critical Table CDS Command Payload	611
<b>CFE_TBL_DeleteCDSCmd</b>	Delete Critical Table CDS Command	611
<b>CFE_TBL_DumpCmd</b>		612
<b>CFE_TBL_DumpCmd_Payload</b>	Dump Table Command Payload	612

<b>CFE_TBL_DumpRegistryCmd</b>		
Dump Registry Command		613
<b>CFE_TBL_DumpRegistryCmd_Payload</b>		
Dump Registry Command Payload		614
<b>CFE_TBL_File_Hdr</b>		
The definition of the header fields that are included in CFE Table Data files		614
<b>CFE_TBL_FileDef</b>		
Table File summary object		615
<b>CFE_TBL_HousekeepingTlm</b>		617
<b>CFE_TBL_HousekeepingTlm_Payload</b>		617
<b>CFE_TBL_Info</b>		
Table Info		621
<b>CFE_TBL_LoadCmd</b>		
Load Table Command		623
<b>CFE_TBL_LoadCmd_Payload</b>		
Load Table Command Payload		624
<b>CFE_TBL_NoArgsCmd</b>		
Generic "no arguments" command		625
<b>CFE_TBL_NotifyCmd</b>		625
<b>CFE_TBL_NotifyCmd_Payload</b>		
Table Management Notification Command Payload		626
<b>CFE_TBL_SendRegistryCmd</b>		
Send Table Registry Command		626
<b>CFE_TBL_SendRegistryCmd_Payload</b>		
Send Table Registry Command Payload		627
<b>CFE_TBL_TableRegistryTlm</b>		627
<b>CFE_TBL_TblRegPacket_Payload</b>		628
<b>CFE_TBL_ValidateCmd</b>		
Validate Table Command		632
<b>CFE_TBL_ValidateCmd_Payload</b>		
Validate Table Command Payload		632
<b>CFE_TIME_DiagnosticTlm</b>		633
<b>CFE_TIME_DiagnosticTlm_Payload</b>		633
<b>CFE_TIME_HousekeepingTlm</b>		642
<b>CFE_TIME_HousekeepingTlm_Payload</b>		642

<b>CFE_TIME_LeapsCmd_Payload</b>	Set leap seconds command payload	645
<b>CFE_TIME_NoArgsCmd</b>	Generic no argument command	645
<b>CFE_TIME_OneHzAdjustmentCmd</b>	Generic seconds, subseconds adjustment command	646
<b>CFE_TIME_OneHzAdjustmentCmd_Payload</b>	Generic seconds, subseconds command payload	646
<b>CFE_TIME_SetLeapSecondsCmd</b>	Set leap seconds command	647
<b>CFE_TIME_SetSignalCmd</b>	Set tone signal source command	648
<b>CFE_TIME_SetSourceCmd</b>	Set time data source command	648
<b>CFE_TIME_SetStateCmd</b>	Set clock state command	649
<b>CFE_TIME_SignalCmd_Payload</b>	Set tone signal source command payload	649
<b>CFE_TIME_SourceCmd_Payload</b>	Set time data source command payload	650
<b>CFE_TIME_StateCmd_Payload</b>	Set clock state command payload	651
<b>CFE_TIME_SysTime</b>	Data structure used to hold system time values	651
<b>CFE_TIME_TimeCmd</b>	Generic seconds, microseconds argument command	652
<b>CFE_TIME_TimeCmd_Payload</b>	Generic seconds, microseconds command payload	652
<b>CFE_TIME_ToneDataCmd</b>	Time at tone data command	653
<b>CFE_TIME_ToneDataCmd_Payload</b>	Time at tone data command payload	654
<b>CS_AppData_t</b>	CS global data structure	654
<b>CS_AppNameCmd_Payload_t</b>	Payload for commanding by app name	662
<b>CS_AppNameCmd_t</b>	Command type for commanding by app name	662

<a href="#"><b>CS_Def_App_Table_Entry_t</b></a>	Data structure for the App definition table	663
<a href="#"><b>CS_Def_EepromMemory_Table_Entry_t</b></a>	Data structure for the EEPROM or Memory definition table	664
<a href="#"><b>CS_Def_Tables_Table_Entry_t</b></a>	Data structure for the Tables definition table	665
<a href="#"><b>CS_EntryCmd_Payload_t</b></a>	Payload for commands using Memory or EEPROM tables	665
<a href="#"><b>CS_EntryCmd_t</b></a>	Command type for commands using Memory or EEPROM tables	666
<a href="#"><b>CS_GetEntryIDCmd_Payload_t</b></a>	Get entry ID command payload	667
<a href="#"><b>CS_GetEntryIDCmd_t</b></a>	Get entry ID command	667
<a href="#"><b>CS_HkPacket_Payload_t</b></a>	Housekeeping Payload Structure	668
<a href="#"><b>CS_HkPacket_t</b></a>	Housekeeping Packet Structure	673
<a href="#"><b>CS_NoArgsCmd_t</b></a>	No arguments command data type	674
<a href="#"><b>CS_OneShotCmd_Payload_t</b></a>	Payload for sending one shot calculation	675
<a href="#"><b>CS_OneShotCmd_t</b></a>	Command type for sending one shot calculation	675
<a href="#"><b>CS_Res_App_Table_Entry_t</b></a>	Data structure for the app result table	676
<a href="#"><b>CS_Res_EepromMemory_Table_Entry_t</b></a>	Data structure for the Eeporom or Memory results table	678
<a href="#"><b>CS_Res_Tables_Table_Entry_t</b></a>	Data structure for the Tables result table	679
<a href="#"><b>CS_TableNameCmd_Payload_t</b></a>	Payload for commanding by table name	682
<a href="#"><b>CS_TableNameCmd_t</b></a>	Command type for commanding by table name	682
<a href="#"><b>OS_bin_sem_prop_t</b></a>	OSAL binary semaphore properties	683
<a href="#"><b>OS_condvar_prop_t</b></a>	OSAL condition variable properties	683

<b>OS_count_sem_prop_t</b> OSAL counting semaphore properties	684
<b>os_dirent_t</b> Directory entry	685
<b>OS_FdSet</b> An abstract structure capable of holding several OSAL IDs	685
<b>OS_file_prop_t</b> OSAL file properties	686
<b>os_fsinfo_t</b> OSAL file system info	686
<b>os_fstat_t</b> File system status	687
<b>OS_heap_prop_t</b> OSAL heap properties	688
<b>OS_module_address_t</b> OSAL module address properties	689
<b>OS_module_prop_t</b> OSAL module properties	690
<b>OS_mut_sem_prop_t</b> OSAL mutex properties	690
<b>OS_queue_prop_t</b> OSAL queue properties	691
<b>OS_SockAddr_t</b> Encapsulates a generic network address	692
<b>OS_SockAddrData_t</b> Storage buffer for generic network address	692
<b>OS_socket_prop_t</b> Encapsulates socket properties	693
<b>OS_static_symbol_record_t</b> Associates a single symbol name with a memory address	694
<b>OS_statvfs_t</b>	694
<b>OS_task_prop_t</b> OSAL task properties	695
<b>OS_time_t</b> OSAL time interval structure	696
<b>OS_timebase_prop_t</b> Time base properties	696

<a href="#">OS_timer_prop_t</a>	
Timer properties	<a href="#">697</a>

## 9 File Index

### 9.1 File List

Here is a list of all files with brief descriptions:

<a href="#">apps/cs/fsw/inc/cs_events.h</a>	<a href="#">698</a>
<a href="#">apps/cs/fsw/inc/cs_mission_cfg.h</a>	<a href="#">705</a>
<a href="#">apps/cs/fsw/inc/cs_msg.h</a>	<a href="#">705</a>
<a href="#">apps/cs/fsw/inc/cs_msgdefs.h</a>	<a href="#">706</a>
<a href="#">apps/cs/fsw/inc/cs_msgids.h</a>	<a href="#">710</a>
<a href="#">apps/cs/fsw/inc/cs_perfids.h</a>	<a href="#">710</a>
<a href="#">apps/cs/fsw/inc/cs_platform_cfg.h</a>	<a href="#">710</a>
<a href="#">apps/cs/fsw/inc/cs_tbldefs.h</a>	<a href="#">712</a>
<a href="#">apps/cs/fsw/src/cs_app.c</a>	<a href="#">724</a>
<a href="#">apps/cs/fsw/src/cs_app.h</a>	<a href="#">736</a>
<a href="#">apps/cs/fsw/src/cs_app_cmds.c</a>	<a href="#">749</a>
<a href="#">apps/cs/fsw/src/cs_app_cmds.h</a>	<a href="#">755</a>
<a href="#">apps/cs/fsw/src/cs_cmds.c</a>	<a href="#">761</a>
<a href="#">apps/cs/fsw/src/cs_cmds.h</a>	<a href="#">776</a>
<a href="#">apps/cs/fsw/src/cs_compute.c</a>	<a href="#">791</a>
<a href="#">apps/cs/fsw/src/cs_compute.h</a>	<a href="#">799</a>
<a href="#">apps/cs/fsw/src/cs_eeprom_cmds.c</a>	<a href="#">807</a>
<a href="#">apps/cs/fsw/src/cs_eeprom_cmds.h</a>	<a href="#">814</a>
<a href="#">apps/cs/fsw/src/cs_init.c</a>	<a href="#">820</a>
<a href="#">apps/cs/fsw/src/cs_init.h</a>	<a href="#">824</a>
<a href="#">apps/cs/fsw/src/cs_memory_cmds.c</a>	<a href="#">828</a>
<a href="#">apps/cs/fsw/src/cs_memory_cmds.h</a>	<a href="#">835</a>
<a href="#">apps/cs/fsw/src/cs_table_cmds.c</a>	<a href="#">842</a>
<a href="#">apps/cs/fsw/src/cs_table_cmds.h</a>	<a href="#">848</a>
<a href="#">apps/cs/fsw/src/cs_table_processing.c</a>	<a href="#">854</a>

apps/cs/fsw/src/cs_utils.c	865
apps/cs/fsw/src/cs_utils.h	884
apps/cs/fsw/src/cs_verify.h	904
apps/cs/fsw/src/cs_version.h	905
apps/cs/fsw/tables/cs_apptbl.c	905
apps/cs/fsw/tables/cs_eepromtbl.c	906
apps/cs/fsw/tables/cs_memorytbl.c	906
apps/cs/fsw/tables/cs_tablestbl.c	907
build/osal_public_api/inc/osconfig.h	907
cfe/cmake/sample_defs/example_mission_cfg.h	913
cfe/cmake/sample_defs/example_platform_cfg.h	924
cfe/cmake/sample_defs/sample_perfids.h	968
cfe/modules/core_api/config/default_cfe_core_api_base_msgids.h	971
cfe/modules/core_api/config/default_cfe_core_api_interface_cfg.h	972
cfe/modules/core_api/config/default_cfe_mission_cfg.h	974
cfe/modules/core_api/config/default_cfe_msgids.h	974
cfe/modules/core_api/fsw/inc/cfe.h	974
cfe/modules/core_api/fsw/inc/cfe_config.h	975
cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h	978
cfe/modules/core_api/fsw/inc/cfe_endian.h	979
cfe/modules/core_api/fsw/inc/cfe_error.h	979
cfe/modules/core_api/fsw/inc/cfe_es.h	988
cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h	991
cfe/modules/core_api/fsw/inc/cfe_evs.h	996
cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h	998
cfe/modules/core_api/fsw/inc/cfe_fs.h	1000
cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h	1001
cfe/modules/core_api/fsw/inc/cfe_msg.h	1003
cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h	1005
cfe/modules/core_api/fsw/inc/cfe_resourceid.h	1010

<a href="#">cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h</a>	1015
<a href="#">cfe/modules/core_api/fsw/inc/cfe_sb.h</a>	1016
<a href="#">cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h</a>	1018
<a href="#">cfe/modules/core_api/fsw/inc/cfe_tbl.h</a>	1021
<a href="#">cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h</a>	1022
<a href="#">cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h</a>	1025
<a href="#">cfe/modules/core_api/fsw/inc/cfe_time.h</a>	1026
<a href="#">cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h</a>	1028
<a href="#">cfe/modules/core_api/fsw/inc/cfe_version.h</a>	1029
<a href="#">cfe/modules/es/config/default_cfe_es_extern_typedefs.h</a>	1031
<a href="#">cfe/modules/es/config/default_cfe_es_fcncodes.h</a>	1040
<a href="#">cfe/modules/es/config/default_cfe_es_interface_cfg.h</a>	1060
<a href="#">cfe/modules/es/config/default_cfe_es_internal_cfg.h</a>	1063
<a href="#">cfe/modules/es/config/default_cfe_es_mission_cfg.h</a>	1083
<a href="#">cfe/modules/es/config/default_cfe_es_msg.h</a>	1083
<a href="#">cfe/modules/es/config/default_cfe_es_msgdefs.h</a>	1084
<a href="#">cfe/modules/es/config/default_cfe_es_msgids.h</a>	1084
<a href="#">cfe/modules/es/config/default_cfe_es_msgstruct.h</a>	1085
<a href="#">cfe/modules/es/config/default_cfe_es_platform_cfg.h</a>	1092
<a href="#">cfe/modules/es/config/default_cfe_es_topicids.h</a>	1093
<a href="#">cfe/modules/es/fsw/inc/cfe_es_eventids.h</a>	1094
<a href="#">cfe/modules/evs/config/default_cfe_evs_extern_typedefs.h</a>	1119
<a href="#">cfe/modules/evs/config/default_cfe_evs_fcncodes.h</a>	1122
<a href="#">cfe/modules/evs/config/default_cfe_evs_interface_cfg.h</a>	1140
<a href="#">cfe/modules/evs/config/default_cfe_evs_internal_cfg.h</a>	1141
<a href="#">cfe/modules/evs/config/default_cfe_evs_mission_cfg.h</a>	1145
<a href="#">cfe/modules/evs/config/default_cfe_evs_msg.h</a>	1145
<a href="#">cfe/modules/evs/config/default_cfe_evs_msgdefs.h</a>	1145
<a href="#">cfe/modules/evs/config/default_cfe_evs_msgids.h</a>	1146
<a href="#">cfe/modules/evs/config/default_cfe_evs_msgstruct.h</a>	1147

cfe/modules/evs/config/default_cfe_evs_platform_cfg.h	1154
cfe/modules/evs/config/default_cfe_evs_topicids.h	1154
cfe/modules/evs/fsw/inc/cfe_evs_eventids.h	1155
cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h	1167
cfe/modules/fs/config/default_cfe_fs_filedef.h	1167
cfe/modules/fs/config/default_cfe_fs_interface_cfg.h	1169
cfe/modules/fs/config/default_cfe_fs_mission_cfg.h	1170
cfe/modules/msg/fsw/inc/ccsds_hdr.h	1170
cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h	1171
cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h	1171
cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h	1172
cfe/modules/sb/config/default_cfe_sb_fcncodes.h	1174
cfe/modules/sb/config/default_cfe_sb_interface_cfg.h	1184
cfe/modules/sb/config/default_cfe_sb_internal_cfg.h	1185
cfe/modules/sb/config/default_cfe_sb_mission_cfg.h	1193
cfe/modules/sb/config/default_cfe_sb_msg.h	1194
cfe/modules/sb/config/default_cfe_sb_msgdefs.h	1194
cfe/modules/sb/config/default_cfe_sb_msgids.h	1194
cfe/modules/sb/config/default_cfe_sb_msgstruct.h	1195
cfe/modules/sb/config/default_cfe_sb_platform_cfg.h	1200
cfe/modules/sb/config/default_cfe_sb_topicids.h	1200
cfe/modules/sb/fsw/inc/cfe_sb_eventids.h	1201
cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h	1220
cfe/modules/tbl/config/default_cfe_tbl_fcncodes.h	1222
cfe/modules/tbl/config/default_cfe_tbl_interface_cfg.h	1231
cfe/modules/tbl/config/default_cfe_tbl_internal_cfg.h	1232
cfe/modules/tbl/config/default_cfe_tbl_mission_cfg.h	1238
cfe/modules/tbl/config/default_cfe_tbl_msg.h	1238
cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h	1238
cfe/modules/tbl/config/default_cfe_tbl_msgids.h	1239

<a href="#">cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h</a>	1239
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_platform_cfg.h</a>	1244
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_topicids.h</a>	1244
<a href="#">cfe/modules/tbl/fsw/inc/cfe_tbl_eventids.h</a>	1245
<a href="#">cfe/modules/time/config/default_cfe_time_extern_typedefs.h</a>	1265
<a href="#">cfe/modules/time/config/default_cfe_time_fcncodes.h</a>	1270
<a href="#">cfe/modules/time/config/default_cfe_time_interface_cfg.h</a>	1285
<a href="#">cfe/modules/time/config/default_cfe_time_internal_cfg.h</a>	1290
<a href="#">cfe/modules/time/config/default_cfe_time_mission_cfg.h</a>	1295
<a href="#">cfe/modules/time/config/default_cfe_time_msg.h</a>	1295
<a href="#">cfe/modules/time/config/default_cfe_time_msgdefs.h</a>	1296
<a href="#">cfe/modules/time/config/default_cfe_time_msgids.h</a>	1297
<a href="#">cfe/modules/time/config/default_cfe_time_msgstruct.h</a>	1298
<a href="#">cfe/modules/time/config/default_cfe_time_platform_cfg.h</a>	1303
<a href="#">cfe/modules/time/config/default_cfe_time_topicids.h</a>	1303
<a href="#">cfe/modules/time/fsw/inc/cfe_time_eventids.h</a>	1305
<a href="#">osal/src/os/inc/common_types.h</a>	1315
<a href="#">osal/src/os/inc/osapi-binsem.h</a>	1320
<a href="#">osal/src/os/inc/osapi-bsp.h</a>	1321
<a href="#">osal/src/os/inc/osapi-clock.h</a>	1321
<a href="#">osal/src/os/inc/osapi-common.h</a>	1323
<a href="#">osal/src/os/inc/osapi-condvar.h</a>	1325
<a href="#">osal/src/os/inc/osapi-constants.h</a>	1326
<a href="#">osal/src/os/inc/osapi-countsem.h</a>	1327
<a href="#">osal/src/os/inc/osapi-dir.h</a>	1327
<a href="#">osal/src/os/inc/osapi-error.h</a>	1328
<a href="#">osal/src/os/inc/osapi-file.h</a>	1331
<a href="#">osal/src/os/inc/osapi-fs.h</a>	1334
<a href="#">osal/src/os/inc/osapi-heap.h</a>	1336
<a href="#">osal/src/os/inc/osapi-idmap.h</a>	1336

<a href="#">osal/src/os/inc/osapi-macros.h</a>	1338
<a href="#">osal/src/os/inc/osapi-module.h</a>	1339
<a href="#">osal/src/os/inc/osapi-mutex.h</a>	1341
<a href="#">osal/src/os/inc/osapi-network.h</a>	1341
<a href="#">osal/src/os/inc/osapi-printf.h</a>	1342
<a href="#">osal/src/os/inc/osapi-queue.h</a>	1342
<a href="#">osal/src/os/inc/osapi-select.h</a>	1343
<a href="#">osal/src/os/inc/osapi-shell.h</a>	1344
<a href="#">osal/src/os/inc/osapi-sockets.h</a>	1344
<a href="#">osal/src/os/inc/osapi-task.h</a>	1347
<a href="#">osal/src/os/inc/osapi-timebase.h</a>	1349
<a href="#">osal/src/os/inc/osapi-timer.h</a>	1350
<a href="#">osal/src/os/inc/osapi-version.h</a>	1351
<a href="#">osal/src/os/inc/osapi.h</a>	1354
<a href="#">psp/fsw/inc/cfe_psp.h</a>	1355
<a href="#">psp/fsw/inc/cfe_psp_error.h</a>	1368
CFE PSP Error header	

## 10 Module Documentation

### 10.1 CFS Checksum Event IDs

#### Macros

- `#define CS_INIT_INF_EID 1`  
*CS Initialization Event ID.*
- `#define CS_NOOP_INF_EID 2`  
*CS No-op Command Event ID.*
- `#define CS_RESET_DBG_EID 3`  
*CS Reset Counters Command Event ID.*
- `#define CS_DISABLE_ALL_INF_EID 4`  
*CS Disable Checksumming Command Event ID.*
- `#define CS_ENABLE_ALL_INF_EID 5`  
*CS Enable Checksumming Command Event ID.*
- `#define CS_DISABLE_CFECORE_INF_EID 6`  
*CS Disable cFE Core Checksumming Command Event ID.*
- `#define CS_ENABLE_CFECORE_INF_EID 7`  
*CS Enable cFE Core Checksumming Command Event ID.*
- `#define CS_DISABLE_OS_INF_EID 8`

- #define CS\_ENABLE\_OS\_INF\_EID 9  
*CS Disable OS Code Checksumming Command Event ID.*
- #define CS\_BASELINE\_CFECORE\_INF\_EID 10  
*CS Enable OS Code Checksumming Command Event ID.*
- #define CS\_NO\_BASELINE\_CFECORE\_INF\_EID 11  
*CS Baseline cFE Core Checksum Event ID.*
- #define CS\_BASELINE\_OS\_INF\_EID 12  
*CS Baseline OS Code Checksum Event ID.*
- #define CS\_NO\_BASELINE\_OS\_INF\_EID 13  
*CS Baseline OS Code Checksum Pending Event ID.*
- #define CS\_RECOMPUTE\_CFECORE\_STARTED\_DBG\_EID 14  
*CS cFE Core Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_CFECORE\_CREATE\_CHDTASK\_ERR\_EID 15  
*CS cFE Core Checksum Failed Child Task Create Event ID.*
- #define CS\_RECOMPUTE\_CFECORE\_CHDTASK\_ERR\_EID 16  
*CS cFE Core Checksum Failed Child Task In Use Event ID.*
- #define CS\_RECOMPUTE\_OS\_STARTED\_DBG\_EID 17  
*CS OS Code Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_OS\_CREATE\_CHDTASK\_ERR\_EID 18  
*CS OS Code Checksum Recompute Failed Child Task Create Event ID.*
- #define CS\_RECOMPUTE\_OS\_CHDTASK\_ERR\_EID 19  
*CS OS Code Checksum Recompute Failed Child Task In Use Event ID.*
- #define CS\_ONESHOT\_STARTED\_DBG\_EID 20  
*CS Oneshot Checksum Started Event ID.*
- #define CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID 21  
*CS Oneshot Checksum Failed Child Task Create Event ID.*
- #define CS\_ONESHOT\_CHDTASK\_ERR\_EID 22  
*CS Oneshot Checksum Failed Child Task In Use Event ID.*
- #define CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID 23  
*CS Oneshot Checksum Failed Invalid Memory Range Event ID.*
- #define CS\_ONESHOT\_CANCELLED\_INF\_EID 24  
*CS Oneshot Checksum Cancelled Event ID.*
- #define CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID 25  
*CS Oneshot Checksum Failed Child Task Delete Event ID.*
- #define CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID 26  
*CS Oneshot Checksum Cancel Failed No Oneshot Active Event ID.*
- #define CS\_EEPROM\_MISCOMPARE\_ERR\_EID 27  
*CS EEPROM Checksum Miscompare Event ID.*
- #define CS\_MEMORY\_MISCOMPARE\_ERR\_EID 28  
*CS Memory Checksum Miscompare Event ID.*
- #define CS\_TABLES\_MISCOMPARE\_ERR\_EID 29  
*CS Table Checksum Miscompare Event ID.*
- #define CS\_APP\_MISCOMPARE\_ERR\_EID 30  
*CS App Checksum Miscompare Event ID.*
- #define CS\_CFECORE\_MISCOMPARE\_ERR\_EID 31  
*CS cFE Core Checksum Miscompare Event ID.*

- #define CS\_OS\_MISCOMPARE\_ERR\_EID 32  
*CS OS Code Checksum Miscompare Event ID.*
- #define CS\_MID\_ERR\_EID 33  
*CS Invalid Command Pipe Message ID Event ID.*
- #define CS\_CC1\_ERR\_EID 34  
*CS Invalid Ground Command Code Event ID.*
- #define CS\_EXIT\_ERR\_EID 35  
*CS App Termination Event ID.*
- #define CS\_LEN\_ERR\_EID 36  
*CS Invalid Message Length Event ID.*
- #define CS\_DISABLE\_EEPROM\_INF\_EID 37  
*CS Disable EEPROM Checksumming Command Event ID.*
- #define CS\_ENABLE\_EEPROM\_INF\_EID 38  
*CS Enable EEPROM Checksumming Command Event ID.*
- #define CS\_BASELINE\_EEPROM\_INF\_EID 39  
*CS Baseline EEPROM Checksum Event ID.*
- #define CS\_NO\_BASELINE\_EEPROM\_INF\_EID 40  
*CS Baseline EEPROM Checksum Pending Event ID.*
- #define CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID 41  
*CS Baseline EEPROM Checksum Failed Invalid Entry ID Event ID.*
- #define CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID 42  
*CS EEPROM Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID 43  
*CS EEPROM Checksum Recompute Failed Child Task Create Event ID.*
- #define CS\_RECOMPUTE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID 44  
*CS EEPROM Checksum Recompute Failed Invalid Entry ID Event ID.*
- #define CS\_RECOMPUTE\_EEPROM\_CHDTASK\_ERR\_EID 45  
*CS EEPROM Checksum Recompute Failed Child Task In Use Event ID.*
- #define CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID 46  
*CS EEPROM Checksum Entry ID Enabled Event ID.*
- #define CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID 47  
*CS EEPROM Checksum Entry ID Enable Failed Invalid Entry ID Event ID.*
- #define CS\_DISABLE\_EEPROM\_ENTRY\_INF\_EID 48  
*CS EEPROM Checksum Entry ID Disabled Event ID.*
- #define CS\_DISABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID 49  
*CS EEPROM Checksum Entry ID Disable Failed Invalid Entry ID Event ID.*
- #define CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID 50  
*CS EEPROM Get Entry ID Event ID.*
- #define CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID 51  
*CS EEPROM Get Entry ID Address Not Found Event ID.*
- #define CS\_DISABLE\_MEMORY\_INF\_EID 52  
*CS Disable Memory Checksumming Command Event ID.*
- #define CS\_ENABLE\_MEMORY\_INF\_EID 53  
*CS Enable Memory Checksumming Command Event ID.*
- #define CS\_BASELINE\_MEMORY\_INF\_EID 54  
*CS Baseline Memory Checksum Event ID.*
- #define CS\_NO\_BASELINE\_MEMORY\_INF\_EID 55

- #define CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID 56
  - CS Baseline Memory Checksum Pending Event ID.
- #define CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID 57
  - CS Baseline Memory Checksum Failed Invalid Entry ID Event ID.
- #define CS\_RECOMPUTE\_MEMORY\_CREATE\_CHDTASK\_ERR\_EID 58
  - CS Memory Checksum Recompute Started Event ID.
- #define CS\_RECOMPUTE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID 59
  - CS Memory Checksum Recompute Failed Child Task Create Event ID.
- #define CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR\_EID 60
  - CS Memory Checksum Recompute Failed Invalid Entry ID Event ID.
- #define CS\_ENABLE\_MEMORY\_ENTRY\_INF\_EID 61
  - CS Memory Checksum Recompute Failed Child Task In Use Event ID.
- #define CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID 62
  - CS Memory Checksum Entry ID Enabled Event ID.
- #define CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID 63
  - CS Memory Checksum Entry ID Enable Failed Invalid Entry ID Event ID.
- #define CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID 64
  - CS Memory Checksum Entry ID Disabled Event ID.
- #define CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID 65
  - CS Memory Get Entry ID Event ID.
- #define CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID 66
  - CS Memory Get Entry ID Address Not Found Event ID.
- #define CS\_DISABLE\_TABLES\_INF\_EID 67
  - CS Disable Tables Checksumming Command Event ID.
- #define CS\_ENABLE\_TABLES\_INF\_EID 68
  - CS Enable Tables Checksumming Command Event ID.
- #define CS\_BASELINE\_TABLES\_INF\_EID 69
  - CS Baseline Table Checksum Event ID.
- #define CS\_NO\_BASELINE\_TABLES\_INF\_EID 70
  - CS Baseline Table Checksum Pending Event ID.
- #define CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID 71
  - CS Baseline Table Checksum Failed Table Not Found Event ID.
- #define CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID 72
  - CS Table Checksum Recompute Started Event ID.
- #define CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID 73
  - CS Table Checksum Recompute Failed Child Task Create Event ID.
- #define CS\_RECOMPUTE\_UNKNOWN\_NAME\_TABLES\_ERR\_EID 74
  - CS Table Checksum Recompute Failed Table Not Found Event ID.
- #define CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID 75
  - CS Table Checksum Recompute Failed Child Task In Use Event ID.
- #define CS\_ENABLE\_TABLES\_NAME\_INF\_EID 76
  - CS Table Checksum Enabled Event ID.
- #define CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID 77
  - CS Table Checksum Failed Table Not Found Event ID.
- #define CS\_DISABLE\_TABLES\_NAME\_INF\_EID 78
  - CS Table Checksum Disabled Event ID.

- #define CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID 79  
*CS Table Checksum Disable Failed Table Not Found Event ID.*
- #define CS\_DISABLE\_APP\_INF\_EID 80  
*CS Disable Apps Checksumming Command Event ID.*
- #define CS\_ENABLE\_APP\_INF\_EID 81  
*CS Enable Apps Checksumming Command Event ID.*
- #define CS\_BASELINE\_APP\_INF\_EID 82  
*CS Baseline App Checksum Event ID.*
- #define CS\_NO\_BASELINE\_APP\_INF\_EID 83  
*CS Baseline App Checksum Pending Event ID.*
- #define CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID 84  
*CS Baseline App Checksum Failed App Not Found Event ID.*
- #define CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID 85  
*CS App Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_APP\_CREATE\_CHDTASK\_ERR\_EID 86  
*CS App Checksum Recompute Failed Create Child Task Event ID.*
- #define CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID 87  
*CS App Checksum Recompute Failed App Not Found Event ID.*
- #define CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID 88  
*CS App Checksum Recompute Failed Child Task In Use Event ID.*
- #define CS\_ENABLE\_APP\_NAME\_INF\_EID 89  
*CS App Checksum Enabled Event ID.*
- #define CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID 90  
*CS App Checksum Enable Failed App Not Found Event ID.*
- #define CS\_DISABLE\_APP\_NAME\_INF\_EID 91  
*CS App Checksum Disabled Event ID.*
- #define CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID 92  
*CS App Checksum Disable Failed App Not Found Event ID.*
- #define CS\_COMPUTE\_APP\_NOT\_FOUND\_ERR\_EID 93  
*CS Apps Table Checksumming Failed App Not Found Event ID.*
- #define CS\_COMPUTE\_TABLES\_NOT\_FOUND\_ERR\_EID 94  
*CS Tables Table Checksumming Failed Table Not Found Event ID.*
- #define CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID 95  
*CS EEMROM or Memory Checksum Recompute Finished Event ID.*
- #define CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID 96  
*CS Table Checksum Recompute Failed Could Not Get Address Event ID.*
- #define CS\_RECOMPUTE\_ERROR\_APP\_ERR\_EID 97  
*CS App Checksum Recompute Failed Could Not Get Address Event ID.*
- #define CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID 98  
*CS Table Recompute Finished Event ID.*
- #define CS\_RECOMPUTE\_FINISH\_APP\_INF\_EID 99  
*CS App Recompute Finished Event ID.*
- #define CS\_ONESHOT\_FINISHED\_INF\_EID 100  
*CS Oneshot Checksum Finished Event ID.*
- #define CS\_VAL\_EEPROM\_STATE\_ERR\_EID 101  
*CS EEPROM Table Validate Failed Illegal State Field Event ID.*
- #define CS\_VAL\_EEPROM\_RANGE\_ERR\_EID 102

- #define CS\_VAL\_MEMORY\_STATE\_ERR\_EID 103
  - CS Memory Table Valide Failed Illegal State Field Event ID.
- #define CS\_VAL\_MEMORY\_RANGE\_ERR\_EID 104
  - CS Memory Table Validate Failed Illegal Checksum Range Event ID.
- #define CS\_VAL\_TABLES\_STATE\_ERR\_EID 105
  - CS Tables Table Validate Failed Illegal State Field Event ID.
- #define CS\_VAL\_APP\_STATE\_ERR\_EID 106
  - CS Apps Table Validate Failed Illegal State Field Event ID.
- #define CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID 107
  - CS EEPROM or Memory Table No Valid Entries Event ID.
- #define CS\_PROCESS\_APP\_NO\_ENTRIES\_INF\_EID 108
  - CS Apps Table No Valid Entries Event ID.
- #define CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID 109
  - CS Tables Table No Valid Entries Event ID.
- #define CS\_TBL\_INIT\_ERR\_EID 110
  - CS Table Initialization Failed Event ID.
- #define CS\_TBL\_UPDATE\_ERR\_EID 111
  - CS Table Update Failed Event ID.
- #define CS\_INIT\_SB\_CREATE\_ERR\_EID 112
  - CS Software Buse Create Pipe Failed Event ID.
- #define CS\_INIT\_SB\_SUBSCRIBE\_HK\_ERR\_EID 113
  - CS Software Bus Subscribe To Housekeeping Failed Event ID.
- #define CS\_INIT\_SB\_SUBSCRIBE\_BACK\_ERR\_EID 114
  - CS Software Bus Subscribe To Background Cycle Failed Event ID.
- #define CS\_INIT\_SB\_SUBSCRIBE\_CMD\_ERR\_EID 115
  - CS Software Bus Subscribe to Command Failed Event ID.
- #define CS\_INIT\_EEPROM\_ERR\_EID 116
  - CS EEPROM Table Initialization Failed Event ID.
- #define CS\_INIT\_MEMORY\_ERR\_EID 117
  - CS Memory Table Initialization Failed Event ID.
- #define CS\_INIT\_TABLES\_ERR\_EID 118
  - CS Tables Table Initialization Failed Event ID.
- #define CS\_INIT\_APP\_ERR\_EID 119
  - CS Apps Table Initialization Failed Event ID.
- #define CS\_COMPUTE\_TABLES\_RELEASE\_ERR\_EID 120
  - CS Table Release Address Failed Event ID.
- #define CS\_COMPUTE\_TABLES\_ERR\_EID 121
  - CS Get Table Info Failed Event ID.
- #define CS\_COMPUTE\_APP\_ERR\_EID 122
  - CS Get App Info Failed Event ID.
- #define CS\_UPDATE\_EEPROM\_ERR\_EID 123
  - CS EEPROM Table Update Failed Event ID.
- #define CS\_UPDATE\_MEMORY\_ERR\_EID 124
  - CS Memory Table Update Failed Event ID.
- #define CS\_UPDATE\_TABLES\_ERR\_EID 125
  - CS Tables Table Update Failed Event ID.

- #define CS\_UPDATE\_APP\_ERR\_EID 126  
*CS Apps Table Update Failed Event ID.*
- #define CS\_OS\_TEXT\_SEG\_INF\_EID 127  
*CS OS Code Checksum Disabled Platform Not Supported Event ID.*
- #define CS\_COMPUTE\_APP\_PLATFORM\_DBG\_EID 128  
*CS App Checksum Failed Platform Not Supported Event ID.*
- #define CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID 129  
*CS Table Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID 130  
*CS Table Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID 131  
*CS App Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID 132  
*CS App Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID 133  
*CS Memory Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_ENABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID 134  
*CS Memory Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID 135  
*CS EEPROM Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID 136  
*CS EEPROM Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_VAL\_TABLES\_DEF\_TBL\_DUPL\_ERR\_EID 137  
*CS Tables Table Validate Failed Duplicate Name Event ID.*
- #define CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID 138  
*CS Tables Table Validate Failed Illegal State With Empty Name Event ID.*
- #define CS\_VAL\_TABLES\_INF\_EID 139  
*CS Tables Table Verification Results Event ID.*
- #define CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID 140  
*CS Apps Table Validate Failed Duplicate Name Event ID.*
- #define CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID 141  
*CS Apps Table Validate Failed Illegal State With Empty Name Event ID.*
- #define CS\_VAL\_APP\_INF\_EID 142  
*CS Apps Table Verification Results Event ID.*
- #define CS\_VAL\_MEMORY\_INF\_EID 143  
*CS Memory Table Verification Results Event ID.*
- #define CS\_VAL\_EEPROM\_INF\_EID 144  
*CS EEPROM Table Verification Results Event ID.*
- #define CS\_CR\_CDS\_REG\_ERR\_EID 145  
*CS Critical Data Store Registration Failed Event ID.*
- #define CS\_CR\_CDS\_CPY\_ERR\_EID 146  
*CS Critical Data Store Copy Failed Event ID.*
- #define CS\_CR\_CDS\_RES\_ERR\_EID 147  
*CS Critical Data Store Restore Failed Event ID.*
- #define CS\_UPDATE\_CDS\_ERR\_EID 148  
*CS Critical Data Store Update Failed Event ID.*
- #define CS\_EXIT\_INF\_EID 149

*CS App Termination Event ID.*

- #define CS\_CFE\_TEXT\_SEG\_INF\_EID 150  
*CS cFE Core Checksum Disabled Address Not Available Event ID.*
- #define CS\_CMD\_COMPUTE\_PROG\_ERR\_EID 152  
*CS Command Failed Due To Recompute Or Oneshot In Progress Event ID.*
- #define CS\_BKGND\_COMPUTE\_PROG\_INF\_EID 153  
*CS Skipping Background Cycle Due To Recompute Or Oneshot In Progress Event ID.*
- #define CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID 154  
*CS Apps Table Validate Failed Illegal State With Long Name Event ID.*

#### 10.1.1 Detailed Description

#### 10.1.2 Macro Definition Documentation

**10.1.2.1 CS\_APP\_MISCOMPARE\_ERR\_EID** #define CS\_APP\_MISCOMPARE\_ERR\_EID 30  
CS App Checksum Miscompare Event ID.

Type: ERROR

Cause:

This event message is issued when a checksum miscompare occurs when checksumming entries in the table of applications to checksum.

Definition at line 406 of file cs\_events.h.

**10.1.2.2 CS\_BASELINE\_APP\_INF\_EID** #define CS\_BASELINE\_APP\_INF\_EID 82  
CS Baseline App Checksum Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the App entry specified command has been received and there is a baseline computed to report.

Definition at line 1091 of file cs\_events.h.

**10.1.2.3 CS\_BASELINE\_CFECORE\_INF\_EID** #define CS\_BASELINE\_CFECORE\_INF\_EID 10  
CS Baseline cFE Core Checksum Event ID.

Type: INFORMATION

**Cause:**

This event message is issued when a report baseline for the cFE core command has been received and there is a baseline computed to report.

Definition at line 148 of file cs\_events.h.

**10.1.2.4 CS\_BASELINE\_EEPROM\_INF\_EID** #define CS\_BASELINE\_EEPROM\_INF\_EID 39  
CS Baseline EEPROM Checksum Event ID.

Type: INFORMATION

**Cause:**

This event message is issued when a report baseline for the EEPROM entry specified command has been received and there is a baseline computed to report.

Definition at line 519 of file cs\_events.h.

**10.1.2.5 CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID** #define CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID 41  
CS Baseline EEPROM Checksum Failed Invalid Entry ID Event ID.

Type: ERROR

**Cause:**

This event message is issued when a report baseline for the command specified entry command has been received but specified Entry ID is invalid.

Definition at line 545 of file cs\_events.h.

**10.1.2.6 CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID** #define CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID 56  
CS Baseline Memory Checksum Failed Invalid Entry ID Event ID.

Type: ERROR

**Cause:**

This event message is issued when a report baseline for the command specified entry command has been received but specified Entry ID is invalid.

Definition at line 744 of file cs\_events.h.

**10.1.2.7 CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID** #define CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID 84

CS Baseline App Checksum Failed App Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when a report baseline for the command specified app command has been received but specified app name cannot be found or is marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1118 of file cs\_events.h.

**10.1.2.8 CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID** #define CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID 71

CS Baseline Table Checksum Failed Table Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when a report baseline for the command specified table command has been received but specified table name cannot be found or is marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 945 of file cs\_events.h.

**10.1.2.9 CS\_BASELINE\_MEMORY\_INF\_EID** #define CS\_BASELINE\_MEMORY\_INF\_EID 54

CS Baseline Memory Checksum Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the Memory entry specified command has been received and there is a baseline computed to report.

Definition at line 718 of file cs\_events.h.

**10.1.2.10 CS\_BASELINE\_OS\_INF\_EID** #define CS\_BASELINE\_OS\_INF\_EID 12

CS Baseline OS Code Checksum Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the OS code segment command has been received and there is a baseline computed to report.

Definition at line 174 of file cs\_events.h.

**10.1.2.11 CS\_BASELINE\_TABLES\_INF\_EID** #define CS\_BASELINE\_TABLES\_INF\_EID 69  
CS Baseline Table Checksum Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the Tables entry specified command has been received and there is a baseline computed to report.

Definition at line 918 of file cs\_events.h.

**10.1.2.12 CS\_BKGND\_COMPUTE\_PROG\_INF\_EID** #define CS\_BKGND\_COMPUTE\_PROG\_INF\_EID 153  
CS Skipping Background Cycle Due To Recompute Or Oneshote In Progress Event ID.

Type: INFORMATION

Cause:

This event message is issued when the background cycle is attempted while a recompute or oneshot is in progress.

Definition at line 1975 of file cs\_events.h.

**10.1.2.13 CS\_CC1\_ERR\_EID** #define CS\_CC1\_ERR\_EID 34  
CS Invalid Ground Command Code Event ID.

Type: ERROR

Cause:

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

Definition at line 454 of file cs\_events.h.

**10.1.2.14 CS\_CFE\_TEXT\_SEG\_INF\_EID** #define CS\_CFE\_TEXT\_SEG\_INF\_EID 150  
CS cFE Core Checksum Disabled Address Not Available Event ID.

Type: INFORMATION

Cause:

This event message is issued when CS is not able to get the address and size of the CFE Text Segment.

Definition at line 1950 of file cs\_events.h.

**10.1.2.15 CS\_CFECore\_Miscompare\_ERR\_EID** #define CS\_CFECore\_Miscompare\_ERR\_EID 31  
CS cFE Core Checksum Miscompare Event ID.

Type: ERROR

Cause:

This event message is issued when a checksum miscompare occurs when checksumming the cFE Core.  
Definition at line 418 of file cs\_events.h.

**10.1.2.16 CS\_Cmd\_Compute\_Prog\_ERR\_EID** #define CS\_Cmd\_Compute\_Prog\_ERR\_EID 152  
CS Command Failed Due To Recompute Or Oneshot In Progress Event ID.

Type: ERROR

Cause:

This event message is issued when certain commands are received while a recompute or oneshot is in progress.  
Affected commands are those that affect the checksum processing (such as enable/disable commands).  
Definition at line 1963 of file cs\_events.h.

**10.1.2.17 CS\_Compute\_App\_ERR\_EID** #define CS\_Compute\_App\_ERR\_EID 122  
CS Get App Info Failed Event ID.

Type: ERROR

Cause:

This event message is issued when there was a problem getting the app information to checksum.  
Definition at line 1585 of file cs\_events.h.

**10.1.2.18 CS\_Compute\_App\_Not\_Found\_ERR\_EID** #define CS\_Compute\_App\_Not\_Found\_ERR\_EID 93  
CS Apps Table Checksumming Failed App Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when an app cannot be found when checksumming.  
Definition at line 1238 of file cs\_events.h.

**10.1.2.19 CS\_COMPUTE\_APP\_PLATFORM\_DBG\_EID** #define CS\_COMPUTE\_APP\_PLATFORM\_DBG\_EID 128  
CS App Checksum Failed Platform Not Supported Event ID.

Type: DEBUG

Cause:

This event message is issued when CS is running on a platform that does not support getting the address information of the applications, such as OS X and Linux.

Definition at line 1655 of file cs\_events.h.

**10.1.2.20 CS\_COMPUTE\_TABLES\_ERR\_EID** #define CS\_COMPUTE\_TABLES\_ERR\_EID 121  
CS Get Table Info Failed Event ID.

Type: ERROR

Cause:

This event message is issued when there was a problem getting the table information to checksum.

Definition at line 1574 of file cs\_events.h.

**10.1.2.21 CS\_COMPUTE\_TABLES\_NOT\_FOUND\_ERR\_EID** #define CS\_COMPUTE\_TABLES\_NOT\_FOUND\_ERR\_EID 94  
CS Tables Table Checksumming Failed Table Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when a table cannot be found when checksumming.

Definition at line 1250 of file cs\_events.h.

**10.1.2.22 CS\_COMPUTE\_TABLES\_RELEASE\_ERR\_EID** #define CS\_COMPUTE\_TABLES\_RELEASE\_ERR\_EID 120  
CS Table Release Address Failed Event ID.

Type: ERROR

Cause:

This event message is issued when Table Services can't release the address for the table being checksummed.

Definition at line 1563 of file cs\_events.h.

**10.1.2.23 CS\_CR\_CDS\_CPY\_ERR\_EID** #define CS\_CR\_CDS\_CPY\_ERR\_EID 146  
CS Critical Data Store Copy Failed Event ID.

Type: ERROR

Cause:

The CS application optionally stores the table states in the Critical Data Store (CDS). This ensures that CS will not overwrite old data storage files following a processor reset. This event is issued if CS cannot copy to the CDS during startup initialization. Subsequent CDS errors are ignored by CS.

Definition at line 1897 of file cs\_events.h.

**10.1.2.24 CS\_CR\_CDS\_REG\_ERR\_EID** #define CS\_CR\_CDS\_REG\_ERR\_EID 145  
CS Critical Data Store Registration Failed Event ID.

Type: ERROR

Cause:

The CS application optionally stores the table states in the Critical Data Store (CDS). This ensures that CS will not overwrite old data storage files following a processor reset. This event is issued if CS cannot register the CDS during startup initialization. Subsequent CDS errors are ignored by CS.

Definition at line 1882 of file cs\_events.h.

**10.1.2.25 CS\_CR\_CDS\_RES\_ERR\_EID** #define CS\_CR\_CDS\_RES\_ERR\_EID 147  
CS Critical Data Store Restore Failed Event ID.

Type: ERROR

Cause:

The CS application optionally stores the table states in the Critical Data Store (CDS). This ensures that CS will not overwrite old data storage files following a processor reset. This event is issued if CS cannot restore from an existing CDS during startup initialization. Subsequent CDS errors are ignored by CS.

Definition at line 1912 of file cs\_events.h.

**10.1.2.26 CS\_DISABLE\_ALL\_INF\_EID** #define CS\_DISABLE\_ALL\_INF\_EID 4  
CS Disable Checksumming Command Event ID.

Type: INFORMATION

Cause:

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

Definition at line 75 of file cs\_events.h.

**10.1.2.27 CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID** #define CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID 132

CS App Disable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully disables an entry (specified by name) in the Apps results table but is unable to find the same entry in the definition table (or the entry is marked as [CS\\_STATE\\_EMPTY](#)).

Definition at line 1707 of file cs\_events.h.

**10.1.2.28 CS\_DISABLE\_APP\_INF\_EID** #define CS\_DISABLE\_APP\_INF\_EID 80

CS Disable Apps Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when disable checksumming for the App table command has been received.

Definition at line 1066 of file cs\_events.h.

**10.1.2.29 CS\_DISABLE\_APP\_NAME\_INF\_EID** #define CS\_DISABLE\_APP\_NAME\_INF\_EID 91

CS App Checksum Disabled Event ID.

Type: INFORMATION

Cause:

This event message is issued when a disable app name command is accepted.

Definition at line 1210 of file cs\_events.h.

**10.1.2.30 CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID** #define CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID 92

CS App Checksum Disable Failed App Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when a disable app name command is received, but has an unknown name or is marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1223 of file cs\_events.h.

**10.1.2.31 CS\_DISABLE\_CFECore\_INF\_EID** #define CS\_DISABLE\_CFECore\_INF\_EID 6  
CS Disable cFE Core Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when disable checksumming for the cFE core command has been received.  
Definition at line 99 of file cs\_events.h.

**10.1.2.32 CS\_DISABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID** #define CS\_DISABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID 135  
CS EEPROM Disable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully disables an entry (specified by name) in the EEPROM results table but identifies the corresponding entry in the definitions table to be set to [CS\\_STATE\\_EMPTY](#).  
Definition at line 1746 of file cs\_events.h.

**10.1.2.33 CS\_DISABLE\_EEPROM\_ENTRY\_INF\_EID** #define CS\_DISABLE\_EEPROM\_ENTRY\_INF\_EID 48  
CS EEPROM Checksum Entry ID Disabled Event ID.

Type: INFORMATION

Cause:

This event message is issued when a disable EEPROM Entry ID command is accepted.  
Definition at line 635 of file cs\_events.h.

**10.1.2.34 CS\_DISABLE\_EEPROM\_INF\_EID** #define CS\_DISABLE\_EEPROM\_INF\_EID 37  
CS Disable EEPROM Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when disable checksumming for the EEPROM table command has been received.  
Definition at line 494 of file cs\_events.h.

**10.1.2.35 CS\_DISABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID** #define CS\_DISABLE\_EEPROM\_INVALID\_ENTRY←  
\_ERR\_EID 49

CS EEPROM Checksum Entry ID Disable Failed Invalid Entry ID Event ID.

Type: ERROR

Cause:

This event message is issued when a disable EEPROM Entry ID command is received, but has an invalid Entry ID.  
Definition at line 647 of file cs\_events.h.

**10.1.2.36 CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID** #define CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID←  
ID 133

CS Memory Disable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully disables an entry (specified by name) in the Memory results table  
but identifies the corresponding entry in the definitions table to be set to [CS\\_STATE\\_EMPTY](#).

Definition at line 1720 of file cs\_events.h.

**10.1.2.37 CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID** #define CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID 63

CS Memory Checksum Entry ID Disabled Event ID.

Type: INFORMATION

Cause:

This event message is issued when a disable Memory Entry ID command is accepted.

Definition at line 835 of file cs\_events.h.

**10.1.2.38 CS\_DISABLE\_MEMORY\_INF\_EID** #define CS\_DISABLE\_MEMORY\_INF\_EID 52

CS Disable Memory Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when disable checksumming for the Memory table command has been received.

Definition at line 693 of file cs\_events.h.

**10.1.2.39 CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID** #define CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_EID 64

CS Memory Checksum Entry ID Disable Failed Invalid Entry ID Event ID.

Type: ERROR

Cause:

This event message is issued when a disable Memory Entry ID command is received, but has an invalid Entry ID.  
Definition at line 847 of file cs\_events.h.

**10.1.2.40 CS\_DISABLE\_OS\_INF\_EID** #define CS\_DISABLE\_OS\_INF\_EID 8

CS Disable OS Code Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when disable checksumming for the OS code segment command has been received.  
Definition at line 123 of file cs\_events.h.

**10.1.2.41 CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID** #define CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID 130

CS Table Disable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully disables an entry (specified by name) in the Tables results table but is unable to find the same entry in the definition table (or the entry is marked as [CS\\_STATE\\_EMPTY](#)).  
Definition at line 1681 of file cs\_events.h.

**10.1.2.42 CS\_DISABLE\_TABLES\_INF\_EID** #define CS\_DISABLE\_TABLES\_INF\_EID 67

CS Disable Tables Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when disable checksumming for the Tables table command has been received.  
Definition at line 893 of file cs\_events.h.

**10.1.2.43 CS\_DISABLE\_TABLES\_NAME\_INF\_EID** #define CS\_DISABLE\_TABLES\_NAME\_INF\_EID 78  
CS Table Checksum Disabled Event ID.

Type: INFORMATION

Cause:

This event message is issued when a disable table name command is accepted.  
Definition at line 1037 of file cs\_events.h.

**10.1.2.44 CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID** #define CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID 79  
CS Table Checksum Disable Failed Table Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when a disable name command is received, but has an unknown name or is marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1050 of file cs\_events.h.

**10.1.2.45 CS\_EEPROM\_MISCOMPARE\_ERR\_EID** #define CS\_EEPROM\_MISCOMPARE\_ERR\_EID 27  
CS EEPROM Checksum Miscompare Event ID.

Type: ERROR

Cause:

This event message is issued when a checksum miscompare occurs when checksumming entries in the EEPROM table.  
Definition at line 370 of file cs\_events.h.

**10.1.2.46 CS\_ENABLE\_ALL\_INF\_EID** #define CS\_ENABLE\_ALL\_INF\_EID 5  
CS Enable Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when enable checksumming command has been received.  
Definition at line 87 of file cs\_events.h.

**10.1.2.47 CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID** #define CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID 131

CS App Enable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully enables an entry (specified by name) in the Apps results table but is unable to find the same entry in the definition table (or the entry is marked as [CS\\_STATE\\_EMPTY](#)).

Definition at line 1694 of file cs\_events.h.

**10.1.2.48 CS\_ENABLE\_APP\_INF\_EID** #define CS\_ENABLE\_APP\_INF\_EID 81

CS Enable Apps Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when enable checksumming for the App table command has been received.

Definition at line 1078 of file cs\_events.h.

**10.1.2.49 CS\_ENABLE\_APP\_NAME\_INF\_EID** #define CS\_ENABLE\_APP\_NAME\_INF\_EID 89

CS App Checksum Enabled Event ID.

Type: INFORMATION

Cause:

This event message is issued when an enable app name command is accepted.

Definition at line 1185 of file cs\_events.h.

**10.1.2.50 CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID** #define CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID 90

CS App Checksum Enable Failed App Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when an enable app name command is received, but has an unknown name or is marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1198 of file cs\_events.h.

**10.1.2.51 CS\_ENABLE\_CFECore\_INF\_EID** #define CS\_ENABLE\_CFECore\_INF\_EID 7  
CS Enable cFE Core Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when enable checksumming for the cFE core command has been received.  
Definition at line 111 of file cs\_events.h.

**10.1.2.52 CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID** #define CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID 136  
CS EEPROM Enable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully enables an entry (specified by name) in the EEPROM results table but identifies the corresponding entry in the definitions table to be set to [CS\\_STATE\\_EMPTY](#).  
Definition at line 1759 of file cs\_events.h.

**10.1.2.53 CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID** #define CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID 46  
CS EEPROM Checksum Entry ID Enabled Event ID.

Type: INFORMATION

Cause:invalid

This event message is issued when an enable EEPROM Entry ID command is accepted.  
Definition at line 611 of file cs\_events.h.

**10.1.2.54 CS\_ENABLE\_EEPROM\_INF\_EID** #define CS\_ENABLE\_EEPROM\_INF\_EID 38  
CS Enable EEPROM Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when enable checksumming for the EEPROM table command has been received.  
Definition at line 506 of file cs\_events.h.

**10.1.2.55 CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID** #define CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_EID 47  
CS EEPROM Checksum Entry ID Enable Failed Invalid Entry ID Event ID.

Type: ERROR

Cause:

This event message is issued when an enable EEPROM Entry ID command is received, but has an invalid Entry ID.  
Definition at line 623 of file cs\_events.h.

**10.1.2.56 CS\_ENABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID** #define CS\_ENABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID 134  
CS Memory Enable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully enables an entry (specified by name) in the Memory results table but identifies the corresponding entry in the definitions table to be set to [CS\\_STATE\\_EMPTY](#).  
Definition at line 1733 of file cs\_events.h.

**10.1.2.57 CS\_ENABLE\_MEMORY\_ENTRY\_INF\_EID** #define CS\_ENABLE\_MEMORY\_ENTRY\_INF\_EID 61  
CS Memory Checksum Entry ID Enabled Event ID.

Type: INFORMATION

Cause:

This event message is issued when an enable Memory Entry ID command is accepted.  
Definition at line 810 of file cs\_events.h.

**10.1.2.58 CS\_ENABLE\_MEMORY\_INF\_EID** #define CS\_ENABLE\_MEMORY\_INF\_EID 53  
CS Enable Memory Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when enable checksumming for the Memory table command has been received.  
Definition at line 705 of file cs\_events.h.

**10.1.2.59 CS\_ENABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID** #define CS\_ENABLE\_MEMORY\_INVALID\_ENTRY\_EID 62

CS Memory Checksum Entry ID Enable Failed Invalid Entry ID Event ID.

Type: ERROR

Cause:

This event message is issued when an enable Memory Entry ID command is received, but has an invalid Entry ID.  
Definition at line 823 of file cs\_events.h.

**10.1.2.60 CS\_ENABLE\_OS\_INF\_EID** #define CS\_ENABLE\_OS\_INF\_EID 9

CS Enable OS Code Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when enable checksumming for the OS code segment command has been received.  
Definition at line 135 of file cs\_events.h.

**10.1.2.61 CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID** #define CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID 129

CS Table Enable Checksum Unable To Find Entry In Definition Table Event ID.

Type: DEBUG

Cause:

This event message is issued when CS successfully enables an entry (specified by name) in the Tables results table but is unable to find the same entry in the definition table (or the entry is marked as [CS\\_STATE\\_EMPTY](#)).  
Definition at line 1668 of file cs\_events.h.

**10.1.2.62 CS\_ENABLE\_TABLES\_INF\_EID** #define CS\_ENABLE\_TABLES\_INF\_EID 68

CS Enable Tables Checksumming Command Event ID.

Type: INFORMATION

Cause:

This event message is issued when enable checksumming for the Tables table command has been received.  
Definition at line 905 of file cs\_events.h.

**10.1.2.63 CS\_ENABLE\_TABLES\_NAME\_INF\_EID** #define CS\_ENABLE\_TABLES\_NAME\_INF\_EID 76  
CS Table Checksum Enabled Event ID.

Type: INFORMATION

Cause:

This event message is issued when an enable Table name command is accepted.  
Definition at line 1012 of file cs\_events.h.

**10.1.2.64 CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID** #define CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID 77  
CS Table Checksum Failed Table Not Found Event ID.

Type: ERROR

Cause:

This event message is issued when an enable Table name command is received, but has an unknown name or is marked as [CS\\_STATE\\_EMPTY](#).  
Definition at line 1025 of file cs\_events.h.

**10.1.2.65 CS\_EXIT\_ERR\_EID** #define CS\_EXIT\_ERR\_EID 35  
CS App Termination Event ID.

Type: ERROR

Cause:

This event message is issued when CS has a critical error in its main loop  
Definition at line 466 of file cs\_events.h.

**10.1.2.66 CS\_EXIT\_INF\_EID** #define CS\_EXIT\_INF\_EID 149  
CS App Termination Event ID.

Type: INFORMATION

Cause:

This event message is issued when CS has exited without error or exception from its main loop  
Definition at line 1938 of file cs\_events.h.

**10.1.2.67 CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID** #define CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID 50  
CS EEPROM Get Entry ID Event ID.

Type: INFORMATION

Cause:

This event message is issued when a Get Entry ID EEPROM command is received and the command specified address is found in the EEPROM table.

Note that more than one of these event messages can be sent per command (if the address is in several entries in the table).

Definition at line 664 of file cs\_events.h.

**10.1.2.68 CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID** #define CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID 51  
CS EEPROM Get Entry ID Address Not Found Event ID.

Type: INFORMATION

Cause:

This event message is issued when the command specified address in a Get Entry ID EEPROM command cannot be found in the EEPROM table.

Definition at line 677 of file cs\_events.h.

**10.1.2.69 CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID** #define CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID 65  
CS Memory Get Entry ID Event ID.

Type: INFORMATION

Cause:

This event message is issued when a Get Entry ID Memory command is received and the command specified address is found in the Memory table.

Note that more than one of these event messages can be sent per command (if the address is in several entries in the table).

Definition at line 864 of file cs\_events.h.

**10.1.2.70 CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID** #define CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID 66  
CS Memory Get Entry ID Address Not Found Event ID.

Type: INFORMATION

Cause:

This event message is issued when the command specified address in a Get Entry ID Memory command cannot be found in the Memory table.

Definition at line 877 of file cs\_events.h.

**10.1.2.71 CS\_INIT\_APP\_ERR\_EID** #define CS\_INIT\_APP\_ERR\_EID 119  
CS Apps Table Initialization Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the table could not be initialized at startup.

Definition at line 1549 of file cs\_events.h.

**10.1.2.72 CS\_INIT\_EEPROM\_ERR\_EID** #define CS\_INIT EEPROM\_ERR\_EID 116  
CS EEPROM Table Initialization Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the table could not be initialized at startup.

Definition at line 1516 of file cs\_events.h.

**10.1.2.73 CS\_INIT\_INF\_EID** #define CS\_INIT\_INF\_EID 1  
CS Initialization Event ID.

Type: INFORMATION

Cause:

This event message is issued when the CFS Checksum has completed initialization.

Definition at line 41 of file cs\_events.h.

**10.1.2.74 CS\_INIT\_MEMORY\_ERR\_EID** #define CS\_INIT\_MEMORY\_ERR\_EID 117  
CS Memory Table Initialization Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the table could not be initialized at startup.  
Definition at line 1527 of file cs\_events.h.

**10.1.2.75 CS\_INIT\_SB\_CREATE\_ERR\_EID** #define CS\_INIT\_SB\_CREATE\_ERR\_EID 112  
CS Software Bus Create Pipe Failed Event ID.

Type: ERROR

Cause:

This event message is issued when CFE\_SB\_CreatePipe fails for the command pipe  
Definition at line 1469 of file cs\_events.h.

**10.1.2.76 CS\_INIT\_SB\_SUBSCRIBE\_BACK\_ERR\_EID** #define CS\_INIT\_SB\_SUBSCRIBE\_BACK\_ERR\_EID 114  
CS Software Bus Subscribe To Background Cycle Failed Event ID.

Type: ERROR

Cause:

This event message is issued when CFE\_SB\_Subscribe fails to subscribe to the background cycle MID.  
Definition at line 1493 of file cs\_events.h.

**10.1.2.77 CS\_INIT\_SB\_SUBSCRIBE\_CMD\_ERR\_EID** #define CS\_INIT\_SB\_SUBSCRIBE\_CMD\_ERR\_EID 115  
CS Software Bus Subscribe to Command Failed Event ID.

Type: ERROR

Cause:

This event message is issued when CFE\_SB\_Subscribe fails to subscribe to the command MID.  
Definition at line 1505 of file cs\_events.h.

**10.1.2.78 CS\_INIT\_SB\_SUBSCRIBE\_HK\_ERR\_EID** #define CS\_INIT\_SB\_SUBSCRIBE\_HK\_ERR\_EID 113  
CS Software Bus Subscribe To Housekeeping Failed Event ID.

Type: ERROR

Cause:

This event message is issued when CFE\_SB\_Subscribe failed to subscribe to the housekeeping MID.  
Definition at line 1481 of file cs\_events.h.

**10.1.2.79 CS\_INIT\_TABLES\_ERR\_EID** #define CS\_INIT\_TABLES\_ERR\_EID 118  
CS Tables Table Initialization Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the table could not be initialized at startup.  
Definition at line 1538 of file cs\_events.h.

**10.1.2.80 CS\_LEN\_ERR\_EID** #define CS\_LEN\_ERR\_EID 36  
CS Invalid Message Length Event ID.

Type: ERROR

Cause:

This event message is issued when command message is received with a message length that doesn't match the expected value.  
Definition at line 478 of file cs\_events.h.

**10.1.2.81 CS\_MEMORY\_MISCOMPARE\_ERR\_EID** #define CS\_MEMORY\_MISCOMPARE\_ERR\_EID 28  
CS Memory Checksum Miscompare Event ID.

Type: ERROR

Cause:

This event message is issued when a checksum miscompare occurs when checksumming entries in the user-defined memory table  
Definition at line 382 of file cs\_events.h.

**10.1.2.82 CS\_MID\_ERR\_EID** #define CS\_MID\_ERR\_EID 33  
CS Invalid Command Pipe Message ID Event ID.

Type: ERROR

Cause:

This event message is issued when a software bus message is received with an invalid message ID.  
Definition at line 442 of file cs\_events.h.

**10.1.2.83 CS\_NO.BASELINE\_APP\_INF\_EID** #define CS\_NO.BASELINE\_APP\_INF\_EID 83  
CS Baseline App Checksum Pending Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the command specified app command has been received but the baseline has not yet been computed.  
Definition at line 1104 of file cs\_events.h.

**10.1.2.84 CS\_NO.BASELINE\_CFECORE\_INF\_EID** #define CS\_NO.BASELINE\_CFECORE\_INF\_EID 11  
CS Baseline cFE Core Checksum Pending Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the cFE core command has been received but the baseline has not yet been computed.  
Definition at line 161 of file cs\_events.h.

**10.1.2.85 CS\_NO.BASELINE\_EEPROM\_INF\_EID** #define CS\_NO.BASELINE\_EEPROM\_INF\_EID 40  
CS Baseline EEPROM Checksum Pending Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the command specified entry command has been received but the baseline has not yet been computed.  
Definition at line 532 of file cs\_events.h.

**10.1.2.86 CS\_NO\_BASELINE\_MEMORY\_INF\_EID** #define CS\_NO\_BASELINE\_MEMORY\_INF\_EID 55  
CS Baseline Memory Checksum Pending Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the command specified entry command has been received but the baseline has not yet been computed.

Definition at line 731 of file cs\_events.h.

**10.1.2.87 CS\_NO\_BASELINE\_OS\_INF\_EID** #define CS\_NO\_BASELINE\_OS\_INF\_EID 13  
CS Baseline OS Code Checksum Pending Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the cFE core command has been received but the baseline has not yet been computed.

Definition at line 187 of file cs\_events.h.

**10.1.2.88 CS\_NO\_BASELINE\_TABLES\_INF\_EID** #define CS\_NO\_BASELINE\_TABLES\_INF\_EID 70  
CS Baseline Table Checksum Pending Event ID.

Type: INFORMATION

Cause:

This event message is issued when a report baseline for the command specified table command has been received but the baseline has not yet been computed.

Definition at line 931 of file cs\_events.h.

**10.1.2.89 CS\_NOOP\_INF\_EID** #define CS\_NOOP\_INF\_EID 2  
CS No-op Command Event ID.

Type: INFORMATION

Cause:

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

Definition at line 52 of file cs\_events.h.

**10.1.2.90 CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID** #define CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID 25

CS Oneshot Checksum Failed Child Task Delete Event ID.

Type: ERROR

Cause:

This event message is issued when a cancel OneShot calculation command has been received and the cancel OneShot failed because CFE\_ES\_DeleteChildTask returned an error.

Definition at line 345 of file cs\_events.h.

**10.1.2.91 CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID** #define CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID 26

CS Oneshot Checksum Cancel Failed No Oneshot Active Event ID.

Type: ERROR

Cause:

This event message is issued when a cancel OneShot command has been received and the cancel OneShot failed because there was no OneShot child task running.

Definition at line 358 of file cs\_events.h.

**10.1.2.92 CS\_ONESHOT\_CANCELLED\_INF\_EID** #define CS\_ONESHOT\_CANCELLED\_INF\_EID 24

CS Oneshot Checksum Cancelled Event ID.

Type: INFORMATION

Cause:

This event message is issued when a cancel OneShot calculation command has been received and the OneShot task has been cancelled.

Definition at line 332 of file cs\_events.h.

**10.1.2.93 CS\_ONESHOT\_CHDTASK\_ERR\_EID** #define CS\_ONESHOT\_CHDTASK\_ERR\_EID 22

CS Oneshot Checksum Failed Child Task In Use Event ID.

Type: ERROR

Cause:

This event message is issued when a OneShot command has been received and the OneShot failed because there is already a CS child task running.

Definition at line 307 of file cs\_events.h.

**10.1.2.94 CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID** #define CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID 21  
CS Oneshot Checksum Failed Child Task Create Event ID.

Type: ERROR

Cause:

This event message is issued when a OneShot calculation command has been received and the OneShot failed because CFE\_ES\_CreateChildTask returned an error.

Definition at line 294 of file cs\_events.h.

**10.1.2.95 CS\_ONESHOT\_FINISHED\_INF\_EID** #define CS\_ONESHOT\_FINISHED\_INF\_EID 100  
CS Oneshot Checksum Finished Event ID.

Type: INFORMATION

Cause:

This event message is issued when a OneShot command finishes

Definition at line 1328 of file cs\_events.h.

**10.1.2.96 CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID** #define CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID 23  
CS Oneshot Checksum Failed Invalid Memory Range Event ID.

Type: ERROR

Cause:

This event message is issued when a OneShot command has been received and the OneShot failed because CFE\_PSP\_MemValidateRange returned an error.

Definition at line 320 of file cs\_events.h.

**10.1.2.97 CS\_ONESHOT\_STARTED\_DBG\_EID** #define CS\_ONESHOT\_STARTED\_DBG\_EID 20  
CS Oneshot Checksum Started Event ID.

Type: DEBUG

Cause:

This event message is issued when a OneShot calculation command has been received and the OneShot task has been started.

Definition at line 281 of file cs\_events.h.

**10.1.2.98 CS\_OS\_MISCOMPARE\_ERR\_EID** #define CS\_OS\_MISCOMPARE\_ERR\_EID 32  
CS OS Code Checksum Miscompare Event ID.

Type: ERROR

Cause:

This event message is issued when a checksum miscompare occurs when checksumming the OS code segment.  
Definition at line 430 of file cs\_events.h.

**10.1.2.99 CS\_OS\_TEXT\_SEG\_INF\_EID** #define CS\_OS\_TEXT\_SEG\_INF\_EID 127  
CS OS Code Checksum Disabled Platform Not Supported Event ID.

Type: INFORMATION

Cause:

This event message is issued when CS is running on a platform that does not support getting the information of the OS text segment, such as OS X and Linux.  
Definition at line 1643 of file cs\_events.h.

**10.1.2.100 CS\_PROCESS\_APP\_NO\_ENTRIES\_INF\_EID** #define CS\_PROCESS\_APP\_NO\_ENTRIES\_INF\_EID 108  
CS Apps Table No Valid Entries Event ID.

Type: INFORMATION

Cause:

This event message is issued when a new definition table has finished being processed and there are no valid entries in that table  
Definition at line 1422 of file cs\_events.h.

**10.1.2.101 CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID** #define CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID 107  
CS EEPROM or Memory Table No Valid Entries Event ID.

Type: INFORMATION

Cause:

This event message is issued when a new definition table has finished being processed and there are no valid entries in that table  
Definition at line 1410 of file cs\_events.h.

**10.1.2.102 CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID** #define CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID 109  
CS Tables Table No Valid Entries Event ID.  
Type: INFORMATION

Cause:

This event message is issued when a new definition table has finished being processed and there are no valid entries in that table.

Definition at line 1434 of file cs\_events.h.

**10.1.2.103 CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID 88  
CS App Checksum Recompute Failed Child Task In Use Event ID.  
Type: ERROR

Cause:

This event message is issued when a recompute baseline for the command specified app command has been received and the recompute failed because there is already a CS child task running.

Definition at line 1173 of file cs\_events.h.

**10.1.2.104 CS\_RECOMPUTE\_APP\_CREATE\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_APP\_CREATE\_CHDERR\_EID 86  
CS App Checksum Recompute Failed Create Child Task Event ID.  
Type: ERROR

Cause:

This event message is issued when a recompute baseline for the specified app command has been received and the recompute failed because CFE\_ES\_CreateChildTask returned an error.

Definition at line 1145 of file cs\_events.h.

**10.1.2.105 CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID** #define CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID 85  
CS App Checksum Recompute Started Event ID.

Type: DEBUG

Cause:

This event message is issued when a recompute baseline for the specified app command has been received and the recompute task has been started.

Definition at line 1131 of file cs\_events.h.

**10.1.2.106 CS\_RECOMPUTE\_CFECore\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_CFECore\_CHDTASK\_ER←  
R\_EID 16

CS cFE Core Checksum Failed Child Task In Use Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute baseline for the cFE core command has been received and the recompute failed because there is already a CS child task running.

Definition at line 228 of file cs\_events.h.

**10.1.2.107 CS\_RECOMPUTE\_CFECore\_CREATE\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_CFECore\_C←  
REATE\_CHDTASK\_ERR\_EID 15

CS cFE Core Checksum Failed Child Task Create Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute baseline for the cFE core command has been received and the recompute failed because CFE\_ES\_CreateChildTask returned an error.

Definition at line 214 of file cs\_events.h.

**10.1.2.108 CS\_RECOMPUTE\_CFECore\_STARTED\_DBG\_EID** #define CS\_RECOMPUTE\_CFECore\_STARTED\_DB←  
G\_EID 14

CS cFE Core Checksum Recompute Started Event ID.

Type: DEBUG

Cause:

This event message is issued when a recompute baseline for the cFE core command has been received and the recompute task has been started.

Definition at line 200 of file cs\_events.h.

**10.1.2.109 CS\_RECOMPUTE\_EEPROM\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_EEPROM\_CHDTASK\_ERR\_←  
EID 45

CS EEPROM Checksum Recompute Failed Child Task In Use Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute baseline for the command specified Entry ID command has been received and the recompute failed because there is already a CS child task running.

Definition at line 599 of file cs\_events.h.

**10.1.2.110 CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID 43  
CS EEPROM Checksum Recompute Failed Child Task Create Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute baseline for the specified EEPROM Entry ID command has been received and the recompute failed because CFE\_ES\_CreateChildTask returned an error.

Definition at line 572 of file cs\_events.h.

**10.1.2.111 CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID** #define CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID 42  
CS EEPROM Checksum Recompute Started Event ID.

Type: DEBUG

**Cause:**

This event message is issued when a recompute baseline for the specified EEPROM Entry ID command has been received and the recompute task has been started.

Definition at line 558 of file cs\_events.h.

**10.1.2.112 CS\_RECOMPUTE\_ERROR\_APP\_ERR\_EID** #define CS\_RECOMPUTE\_ERROR\_APP\_ERR\_EID 97  
CS App Checksum Recompute Failed Could Not Get Address Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute entry for Apps fails because CS cannot get the address of the application.

Definition at line 1286 of file cs\_events.h.

**10.1.2.113 CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID** #define CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID 96  
CS Table Checksum Recompute Failed Could Not Get Address Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute entry for Tables fails because CS cannot get the address of the table.  
Definition at line 1274 of file cs\_events.h.

**10.1.2.114 CS\_RECOMPUTE\_FINISH\_APP\_INF\_EID** #define CS\_RECOMPUTE\_FINISH\_APP\_INF\_EID 99  
CS App Recompute Finished Event ID.

Type: INFORMATION

Cause:

This event message is issued when a recompute entry for an app has finished successfully.  
Definition at line 1317 of file cs\_events.h.

**10.1.2.115 CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID** #define CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID 95  
CS EEMROM or Memory Checksum Recompute Finished Event ID.

Type: INFORMATION

Cause:

This event message is issued when a recompute entry for EEPROM or Memory has finished successfully.  
Definition at line 1262 of file cs\_events.h.

**10.1.2.116 CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID** #define CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID 98  
CS Table Recompute Finished Event ID.

Type: INFORMATION

Cause:

This event message is issued when a recompute entry for a Table has finished successfully.  
For the CS Table Definitions Table only, the checksum value will be incorrect. This is because all entries in this table are disabled while being recomputed and disabling the entry for itself modifies the contents of the table being recomputed.  
Thus, recomputing the CS Tables Definition Table checksum is not recommended.  
Definition at line 1305 of file cs\_events.h.

**10.1.2.117 CS\_RECOMPUTE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID** #define CS\_RECOMPUTE\_INVALID\_ENTRY←  
\_EEPROM\_ERR\_EID 44  
CS EEPROM Checksum Recompute Failed Invalid Entry ID Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute baseline for the command specified entry command has been received but specified Entry ID is invalid.

Definition at line 585 of file cs\_events.h.

**10.1.2.118 CS\_RECOMPUTE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID** #define CS\_RECOMPUTE\_INVALID\_ENTRY←  
\_MEMORY\_ERR\_EID 59  
CS Memory Checksum Recompute Failed Invalid Entry ID Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute baseline for the command specified entry command has been received but specified Entry ID is invalid.

Definition at line 784 of file cs\_events.h.

**10.1.2.119 CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR←  
\_EID 60  
CS Memory Checksum Recompute Failed Child Task In Use Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute baseline for the command specified Entry ID command has been received and the recompute failed because there is already a CS child task running.

Definition at line 798 of file cs\_events.h.

**10.1.2.120 CS\_RECOMPUTE\_MEMORY\_CREATE\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_MEMORY\_CRE←  
ATE\_CHDTASK\_ERR\_EID 58  
CS Memory Checksum Recompute Failed Child Task Create Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute baseline for the specified Memory Entry ID command has been received and the recompute failed because CFE\_ES\_CreateChildTask returned an error.

Definition at line 771 of file cs\_events.h.

**10.1.2.121 CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID** #define CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID 57

CS Memory Checksum Recompute Started Event ID.

Type: DEBUG

**Cause:**

This event message is issued when a recompute baseline for the specified Memory Entry ID command has been received and the recompute task has been started.

Definition at line 757 of file cs\_events.h.

**10.1.2.122 CS\_RECOMPUTE\_OS\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_OS\_CHDTASK\_ERR\_EID 19

CS OS Code Checksum Recompute Failed Child Task In Use Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute baseline for the OS code segment command has been received and the recompute failed because there is already a CS child task running.

Definition at line 269 of file cs\_events.h.

**10.1.2.123 CS\_RECOMPUTE\_OS\_CREATE\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_OS\_CREATE\_CHDTASK\_ERR\_EID 18

CS OS Code Checksum Recompute Failed Child Task Create Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute baseline for the OS code segment command has been received and the recompute failed because CFE\_ES\_CreateChildTask returned an error.

Definition at line 255 of file cs\_events.h.

**10.1.2.124 CS\_RECOMPUTE\_OS\_STARTED\_DBG\_EID** #define CS\_RECOMPUTE\_OS\_STARTED\_DBG\_EID 17  
CS OS Code Checksum Recompute Started Event ID.

Type: DEBUG

Cause:

This event message is issued when a recompute baseline for the OS code segment command has been received and the recompute task has been started.

Definition at line 241 of file cs\_events.h.

**10.1.2.125 CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID 75  
CS Table Checksum Recompute Failed Child Task In Use Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute baseline for the command specified table command has been received and the recompute failed because there is already a CS child task running.

Definition at line 1000 of file cs\_events.h.

**10.1.2.126 CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID** #define CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID 73  
CS Table Checksum Recompute Failed Child Task Create Event ID.

Type: ERROR

Cause:

This event message is issued when a recompute baseline for the specified Tables Entry ID command has been received and the recompute failed because CFE\_ES\_CreateChildTask returned an error.

Definition at line 972 of file cs\_events.h.

**10.1.2.127 CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID** #define CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID 72  
CS Table Checksum Recompute Started Event ID.

Type: DEBUG

**Cause:**

This event message is issued when a recompute baseline for the specified table command has been received and the recompute task has been started.

Definition at line 958 of file cs\_events.h.

**10.1.2.128 CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID** #define CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID 87

CS App Checksum Recompute Failed App Not Found Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute baseline for the command specified app command has been received but specified app cannot be found in CS or is marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1159 of file cs\_events.h.

**10.1.2.129 CS\_RECOMPUTE\_UNKNOWN\_NAME\_TABLES\_ERR\_EID** #define CS\_RECOMPUTE\_UNKNOWN\_NAME\_TABLES\_ERR\_EID 74

CS Table Checksum Recompute Failed Table Not Found Event ID.

Type: ERROR

**Cause:**

This event message is issued when a recompute baseline for the command specified table command has been received but specified table cannot be found in CS or is marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 986 of file cs\_events.h.

**10.1.2.130 CS\_RESET\_DBG\_EID** #define CS\_RESET\_DBG\_EID 3

CS Reset Counters Command Event ID.

Type: DEBUG

**Cause:**

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

Definition at line 63 of file cs\_events.h.

**10.1.2.131 CS\_TABLES\_MISCOMPARE\_ERR\_EID** #define CS\_TABLES\_MISCOMPARE\_ERR\_EID 29  
CS Table Checksum Miscompare Event ID.

Type: ERROR

Cause:

This event message is issued when a checksum miscompare occurs when checksumming entries in the table of tables to checksum.

Definition at line 394 of file cs\_events.h.

**10.1.2.132 CS\_TBL\_INIT\_ERR\_EID** #define CS\_TBL\_INIT\_ERR\_EID 110  
CS Table Initialization Failed Event ID.

Type: ERROR

Cause:

This event message is issued when a table initialization failed.

Definition at line 1445 of file cs\_events.h.

**10.1.2.133 CS\_TBL\_UPDATE\_ERR\_EID** #define CS\_TBL\_UPDATE\_ERR\_EID 111  
CS Table Update Failed Event ID.

Type: ERROR

Cause:

This event message is issued when a problem occurs during a table update.

Definition at line 1456 of file cs\_events.h.

**10.1.2.134 CS\_UPDATE\_APP\_ERR\_EID** #define CS\_UPDATE\_APP\_ERR\_EID 126  
CS Apps Table Update Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the Apps table can not be updated.

Definition at line 1631 of file cs\_events.h.

**10.1.2.135 CS\_UPDATE\_CDS\_ERR\_EID** #define CS\_UPDATE\_CDS\_ERR\_EID 148  
CS Critical Data Store Update Failed Event ID.

Type: ERROR

Cause:

The CS application optionally stores the table states in the Critical Data Store (CDS). This ensures that CS will not overwrite old data storage files following a processor reset. This event is issued if CS cannot update the CDS.  
Definition at line 1926 of file cs\_events.h.

**10.1.2.136 CS\_UPDATE\_EEPROM\_ERR\_EID** #define CS\_UPDATE EEPROM\_ERR\_EID 123  
CS EEPROM Table Update Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the EEPROM table can not be updated.  
Definition at line 1598 of file cs\_events.h.

**10.1.2.137 CS\_UPDATE\_MEMORY\_ERR\_EID** #define CS\_UPDATE\_MEMORY\_ERR\_EID 124  
CS Memory Table Update Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the Meomory table can not be updated.  
Definition at line 1609 of file cs\_events.h.

**10.1.2.138 CS\_UPDATE\_TABLES\_ERR\_EID** #define CS\_UPDATE\_TABLES\_ERR\_EID 125  
CS Tables Table Update Failed Event ID.

Type: ERROR

Cause:

This event message is issued when the Tables table can not be updated.  
Definition at line 1620 of file cs\_events.h.

**10.1.2.139 CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID** #define CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID 140  
CS Apps Table Validate Failed Duplicate Name Event ID.

Type: ERROR

Cause:

This event message is issued when CS validation for the Apps definition table finds more than one entry with the same table name. Only one entry per table is allowed. This event is only issued if it is the first error found during validation.  
Definition at line 1812 of file cs\_events.h.

**10.1.2.140 CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID** #define CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID 154  
CS Apps Table Validate Failed Illegal State With Long Name Event ID.

Type: ERROR

Cause:

This event message is issued when CS validation for the Apps definition table finds an entry that contains a non-terminated name field.  
Definition at line 1987 of file cs\_events.h.

**10.1.2.141 CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID** #define CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID 141  
CS Apps Table Validate Failed Illegal State With Empty Name Event ID.

Type: ERROR

Cause:

This event message is issued when CS validation for the Apps definition table finds an entry that contains a zero-length name field with a state field that is not marked as [CS\\_STATE\\_EMPTY](#). This event is only issued if it is the first error found during validation.

Definition at line 1825 of file cs\_events.h.

**10.1.2.142 CS\_VAL\_APP\_INF\_EID** #define CS\_VAL\_APP\_INF\_EID 142  
CS Apps Table Verification Results Event ID.

Type: INFORMATION

**Cause:**

This event message when CS completes validation of the Apps definition table. This message reports the number of successful ([CS\\_STATE\\_ENABLED](#) or [CS\\_STATE\\_DISABLED](#)) entries, the number of bad entries (due to invalid state definitions or duplicate names), and the number of entries marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1839 of file cs\_events.h.

**10.1.2.143 CS\_VAL\_APP\_STATE\_ERR\_EID** #define CS\_VAL\_APP\_STATE\_ERR\_EID 106  
CS Apps Table Validate Failed Illegal State Field Event ID.

Type: ERROR

**Cause:**

This event message is issued when the App table validation function detects an illegal state  
Definition at line 1398 of file cs\_events.h.

**10.1.2.144 CS\_VAL\_EEPROM\_INF\_EID** #define CS\_VAL\_EEPROM\_INF\_EID 144  
CS EEPROM Table Verification Results Event ID.

Type: INFORMATION

**Cause:**

This event message when CS completes validation of the EEPROM definition table. This message reports the number of successful ([CS\\_STATE\\_ENABLED](#) or [CS\\_STATE\\_DISABLED](#)) entries, the number of bad entries (due to invalid state definitions or bad range), and the number of entries marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1867 of file cs\_events.h.

**10.1.2.145 CS\_VAL\_EEPROM\_RANGE\_ERR\_EID** #define CS\_VAL\_EEPROM\_RANGE\_ERR\_EID 102  
CS EEPROM Table Validate Failed Illegal Checksum Range Event ID.

Type: ERROR

**Cause:**

This event message is issued when the EEPROM table validation function detects an illegal checksum range  
Definition at line 1354 of file cs\_events.h.

**10.1.2.146 CS\_VAL\_EEPROM\_STATE\_ERR\_EID** #define CS\_VAL\_EEPROM\_STATE\_ERR\_EID 101  
CS EEPROM Table Validate Failed Illegal State Field Event ID.

Type: ERROR

Cause:

This event message is issued when the EEPROM table validation function detects an illegal state  
Definition at line 1343 of file cs\_events.h.

**10.1.2.147 CS\_VAL\_MEMORY\_INF\_EID** #define CS\_VAL\_MEMORY\_INF\_EID 143  
CS Memory Table Verification Results Event ID.

Type: INFORMATION

Cause:

This event message when CS completes validation of the Memory definition table. This message reports the number of successful ([CS\\_STATE\\_ENABLED](#) or [CS\\_STATE\\_DISABLED](#)) entries, the number of bad entries (due to invalid state definitions or bad range), and the number of entries marked as [CS\\_STATE\\_EMPTY](#).  
Definition at line 1853 of file cs\_events.h.

**10.1.2.148 CS\_VAL\_MEMORY\_RANGE\_ERR\_EID** #define CS\_VAL\_MEMORY\_RANGE\_ERR\_EID 104  
CS Memory Table Validate Failed Illegal Checksum Range Event ID.

Type: ERROR

Cause:

This event message is issued when the Memory table validation function detects an illegal checksum range  
Definition at line 1376 of file cs\_events.h.

**10.1.2.149 CS\_VAL\_MEMORY\_STATE\_ERR\_EID** #define CS\_VAL\_MEMORY\_STATE\_ERR\_EID 103  
CS Memory Table Valide Failed Illegal State Field Event ID.

Type: ERROR

Cause:

This event message is issued when the Memory table validation function detects an illegal state  
Definition at line 1365 of file cs\_events.h.

**10.1.2.150 CS\_VAL\_TABLES\_DEF\_TBL\_DUPL\_ERR\_EID** #define CS\_VAL\_TABLES\_DEF\_TBL\_DUPL\_ERR\_EID 137

CS Tables Table Validate Failed Duplicate Name Event ID.

Type: ERROR

Cause:

This event message is issued when CS validation for the Tables definition table finds more than one entry with the same table name. Only one entry per table is allowed. This event is only issued if it is the first error found during validation. Definition at line 1772 of file cs\_events.h.

**10.1.2.151 CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID** #define CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID 138

CS Tables Table Validate Failed Illegal State With Empty Name Event ID.

Type: ERROR

Cause:

This event message is issued when CS validation for the Tables definition table finds an entry that contains a zero-length name field with a state field that is not marked as [CS\\_STATE\\_EMPTY](#). This event is only issued if it is the first error found during validation.

Definition at line 1785 of file cs\_events.h.

**10.1.2.152 CS\_VAL\_TABLES\_INF\_EID** #define CS\_VAL\_TABLES\_INF\_EID 139

CS Tables Table Verification Results Event ID.

Type: INFORMATION

Cause:

This event message when CS completes validation of the tables definition table. This message reports the number of successful ([CS\\_STATE\\_ENABLED](#) or [CS\\_STATE\\_DISABLED](#)) entries, the number of bad entries (due to invalid state definitions or duplicate names), and the number of entries marked as [CS\\_STATE\\_EMPTY](#).

Definition at line 1799 of file cs\_events.h.

**10.1.2.153 CS\_VAL\_TABLES\_STATE\_ERR\_EID** #define CS\_VAL\_TABLES\_STATE\_ERR\_EID 105

CS Tables Table Validate Failed Illegal State Field Event ID.

Type: ERROR

Cause:

This event message is issued when the Tables table validation function detects an illegal state  
Definition at line 1387 of file cs\_events.h.

## 10.2 CFS Checksum Mission Configuration

### Macros

- `#define CS_DEFAULT_ALGORITHM CFE_MISSION_ES_DEFAULT_CRC`  
*default CRC algorithm*
- `#define CS_APPMAIN_PERF_ID 29`  
*Main application performance ID.*

#### 10.2.1 Detailed Description

#### 10.2.2 Macro Definition Documentation

##### 10.2.2.1 CS\_APPMAIN\_PERF\_ID `#define CS_APPMAIN_PERF_ID 29`

Main application performance ID.

Definition at line 32 of file cs\_perfids.h.

##### 10.2.2.2 CS\_DEFAULT\_ALGORITHM `#define CS_DEFAULT_ALGORITHM CFE_MISSION_ES_DEFAULT_CRC`

default CRC algorithm

#### Description:

This parameter is the algorithm used by CS to checksum the requested data.

#### Limits:

This parameter is limited to either `CFE_MISSION_ES_DEFAULT_CRC`, or `CFE_ES_CrcType_CRC_16`

Definition at line 46 of file cs\_mission\_cfg.h.

## 10.3 CFS Checksum Telemetry

### Data Structures

- struct [CS\\_HkPacket\\_Payload\\_t](#)  
*Housekeeping Payload Structure.*
- struct [CS\\_HkPacket\\_t](#)  
*Housekeeping Packet Structure.*

#### 10.3.1 Detailed Description

## 10.4 CFS Checksum Command Structures

### Data Structures

- struct [CS\\_GetEntryIDCmd\\_Payload\\_t](#)  
*Get entry ID command payload.*
- struct [CS\\_EntryCmd\\_Payload\\_t](#)  
*Payload for commands using Memory or EEPROM tables.*
- struct [CS\\_TableNameCmd\\_Payload\\_t](#)  
*Payload for commanding by table name.*
- struct [CS\\_AppNameCmd\\_Payload\\_t](#)  
*Payload for commanding by app name.*
- struct [CS\\_OneShotCmd\\_Payload\\_t](#)  
*Payload for sending one shot calculation.*
- struct [CS\\_NoArgsCmd\\_t](#)  
*No arguments command data type.*
- struct [CS\\_GetEntryIDCmd\\_t](#)  
*Get entry ID command.*
- struct [CS\\_EntryCmd\\_t](#)  
*Command type for commands using Memory or EEPROM tables.*
- struct [CS\\_TableNameCmd\\_t](#)  
*Command type for commanding by table name.*
- struct [CS\\_AppNameCmd\\_t](#)  
*Command type for commanding by app name.*
- struct [CS\\_OneShotCmd\\_t](#)  
*Command type for sending one shot calculation.*

#### 10.4.1 Detailed Description

## 10.5 CFS Checksum Command Codes

### Macros

- #define CS\_NOOP\_CC 0  
*Noop.*
- #define CS\_RESET\_CC 1  
*Reset Counters.*
- #define CS\_ONE\_SHOT\_CC 2  
*Start One shot calculation.*
- #define CS\_CANCEL\_ONE\_SHOT\_CC 3  
*Cancel one shot.*
- #define CS\_ENABLE\_ALL\_CS\_CC 4  
*Enable Global Checksumming.*
- #define CS\_DISABLE\_ALL\_CS\_CC 5  
*Disable Global Checksumming.*
- #define CS\_ENABLE\_CFE\_CORE\_CC 6  
*Enable checksumming of cFE core.*
- #define CS\_DISABLE\_CFE\_CORE\_CC 7  
*Disable checksumming of cFE core.*
- #define CS\_REPORT\_BASELINE\_CFE\_CORE\_CC 8  
*Report Baseline checksum of cFE core.*
- #define CS\_RECOMPUTE\_BASELINE\_CFE\_CORE\_CC 9  
*Recompute Baseline checksum of cFE core.*
- #define CS\_ENABLE\_OS\_CC 10  
*Enable checksumming of OS code segment.*
- #define CS\_DISABLE\_OS\_CC 11  
*Disable checksumming of OS code segment.*
- #define CS\_REPORT\_BASELINE\_OS\_CC 12  
*Report Baseline checksum of OS code segment.*
- #define CS\_RECOMPUTE\_BASELINE\_OS\_CC 13  
*Recompute Baseline checksum of OS code segment.*
- #define CS\_ENABLE\_EEPROM\_CC 14  
*Enable checksumming for EEPROM table.*
- #define CS\_DISABLE\_EEPROM\_CC 15  
*Disable checksumming for EEPROM table.*
- #define CS\_REPORT\_BASELINE\_EEPROM\_CC 16  
*Report Baseline checksum of EEPROM Entry.*
- #define CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC 17  
*Recompute Baseline checksum of EEPROM Entry.*
- #define CS\_ENABLE\_ENTRY\_EEPROM\_CC 18  
*Enable checksumming for an EEPROM entry.*
- #define CS\_DISABLE\_ENTRY\_EEPROM\_CC 19  
*Disable checksumming for an EEPROM entry.*
- #define CS\_GET\_ENTRY\_ID\_EEPROM\_CC 20  
*Get the Entry ID for a given EEPROM address.*
- #define CS\_ENABLE\_MEMORY\_CC 21  
*Enable checksumming for Memory table.*

- #define CS\_DISABLE\_MEMORY\_CC 22  
*Disable checksumming for Memory table.*
- #define CS\_REPORT\_BASELINE\_MEMORY\_CC 23  
*Report Baseline checksum of Memory Entry.*
- #define CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC 24  
*Recompute Baseline checksum of Memory Entry.*
- #define CS\_ENABLE\_ENTRY\_MEMORY\_CC 25  
*Enable checksumming for a Memory entry.*
- #define CS\_DISABLE\_ENTRY\_MEMORY\_CC 26  
*Disable checksumming for a Memory entry.*
- #define CS\_GET\_ENTRY\_ID\_MEMORY\_CC 27  
*Get the Entry ID for a given Memory address.*
- #define CS\_ENABLE\_TABLES\_CC 28  
*Enable checksumming for Tables table.*
- #define CS\_DISABLE\_TABLES\_CC 29  
*Disable checksumming for Tables table.*
- #define CS\_REPORT\_BASELINE\_TABLE\_CC 30  
*Report Baseline checksum of a table.*
- #define CS\_RECOMPUTE\_BASELINE\_TABLE\_CC 31  
*Recompute Baseline checksum of a table.*
- #define CS\_ENABLE\_NAME\_TABLE\_CC 32  
*Enable checksumming for a table.*
- #define CS\_DISABLE\_NAME\_TABLE\_CC 33  
*Disable checksumming for a table.*
- #define CS\_ENABLE\_APPS\_CC 34  
*Enable checksumming for App table.*
- #define CS\_DISABLE\_APPS\_CC 35  
*Disable checksumming for App table.*
- #define CS\_REPORT\_BASELINE\_APP\_CC 36  
*Report Baseline checksum of an app.*
- #define CS\_RECOMPUTE\_BASELINE\_APP\_CC 37  
*Recompute Baseline checksum of an app.*
- #define CS\_ENABLE\_NAME\_APP\_CC 38  
*Enable checksumming for an app.*
- #define CS\_DISABLE\_NAME\_APP\_CC 39  
*Disable checksumming for an app.*

### 10.5.1 Detailed Description

### 10.5.2 Macro Definition Documentation

**10.5.2.1 CS\_CANCEL\_ONE\_SHOT\_CC** #define CS\_CANCEL\_ONE\_SHOT\_CC 3  
Cancel one shot.

#### Description

Cancels a one shot calculation that is already in progress.

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ONESHOT\\_CANCELLED\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- A one shot calculation is not in progress
- The child task could not be deleted

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_ONESHOT\\_CANCEL\\_NO\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_ONESHOT\\_CANCEL\\_DELETE\\_CHDTASK\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ONE\\_SHOT\\_CC](#)

Definition at line 182 of file `cs_msgdefs.h`.

**10.5.2.2 CS\_DISABLE\_ALL\_CS\_CC** #define CS\_DISABLE\_ALL\_CS\_CC 5  
Disable Global Checksumming.

#### Description

Disables all background checking

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_ALL\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.ChecksumState](#) set to [CS\\_STATE\\_DISABLED](#)

### Error Conditions

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

- Command packet length not as expected

### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

### Criticality

None

### See also

[CS\\_ENABLE\\_ALL\\_CS\\_CC](#)

Definition at line 246 of file cs\_msgdefs.h.

## 10.5.2.3 CS\_DISABLE\_APPS\_CC [#define CS\\_DISABLE\\_APPS\\_CC 35](#)

Disable checksumming for App table.

### Description

Disable the App table background checksumming

### Command Structure

[CS\\_NoArgsCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_APP\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.AppCSSState](#) set to [CS\\_STATE\\_DISABLED](#)

### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_APPS\\_CC](#)

Definition at line 1296 of file cs\_msgdefs.h.

**10.5.2.4 CS\_DISABLE\_CFE\_CORE\_CC** `#define CS_DISABLE_CFE_CORE_CC 7`  
Disable checksumming of cFE core.

#### Description

Disables background checking on the cFE core code segment

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_CFECORE\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.CfeCoreCSState](#) set to [CS\\_STATE\\_DISABLED](#)

#### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_CFE\\_CORE\\_CC](#)

Definition at line 310 of file cs\_msgdefs.h.

**10.5.2.5 CS\_DISABLE\_EEPROM\_CC** #define CS\_DISABLE\_EEPROM\_CC 15  
Disable checksumming for EEPROM table.

#### Description

Disable the EEPROM table background checksumming

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_EEPROM\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.EepromCSState](#) set to [CS\\_STATE\\_DISABLED](#)

#### Error Conditions

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

- Command packet length not as expected

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_EEPROM\\_CC](#)

Definition at line 588 of file cs\_msgdefs.h.

**10.5.2.6 CS\_DISABLE\_ENTRY\_EEPROM\_CC** #define CS\_DISABLE\_ENTRY\_EEPROM\_CC 19  
Disable checksumming for an EEPROM entry.

#### Description

Disable the EEPROM entry background checksumming

#### Command Structure

[CS\\_EntryCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_EEPROM\\_ENTRY\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified entry was invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_DISABLE\\_EEPROM\\_INVALID\\_ENTRY\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_ENTRY\\_EEPROM\\_CC](#)

Definition at line 737 of file cs\_msgdefs.h.

### 10.5.2.7 CS\_DISABLE\_ENTRY\_MEMORY\_CC #define CS\_DISABLE\_ENTRY\_MEMORY\_CC 26

Disable checksumming for a Memory entry.

#### Description

Disable the Memory entry background checksumming

#### Command Structure

[CS\\_EntryCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_MEMORY\\_ENTRY\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified entry was invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_DISABLE\\_MEMORY\\_INVALID\\_ENTRY\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_ENTRY\\_MEMORY\\_CC](#)

Definition at line 981 of file cs\_msgdefs.h.

**10.5.2.8 CS\_DISABLE\_MEMORY\_CC** #define CS\_DISABLE\_MEMORY\_CC 22  
Disable checksumming for Memory table.

#### Description

Disable the Memory table background checksumming

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_MEMORY\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.MemoryCSState](#) set to [CS\\_STATE\\_DISABLED](#)

#### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_MEMORY\\_CC](#)

Definition at line 833 of file cs\_msgdefs.h.

**10.5.2.9 CS\_DISABLE\_NAME\_APP\_CC** #define CS\_DISABLE\_NAME\_APP\_CC 39  
Disable checksumming for an app.

#### Description

Disable background checking of the app

#### Command Structure

[CS\\_AppNameCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_APP\\_NAME\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified app name was invalid

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_DISABLE\\_APP\\_NAME\\_INF\\_EID](#)
- Error specific event message [CS\\_DISABLE\\_APP\\_UNKNOWN\\_NAME\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_NAME\\_APP\\_CC](#)

Definition at line 1444 of file cs\_msgdefs.h.

**10.5.2.10 CS\_DISABLE\_NAME\_TABLE\_CC** #define CS\_DISABLE\_NAME\_TABLE\_CC 33  
Disable checksumming for a table.

#### Description

Disable background checking of the table

#### Command Structure

[CS\\_TableNameCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_TABLES\\_NAME\\_INF\\_EID](#) informational event message will be generated when the command is received

### Error Conditions

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

- Command packet length not as expected
- Command specified table name was invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_DISABLE\\_TABLES\\_NAME\\_INF\\_EID](#)
- Error specific event message [CS\\_DISABLE\\_TABLES\\_UNKNOWN\\_NAME\\_ERR\\_EID](#)

### Criticality

None

### See also

[CS\\_DISABLE\\_NAME\\_TABLE\\_CC](#)

Definition at line 1232 of file cs\_msgdefs.h.

### **10.5.2.11 CS\_DISABLE\_OS\_CC #define CS\_DISABLE\_OS\_CC 11**

Disable checksumming of OS code segment.

### Description

Disables background checking on the OS code segment code segment

### Command Structure

[CS\\_NoArgsCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_OS\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.OSCSState](#) set to [CS\\_STATE\\_DISABLED](#)

### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_OS\\_CC](#)

Definition at line 450 of file cs\_msgdefs.h.

### **10.5.2.12 CS\_DISABLE\_TABLES\_CC** #define CS\_DISABLE\_TABLES\_CC 29

Disable checksumming for Tables table.

#### Description

Disable the Tables table background checksumming

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_DISABLE\\_TABLES\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.TablesCSSState](#) set to [CS\\_STATE\\_DISABLED](#)

#### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ENABLE\\_TABLES\\_CC](#)

Definition at line 1078 of file cs\_msgdefs.h.

**10.5.2.13 CS\_ENABLE\_ALL\_CS\_CC** #define CS\_ENABLE\_ALL\_CS\_CC 4  
Enable Global Checksumming.

#### Description

Allows CS to continue background checking

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_ALL\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.ChecksumState](#) set to [CS\\_STATE\\_ENABLED](#)

#### Error Conditions

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

- Command packet length not as expected

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_ALL\\_CS\\_CC](#)

Definition at line 214 of file cs\_msgdefs.h.

**10.5.2.14 CS\_ENABLE\_APPS\_CC** #define CS\_ENABLE\_APPS\_CC 34  
Enable checksumming for App table.

#### Description

Allow the App table to checksummed in the background

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_APP\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.AppCSSState](#) set to [CS\\_STATE\\_ENABLED](#)

#### Error Conditions

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

- Command packet length not as expected

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_APPS\\_CC](#)

Definition at line 1264 of file cs\_msgdefs.h.

### 10.5.2.15 CS\_ENABLE\_CFE\_CORE\_CC #define CS\_ENABLE\_CFE\_CORE\_CC 6

Enable checksumming of cFE core.

#### Description

Enables background checking on the cFE core code segment

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_CFECore\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.CfeCoreCSSState](#) set to [CS\\_STATE\\_ENABLED](#)

#### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_CFE\\_CORE\\_CC](#)

Definition at line 278 of file cs\_msgdefs.h.

**10.5.2.16 CS\_ENABLE EEPROM\_CC** #define CS\_ENABLE EEPROM\_CC 14  
Enable checksumming for EEPROM table.

#### Description

Allow the EEPROM table to checksummed in the background

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE EEPROM\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.EepromCSState](#) set to [CS\\_STATE\\_ENABLED](#)

#### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE EEPROM\\_CC](#)

Definition at line 556 of file cs\_msgdefs.h.

**10.5.2.17 CS\_ENABLE\_ENTRY\_EEPROM\_CC** #define CS\_ENABLE\_ENTRY\_EEPROM\_CC 18  
Enable checksumming for an EEPROM entry.

#### Description

Allow the EEPROM entry to checksummed in the background

#### Command Structure

[CS\\_EntryCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_EEPROM\\_ENTRY\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified entry was invalid

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_ENABLE\\_EEPROM\\_INVALID\\_ENTRY\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_ENTRY\\_EEPROM\\_CC](#)

Definition at line 704 of file cs\_msgdefs.h.

**10.5.2.18 CS\_ENABLE\_ENTRY\_MEMORY\_CC** #define CS\_ENABLE\_ENTRY\_MEMORY\_CC 25  
Enable checksumming for a Memory entry.

#### Description

Allow the Memory entry to checksummed in the background

#### Command Structure

[CS\\_EntryCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_MEMORY\\_ENTRY\\_INF\\_EID](#) informational event message will be generated when the command is received

### Error Conditions

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

- Command packet length not as expected
- Command specified entry was invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_ENABLE\\_MEMORY\\_INVALID\\_ENTRY\\_ERR\\_EID](#)

### Criticality

None

### See also

[CS\\_DISABLE\\_ENTRY\\_MEMORY\\_CC](#)

Definition at line 948 of file cs\_msgdefs.h.

**10.5.2.19 CS\_ENABLE\_MEMORY\_CC** `#define CS_ENABLE_MEMORY_CC 21`  
Enable checksumming for Memory table.

### Description

Allow the Memory table to checksummed in the background

### Command Structure

[CS\\_NoArgsCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_MEMORY\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.MemoryCSState](#) set to [CS\\_STATE\\_ENABLED](#)

### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_MEMORY\\_CC](#)

Definition at line 801 of file cs\_msgdefs.h.

### 10.5.2.20 CS\_ENABLE\_NAME\_APP\_CC #define CS\_ENABLE\_NAME\_APP\_CC 38

Enable checksumming for an app.

#### Description

Allow the app to checksummed in the background

#### Command Structure

[CS\\_AppNameCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_APP\\_NAME\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified app name was invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_ENABLE\\_APP\\_UNKNOWN\\_NAME\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_NAME\\_APP\\_CC](#)

Definition at line 1411 of file cs\_msgdefs.h.

**10.5.2.21 CS\_ENABLE\_NAME\_TABLE\_CC** #define CS\_ENABLE\_NAME\_TABLE\_CC 32  
Enable checksumming for a table.

#### Description

Allow the table to checksummed in the background

#### Command Structure

[CS\\_TableNameCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_TABLES\\_NAME\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified table name was invalid

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_ENABLE\\_TABLES\\_UNKNOWN\\_NAME\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_NAME\\_TABLE\\_CC](#)

Definition at line 1199 of file cs\_msgdefs.h.

**10.5.2.22 CS\_ENABLE\_OS\_CC** #define CS\_ENABLE\_OS\_CC 10  
Enable checksumming of OS code segment.

#### Description

Enables background checking on the OS code segment

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_OS\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- [CS\\_HkPacket\\_Payload\\_t.OSCSState](#) set to [CS\\_STATE\\_ENABLED](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_OS\\_CC](#)

Definition at line 418 of file cs\_msgdefs.h.

**10.5.2.23 CS\_ENABLE\_TABLES\_CC** #define CS\_ENABLE\_TABLES\_CC 28  
Enable checksumming for Tables table.

#### Description

Allow the Tables table to checksummed in the background

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ENABLE\\_TABLES\\_INF\\_EID](#) informational event message will be generated when the command is received
- [CS\\_HkPacket\\_Payload\\_t.TablesCSSState](#) set to [CS\\_STATE\\_ENABLED](#)

#### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_DISABLE\\_TABLES\\_CC](#)

Definition at line 1046 of file cs\_msgdefs.h.

**10.5.2.24 CS\_GET\_ENTRY\_ID\_EEPROM\_CC** #define CS\_GET\_ENTRY\_ID\_EEPROM\_CC 20  
Get the Entry ID for a given EEPROM address.

#### Description

Gets the Entry ID of an EEPROM address to use in subsequent commands.

#### Command Structure

[CS\\_GetEntryIDCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_GET\\_ENTRY\\_ID\\_EEPROM\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified address was not found in the table

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_GET\\_ENTRY\\_ID\\_EEPROM\\_NOT\\_FOUND\\_INF\\_EID](#)

#### Criticality

None

Definition at line 769 of file cs\_msgdefs.h.

**10.5.2.25 CS\_GET\_ENTRY\_ID\_MEMORY\_CC** #define CS\_GET\_ENTRY\_ID\_MEMORY\_CC 27  
Get the Entry ID for a given Memory address.

#### Description

Gets the Entry ID of a Memory address to use in subsequent commands.

#### Command Structure

[CS\\_GetEntryIDCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_GET\\_ENTRY\\_ID\\_MEMORY\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- Command specified address was not found in the table

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_GET\\_ENTRY\\_ID\\_MEMORY\\_NOT\\_FOUND\\_INF\\_EID](#)

#### Criticality

None

Definition at line 1014 of file cs\_msgdefs.h.

**10.5.2.26 CS\_NOOP\_CC** #define CS\_NOOP\_CC 0  
Noop.

#### Description

Implements the Noop command that insures the CS task is alive

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_NOOP\\_INF\\_EID](#) informational event message will be generated when the command is received

### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

### Criticality

None

### See also

[CS\\_RESET\\_CC](#)

Definition at line 64 of file `cs_msgdefs.h`.

### **10.5.2.27 CS\_ONE\_SHOT\_CC** #define CS\_ONE\_SHOT\_CC 2

Start One shot calculation.

#### Description

Computes a checksum on the command specified address and size of memory at the command specified rate.  
This command spawns a child task to complete the checksum.

#### Command Structure

[CS\\_OneShotCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_ONESHOT\\_STARTED\\_DBG\\_EID](#) debug event message will be generated when the command is received
- The [CS\\_ONESHOT\\_FINISHED\\_INF\\_EID](#) informational message will be generated when the computation finishes.
- [CS\\_HkPacket\\_Payload\\_t.LastOneShotChecksum](#) will be updated to the new value

### Error Conditions

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

- Command packet length not as expected
- The address and size cannot be validated
- A child task (recompute baseline or one shot ) is already running, precluding starting another. Only one child task is allowed to run at any given time.
- The child task failed to be created

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_ONESHOT\\_MEMVALIDATE\\_ERR\\_EID](#)
- Error specific event message [CS\\_ONESHOT\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_ONESHOT\\_CREATE\\_CHDTASK\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_CANCEL\\_ONE\\_SHOT\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#),  
[CS\\_RECOMPUTE\\_BASELINE\\_EEPROM\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_TA](#),  
[CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#)

Definition at line 147 of file cs\_msgdefs.h.

#### 10.5.2.28 CS\_RECOMPUTE\_BASELINE\_APP\_CC #define CS\_RECOMPUTE\_BASELINE\_APP\_CC 37

Recompute Baseline checksum of an app.

#### Description

Recompute the baseline checksum of the app and use that value as the new baseline. This command spawns a child task to do the recompute.

#### Command Structure

[CS\\_AppNameCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_RECOMPUTE\\_APP\\_STARTED\\_DBG\\_EID](#) debug event message will be generated when the command is received
- The [CS\\_RECOMPUTE\\_FINISH\\_APP\\_INF\\_EID](#) informational event message will be generated when the recompute is finished

#### Error Conditions

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

- Command packet length not as expected
- The command specified app name is invalid
- A child task (recompute baseline or one shot) is already running, precluding starting another. Only one child task is allowed to run at any given time.
- The child task failed to be created by Executive Services (ES)

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_UNKNOWN\\_NAME\\_APP\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_APP\\_CREATE\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_APP\\_CHDTASK\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ONE\\_SHOT\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#),  
[CS\\_RECOMPUTE\\_BASELINE\\_EEPROM\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_TA](#)

Definition at line 1378 of file `cs_msgdefs.h`.

**10.5.2.29 CS\_RECOMPUTE\_BASELINE\_CFE\_CORE\_CC** `#define CS_RECOMPUTE_BASELINE_CFE_CORE_CC 9`  
Recompute Baseline checksum of cFE core.

#### Description

Recomputes the baseline checksum of the cFE core and use the new value as the baseline.

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_RECOMPUTE\\_CFECORE\\_STARTED\\_DBG\\_EID](#) debug event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- A child task (recompute baseline or one shot) is already running, precluding starting another. Only one child task is allowed to run at any given time.
- The child task failed to be created by Executive Services (ES)

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_CFECORE\\_CREATE\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_CFECORE\\_CHDTASK\\_ERR\\_EID](#)

**Criticality**

None

**See also**

[CS\\_ONE\\_SHOT\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_EEPROM\\_CC](#),  
[CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_TABLE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#)

Definition at line 386 of file cs\_msgdefs.h.

### 10.5.2.30 CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC #define CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC 17

Recompute Baseline checksum of EEPROM Entry.

**Description**

Recompute the baseline checksum of the EEPROM table entry and use that value as the new baseline. This command spawns a child task to do the recompute.

**Command Structure**

[CS\\_EntryCmd\\_t](#)

**Command Verification**

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_RECOMPUTE\\_EEPROM\\_STARTED\\_DBG\\_EID](#) debug event message will be generated when the command is received
- The [CS\\_RECOMPUTE\\_FINISH\\_EEPROM\\_MEMORY\\_INF\\_EID](#) informational event message will be generated when the recompute is finished

**Error Conditions**

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

- Command packet length not as expected
- The command specified Entry ID is invalid
- A child task (recompute baseline or one shot) is already running, precluding starting another. Only one child task is allowed to run at any given time.
- The child task failed to be created by Executive Services (ES)

**Evidence of failure may be found in the following telemetry:**

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_INVALID\\_ENTRY\\_EEPROM\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_EEPROM\\_CREATE\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_EEPROM\\_CHDTASK\\_ERR\\_EID](#)

**Criticality**

None

**See also**

[CS\\_ONE\\_SHOT\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#),  
[CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_TABLE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#)

Definition at line 671 of file `cs_msgdefs.h`.

### 10.5.2.31 CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC #define CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC 24

Recompute Baseline checksum of Memory Entry.

**Description**

Recompute the baseline checksum of the Memory table entry and use that value as the new baseline. This command spawns a child task to do the recompute.

**Command Structure**

[CS\\_EntryCmd\\_t](#)

**Command Verification**

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_RECOMPUTE\\_MEMORY\\_STARTED\\_DBG\\_EID](#) debug event message will be generated when the command is received
- The [CS\\_RECOMPUTE\\_FINISH EEPROM\\_MEMORY\\_INF\\_EID](#) informational event message will be generated when the recompute is finished

**Error Conditions**

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

- Command packet length not as expected
- The command specified Entry ID is invalid
- A child task (recompute baseline or one shot) is already running, precluding starting another. Only one child task is allowed to run at any given time.
- The child task failed to be created by Executive Services (ES)

**Evidence of failure may be found in the following telemetry:**

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_INVALID\\_ENTRY\\_MEMORY\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_MEMORY\\_CREATE\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_MEMORY\\_CHDTASK\\_ERR\\_EID](#)

**Criticality**

None

**See also**

[CS\\_ONE\\_SHOT\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#),  
[CS\\_RECOMPUTE\\_BASELINE\\_EEPROM\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_TABLE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#)

Definition at line 915 of file cs\_msgdefs.h.

### 10.5.2.32 CS\_RECOMPUTE\_BASELINE\_OS\_CC #define CS\_RECOMPUTE\_BASELINE\_OS\_CC 13

Recompute Baseline checksum of OS code segment.

**Description**

Recomputes the baseline checksum of the OS code segment and use the new value as the baseline.

**Command Structure**

[CS\\_NoArgsCmd\\_t](#)

**Command Verification**

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_RECOMPUTE\\_OS\\_STARTED\\_DBG\\_EID](#) debug event message will be generated when the command is received

**Error Conditions**

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

- Command packet length not as expected
- A child task (recompute baseline or one shot) is already running, precluding starting another. Only one child task is allowed to run at any given time.
- The child task failed to be created by Executive Services (ES)

**Evidence of failure may be found in the following telemetry:**

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_OS\\_CREATE\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_OS\\_CHDTASK\\_ERR\\_EID](#)

**Criticality**

None

**See also**

[CS\\_ONE\\_SHOT\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_EEPROM\\_CC](#),  
[CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_TABLE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#)

Definition at line 524 of file cs\_msgdefs.h.

**10.5.2.33 CS\_RECOMPUTE\_BASELINE\_TABLE\_CC** #define CS\_RECOMPUTE\_BASELINE\_TABLE\_CC 31  
Recompute Baseline checksum of a table.

#### Description

Recompute the baseline checksum of the table and use that value as the new baseline. This command spawns a child task to do the recompute.

#### Command Structure

[CS\\_TableNameCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_RECOMPUTE\\_TABLES\\_STARTED\\_DBG\\_EID](#) debug event message will be generated when the command is received
- The [CS\\_RECOMPUTE\\_FINISH\\_TABLES\\_INF\\_EID](#) informational event message will be generated when the recompute is finished. However, for the CS Table Definitions Table only, the checksum value will be incorrect. This is because all entries in this table are disabled while being recomputed and disabling the entry for itself modifies the contents of the table being recomputed. Thus, recomputing the CS Tables Definition Table checksum is not recommended.

#### Error Conditions

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

- Command packet length not as expected
- The command specified table name is invalid
- A child task (recompute baseline or one shot) is already running, precluding starting another. Only one child task is allowed to run at any given time.
- The child task failed to be created by Executive Services (ES)

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_UNKNOWN\\_NAME\\_TABLES\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_TABLES\\_CREATE\\_CHDTASK\\_ERR\\_EID](#)
- Error specific event message [CS\\_RECOMPUTE\\_TABLES\\_CHDTASK\\_ERR\\_EID](#)

#### Criticality

None

#### See also

[CS\\_ONE\\_SHOT\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#),  
[CS\\_RECOMPUTE\\_BASELINE EEPROM\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_AP](#)

Definition at line 1166 of file `cs_msgdefs.h`.

**10.5.2.34 CS\_REPORT\_BASELINE\_APP\_CC** #define CS\_REPORT\_BASELINE\_APP\_CC 36  
Report Baseline checksum of an app.

#### Description

Reports the baseline checksum of the app that has already been calculated.

#### Command Structure

[CS\\_AppNameCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_BASELINE\\_APP\\_INF\\_EID](#) informational event message will be generated when the command is received, or
- The [CS\\_NO\\_BASELINE\\_APP\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- The command specified able name is invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_BASELINE\\_INVALID\\_NAME\\_APP\\_ERR\\_EID](#)

#### Criticality

None

Definition at line 1330 of file cs\_msgdefs.h.

**10.5.2.35 CS\_REPORT\_BASELINE\_CFE\_CORE\_CC** #define CS\_REPORT\_BASELINE\_CFE\_CORE\_CC 8  
Report Baseline checksum of cFE core.

#### Description

Reports the baseline checksum of the cFE core that has already been calculated.

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_BASELINE\\_CFECORE\\_INF\\_EID](#) informational event message will be generated when the command is received, or
- The [CS\\_NO\\_BASELINE\\_CFECORE\\_INF\\_EID](#) informational event message will be generated when the command is received

### Error Conditions

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

- Command packet length not as expected

### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

### Criticality

None

Definition at line 343 of file cs\_msgdefs.h.

**10.5.2.36 CS\_REPORT\_BASELINE\_EEPROM\_CC** #define CS\_REPORT\_BASELINE\_EEPROM\_CC 16  
Report Baseline checksum of EEPROM Entry.

### Description

Reports the baseline checksum of the EEPROM table entry that has already been calculated.

### Command Structure

[CS\\_EntryCmd\\_t](#)

### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_BASELINE\\_EEPROM\\_INF\\_EID](#) informational event message will be generated when the command is received, or
- The [CS\\_NO\\_BASELINE\\_EEPROM\\_INF\\_EID](#) informational event message will be generated when the command is received

### Error Conditions

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

- Command packet length not as expected
- The command specified Entry ID is invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_BASELINE\\_INVALID\\_ENTRY\\_EEPROM\\_ERR\\_EID](#)

#### Criticality

None

Definition at line 623 of file cs\_msgdefs.h.

**10.5.2.37 CS\_REPORT\_BASELINE\_MEMORY\_CC** #define CS\_REPORT\_BASELINE\_MEMORY\_CC 23  
Report Baseline checksum of Memory Entry.

#### Description

Reports the baseline checksum of the Memory table entry that has already been calculated.

#### Command Structure

[CS\\_EntryCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_BASELINE\\_MEMORY\\_INF\\_EID](#) informational event message will be generated when the command is received, or
- The [CS\\_NO\\_BASELINE\\_MEMORY\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected
- The command specified Entry ID is invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_BASELINE\\_INVALID\\_ENTRY\\_MEMORY\\_ERR\\_EID](#)

#### Criticality

None

Definition at line 867 of file cs\_msgdefs.h.

**10.5.2.38 CS\_REPORT\_BASELINE\_OS\_CC** #define CS\_REPORT\_BASELINE\_OS\_CC 12  
Report Baseline checksum of OS code segment.

#### Description

Reports the baseline checksum of the OS code segment that has already been calculated.

#### Command Structure

[CS\\_NoArgsCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_BASELINE\\_OS\\_INF\\_EID](#) informational event message will be generated when the command is received, or
- The [CS\\_NO\\_BASELINE\\_OS\\_INF\\_EID](#) informational event message will be generated when the command is received

#### Error Conditions

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

- Command packet length not as expected

#### Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

#### Criticality

None

Definition at line 482 of file cs\_msgdefs.h.

**10.5.2.39 CS\_REPORT\_BASELINE\_TABLE\_CC** #define CS\_REPORT\_BASELINE\_TABLE\_CC 30  
Report Baseline checksum of a table.

#### Description

Reports the baseline checksum of the table that has already been calculated.

#### Command Structure

[CS\\_TableNameCmd\\_t](#)

#### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will increment
- The [CS\\_BASELINE\\_TABLES\\_INF\\_EID](#) informational event message will be generated when the command is received, or
- The [CS\\_NO\\_BASELINE\\_TABLES\\_INF\\_EID](#) informational event message will be generated when the command is received

### Error Conditions

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

- Command packet length not as expected
- The command specified able name is invalid

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)
- Error specific event message [CS\\_BASELINE\\_INVALID\\_NAME\\_TABLES\\_ERR\\_EID](#)

### Criticality

None

Definition at line 1112 of file cs\_msgdefs.h.

#### **10.5.2.40 CS\_RESET\_CC #define CS\_RESET\_CC 1**

Reset Counters.

##### Description

Resets the CS housekeeping counters

##### Command Structure

[CS\\_NoArgsCmd\\_t](#)

##### Command Verification

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

- [CS\\_HkPacket\\_Payload\\_t.CmdCounter](#) will be cleared
- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will be cleared
- The [CS\\_RESET\\_DBG\\_EID](#) informational event message will be generated when the command is executed

### Error Conditions

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

- Command packet length not as expected

Evidence of failure may be found in the following telemetry:

- [CS\\_HkPacket\\_Payload\\_t.CmdErrCounter](#) will increment
- Error specific event message [CS\\_LEN\\_ERR\\_EID](#)

### Criticality

None

##### See also

[CS\\_NOOP\\_CC](#)

Definition at line 96 of file cs\_msgdefs.h.

## 10.6 CFS Checksum Command Message IDs

### Macros

- `#define CS_CMD_MID (0x189F)`  
*CS Command Message ID.*
- `#define CS_SEND_HK_MID (0x18A0)`  
*CS Housekeeping Request Message ID.*
- `#define CS_BACKGROUND_CYCLE_MID (0x18A1)`  
*CS Background Cycle Message ID.*

#### 10.6.1 Detailed Description

#### 10.6.2 Macro Definition Documentation

##### 10.6.2.1 CS\_BACKGROUND\_CYCLE\_MID `#define CS_BACKGROUND_CYCLE_MID (0x18A1)`

CS Background Cycle Message ID.

Definition at line 34 of file cs\_msgids.h.

##### 10.6.2.2 CS\_CMD\_MID `#define CS_CMD_MID (0x189F)`

CS Command Message ID.

Definition at line 32 of file cs\_msgids.h.

##### 10.6.2.3 CS\_SEND\_HK\_MID `#define CS_SEND_HK_MID (0x18A0)`

CS Housekeeping Request Message ID.

Definition at line 33 of file cs\_msgids.h.

## 10.7 CFS Checksum Telemetry Message IDs

### Macros

- #define CS\_HK\_TLM\_MID (0x08A4)  
*CS Housekeeping Telemetry Message ID.*

#### 10.7.1 Detailed Description

#### 10.7.2 Macro Definition Documentation

##### 10.7.2.1 CS\_HK\_TLM\_MID #define CS\_HK\_TLM\_MID (0x08A4)

CS Housekeeping Telemetry Message ID.

Definition at line 43 of file cs\_msgids.h.

## 10.8 CFS Checksum Platform Configuration

### Macros

- `#define CS_DEF_EEPROM_TABLE_FILENAME "/cf/cs_eepromtbl.tbl"`  
*EEPROM File Table – default table filename.*
- `#define CS_DEF_MEMORY_TABLE_FILENAME "/cf/cs_memorytbl.tbl"`  
*Memory Address File Table – default table filename.*
- `#define CS_DEF_TABLES_TABLE_FILENAME "/cf/cs_tablestbl.tbl"`  
*Tables File Table – default table filename.*
- `#define CS_DEF_APP_TABLE_FILENAME "/cf/cs_apptbl.tbl"`  
*Application File Table – default table filename.*
- `#define CS_PIPE_DEPTH (3 * CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT)`  
*Application Pipe Depth.*
- `#define CS_MAX_NUM EEPROM_TABLE_ENTRIES 16`  
*Maximum number of entries in the EEPROM table to checksum.*
- `#define CS_MAX_NUM_MEMORY_TABLE_ENTRIES 16`  
*Maximum number of entries in the Memory table to checksum.*
- `#define CS_MAX_NUM_TABLES_TABLE_ENTRIES 24`  
*Maximum number of tables to checksum.*
- `#define CS_MAX_NUM_APP_TABLE_ENTRIES 24`  
*Maximum number of applications to checksum.*
- `#define CS_DEFAULT_BYTES_PER_CYCLE (1024 * 16)`  
*Default number of bytes to checksum per cycle.*
- `#define CS_CHILD_TASK_PRIORITY 200`  
*CS Child Task Priority.*
- `#define CS_CHILD_TASK_DELAY 1000`  
*Delay between checksumming cycles for child task.*
- `#define CS_STARTUP_TIMEOUT 60000`  
*Timeout for waiting for other apps to start.*
- `#define CS_OSCS_CHECKSUM_STATE CS_STATE_ENABLED`  
*Desired state of the checksumming of OS code segment at power on.*
- `#define CS_CFECORE_CHECKSUM_STATE CS_STATE_ENABLED`  
*Desired state of the checksumming of CFE core checksum at power on.*
- `#define CS_EEPROM_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the EEPROM table at power on.*
- `#define CS_MEMORY_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the Memory table at power on.*
- `#define CS_APPS_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the Applications table at power on.*
- `#define CS_TABLES_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the Tables table at power on.*
- `#define CS_PRESERVE_STATES_ON_PROCESSOR_RESET true`  
*Whether to preserve checksum states on processor reset.*
- `#define CS_CDS_NAME "CS_CDS"`  
*Name of the Critical Data Store Used for CS.*
- `#define CS_MISSION_REV 0`  
*Mission specific version number for CS application.*

### 10.8.1 Detailed Description

### 10.8.2 Macro Definition Documentation

#### 10.8.2.1 CS\_APPS\_TBL\_POWERON\_STATE `#define CS_APPS_TBL_POWERON_STATE CS_STATE_ENABLED`

Desired state of the Applications table at power on.

##### Description:

This determines the default state the EEPROM table should be in at power on

##### Limits:

This can either be CS\_STATE\_ENABLED or CS\_STATE\_DISABLED

Definition at line 269 of file cs\_platform\_cfg.h.

#### 10.8.2.2 CS\_CDS\_NAME `#define CS_CDS_NAME "CS_CDS"`

Name of the Critical Data Store Used for CS.

##### Description:

Name of the Critical Data Store for CS This CDS is used to preserve the checksum enabled states for EEROM, Memory, Apps, Tables tables as well as OS code segment and cFE core states

##### Limits:

Must be a unique string with regards to CDS

Definition at line 307 of file cs\_platform\_cfg.h.

#### 10.8.2.3 CS\_CFECORE\_CHECKSUM\_STATE `#define CS_CFECORE_CHECKSUM_STATE CS_STATE_ENABLED`

Desired state of the checksumming of CFE core checksum at power on.

##### Description:

This determines the default state the checksumming of CFE core should be in at power on

##### Limits:

This can either be CS\_STATE\_ENABLED or CS\_STATE\_DISABLED

Definition at line 233 of file cs\_platform\_cfg.h.

#### 10.8.2.4 CS\_CHILD\_TASK\_DELAY `#define CS_CHILD_TASK_DELAY 1000`

Delay between checksumming cycles for child task.

##### Description:

CS child tasks perform checksum cycles like the main App. Since the child tasks aren't scheduled, there needs to be some other mechanism to prevent it from hogging the CPU. This parameter specifies the number of milliseconds to delay in between cycles.

##### Limits:

CS does not place limits on this parameter. It is intended to be configurable to prevent the child task from hogging the CPU

Definition at line 192 of file cs\_platform\_cfg.h.

**10.8.2.5 CS\_CHILD\_TASK\_PRIORITY** #define CS\_CHILD\_TASK\_PRIORITY 200  
CS Child Task Priority.

**Description:**

Priority of child tasks created by CS. 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

Definition at line 176 of file cs\_platform\_cfg.h.

**10.8.2.6 CS\_DEF\_APP\_TABLE\_FILENAME** #define CS\_DEF\_APP\_TABLE\_FILENAME "/cf/cs\_apptbl.tbl"  
Application File Table – default table filename.

**Description:**

This parameter defines the default filename for the Application List File Table.

**Limits:**

The string length (including string terminator) cannot exceed [OS\\_MAX\\_PATH\\_LEN](#). (limit is not verified)

Definition at line 85 of file cs\_platform\_cfg.h.

**10.8.2.7 CS\_DEF\_EEPROM\_TABLE\_FILENAME** #define CS\_DEF\_EEPROM\_TABLE\_FILENAME "/cf/cs\_eepromtbl.tbl"  
EEPROM File Table – default table filename.

**Description:**

This parameter defines the default filename for the EEPROM addresses File Table.

**Limits:**

The string length (including string terminator) cannot exceed [OS\\_MAX\\_PATH\\_LEN](#). (limit is not verified)

Definition at line 46 of file cs\_platform\_cfg.h.

**10.8.2.8 CS\_DEF\_MEMORY\_TABLE\_FILENAME** #define CS\_DEF\_MEMORY\_TABLE\_FILENAME "/cf/cs\_memorytbl.tbl"  
Memory Address File Table – default table filename.

**Description:**

This parameter defines the default filename for the Memory addresses File Table.

**Limits:**

The string length (including string terminator) cannot exceed [OS\\_MAX\\_PATH\\_LEN](#). (limit is not verified)

Definition at line 59 of file cs\_platform\_cfg.h.

**10.8.2.9 CS\_DEF\_TABLES\_TABLE\_FILENAME** #define CS\_DEF\_TABLES\_TABLE\_FILENAME "/cf/cs\_tablestbl.  
tbl"

Tables File Table – default table filename.

**Description:**

This parameter defines the default filename for the Tables List File Table.

**Limits:**

The string length (including string terminator) cannot exceed [OS\\_MAX\\_PATH\\_LEN](#). (limit is not verified)

Definition at line 72 of file cs\_platform\_cfg.h.

**10.8.2.10 CS\_DEFAULT\_BYTES\_PER\_CYCLE** #define CS\_DEFAULT\_BYTES\_PER\_CYCLE (1024 \* 16)

Default number of bytes to checksum per cycle.

**Description:**

The default number of bytes that are checksummed in a single CS cycle

**Limits:**

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. Note that "0" is a valid value, and will result in a checksum of 0.

Definition at line 164 of file cs\_platform\_cfg.h.

**10.8.2.11 CS\_EEPROM\_TBL\_POWERON\_STATE** #define CS\_EEPROM\_TBL\_POWERON\_STATE [CS\\_STATE\\_ENABLED](#)

Desired state of the EEPROM table at power on.

**Description:**

This determines the default state the EEPROM table should be in at power on

**Limits:**

This can either be [CS\\_STATE\\_ENABLED](#) or [CS\\_STATE\\_DISABLED](#)

Definition at line 245 of file cs\_platform\_cfg.h.

**10.8.2.12 CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES** #define CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES 24

Maximum number of applications to checksum.

**Description:**

Maximum number of entries in the table of applications to checksum

**Limits:**

This parameter is limited by the maximum number of applications allowed in the system. This parameter is limited to [CFE\\_PLATFORM\\_ES\\_MAX\\_APPLICATIONS](#)

Definition at line 150 of file cs\_platform\_cfg.h.

**10.8.2.13 CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES** #define CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES 16  
Maximum number of entries in the EEPROM table to checksum.

**Description:**

Maximum number of entries that can be in the table of EEPROM areas to checksum.

**Limits:**

This parameter is limited by the uint16 datatype that defines it. This parameter is limited to 65535.

Definition at line 113 of file cs\_platform\_cfg.h.

**10.8.2.14 CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES** #define CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES 16  
Maximum number of entries in the Memory table to checksum.

**Description:**

Maximum number of entries that can be in the table of Memory areas to checksum.

**Limits:**

This parameter is limited by the uint16 datatype that defines it. This parameter is limited to 65535.

Definition at line 126 of file cs\_platform\_cfg.h.

**10.8.2.15 CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES** #define CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES 24  
Maximum number of tables to checksum.

**Description:**

Maximum number of entries in the table of tables to checksum

**Limits:**

This parameter is limited by the maximum number of tables allowed in the system. This parameter is limited to [CFE\\_PLATFORM\\_TBL\\_MAX\\_NUM\\_TABLES](#)

Definition at line 138 of file cs\_platform\_cfg.h.

**10.8.2.16 CS\_MEMORY\_TBL\_POWERON\_STATE** #define CS\_MEMORY\_TBL\_POWERON\_STATE [CS\\_STATE\\_ENABLED](#)  
Desired state of the Memory table at power on.

**Description:**

This determines the default state the Memory table should be in at power on

**Limits:**

This can either be CS\_STATE\_ENABLED or CS\_STATE\_DISABLED

Definition at line 257 of file cs\_platform\_cfg.h.

**10.8.2.17 CS\_MISSION\_REV #define CS\_MISSION\_REV 0**

Mission specific version number for CS 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 "cs\_version.h".

**Limits:**

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

Definition at line 323 of file cs\_platform\_cfg.h.

**10.8.2.18 CS\_OSCS\_CHECKSUM\_STATE #define CS\_OSCS\_CHECKSUM\_STATE CS\_STATE\_ENABLED**

Desired state of the checksumming of OS code segment at power on.

**Description:**

This determines the default state the checksumming of OS code segment should be in at power on

**Limits:**

This can either be CS\_STATE\_ENABLED or CS\_STATE\_DISABLED

Definition at line 221 of file cs\_platform\_cfg.h.

**10.8.2.19 CS\_PIPE\_DEPTH #define CS\_PIPE\_DEPTH (3 \* CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT)**

Application Pipe Depth.

**Description:**

This parameter defines the depth of the CS input pipe. The depth should be deep enough to accommodate all of the DS command packets and all of the subscribed telemetry packets that might be generated by applications with a priority higher than the CS application.

**Limits:**

The value must be greater than zero

Definition at line 100 of file cs\_platform\_cfg.h.

**10.8.2.20 CS\_PRESERVE\_STATES\_ON\_PROCESSOR\_RESET #define CS\_PRESERVE\_STATES\_ON\_PROCESSOR\_RESET true**

Whether to preserve checksum states on processor reset.

**Description:**

This determines whether to preserve checksum enabled/disabled states on processor reset

**Limits:**

None

Definition at line 293 of file cs\_platform\_cfg.h.

**10.8.2.21 CS\_STARTUP\_TIMEOUT** #define CS\_STARTUP\_TIMEOUT 60000  
Timeout for waiting for other apps to start.

**Description:**

CS waits for all of the other applications that are listed in the startup script to start before entering its main loop. If not all of those apps start, CS can pend indefinitely without a timeout. Once CS waits this amount of time ( in milliseconds) it will start regardless of the status of the rest of the apps in the startup script.

**Limits:**

CS does not place limits on this parameter. It is intended to be configurable to allow enough time for all apps to start.

Definition at line 209 of file cs\_platform\_cfg.h.

**10.8.2.22 CS\_TABLES\_TBL\_POWERON\_STATE** #define CS\_TABLES\_TBL\_POWERON\_STATE CS\_STATE\_ENABLED  
Desired state of the Tables table at power on.

**Description:**

This determines the default state the EEPROM table should be in at power on

**Limits:**

This can either be CS\_STATE\_ENABLED or CS\_STATE\_DISABLED

Definition at line 281 of file cs\_platform\_cfg.h.

## 10.9 CFS Checksum Version

### Version Numbers

#### Macros

- `#define CS_MAJOR_VERSION 2`  
*Major version number.*
- `#define CS_MINOR_VERSION 5`  
*Minor version number.*
- `#define CS_REVISION 99`  
*Revision number.*

#### 10.9.1 Detailed Description

##### Version Numbers

#### 10.9.2 Macro Definition Documentation

##### 10.9.2.1 CS\_MAJOR\_VERSION `#define CS_MAJOR_VERSION 2`

Major version number.

Definition at line 33 of file cs\_version.h.

##### 10.9.2.2 CS\_MINOR\_VERSION `#define CS_MINOR_VERSION 5`

Minor version number.

Definition at line 34 of file cs\_version.h.

##### 10.9.2.3 CS\_REVISION `#define CS_REVISION 99`

Revision number.

Definition at line 35 of file cs\_version.h.

## 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\_VALIDATION\_FAILURE ((int32)0xc8000007)  
*Request or input value failed basic structural validation.*
- #define CFE\_STATUS\_RANGE\_ERROR ((int32)0xc8000008)  
*Request or input value is out of range.*
- #define CFE\_STATUS\_INCORRECT\_STATE ((int32)0xc8000009)  
*Cannot process request at this time.*
- #define CFE\_STATUS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc800ffff)  
*Not Implemented.*
- #define CFE\_EVS\_UNKNOWN\_FILTER ((CFE\_Status\_t)0xc2000001)  
*Unknown Filter.*
- #define CFE\_EVS\_APP\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000002)  
*Application Not Registered.*
- #define CFE\_EVS\_APP\_ILLEGAL\_APP\_ID ((CFE\_Status\_t)0xc2000003)  
*Illegal Application ID.*
- #define CFE\_EVS\_APP\_FILTER\_OVERLOAD ((CFE\_Status\_t)0xc2000004)  
*Application Filter Overload.*
- #define CFE\_EVS\_RESET\_AREA\_POINTER ((CFE\_Status\_t)0xc2000005)  
*Reset Area Pointer Failure.*
- #define CFE\_EVS\_EVT\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000006)  
*Event Not Registered.*
- #define CFE\_EVS\_FILE\_WRITE\_ERROR ((CFE\_Status\_t)0xc2000007)  
*File Write Error.*
- #define CFE\_EVS\_INVALID\_PARAMETER ((CFE\_Status\_t)0xc2000008)  
*Invalid Pointer.*
- #define CFE\_EVS\_APP\_SQUELCHED ((CFE\_Status\_t)0xc2000009)  
*Event squelched.*
- #define CFE\_EVS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc200ffff)  
*Not Implemented.*
- #define CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID ((CFE\_Status\_t)0xc4000001)  
*Resource ID is not valid.*

- #define CFE\_ES\_ERR\_NAME\_NOT\_FOUND ((CFE\_Status\_t)0xc4000002)  
*Resource Name Error.*
- #define CFE\_ES\_ERR\_APP\_CREATE ((CFE\_Status\_t)0xc4000004)  
*Application Create Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_CREATE ((CFE\_Status\_t)0xc4000005)  
*Child Task Create Error.*
- #define CFE\_ES\_ERR\_SYS\_LOG\_FULL ((CFE\_Status\_t)0xc4000006)  
*System Log Full.*
- #define CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE ((CFE\_Status\_t)0xc4000008)  
*Memory Block Size Error.*
- #define CFE\_ES\_ERR\_LOAD\_LIB ((CFE\_Status\_t)0xc4000009)  
*Load Library Error.*
- #define CFE\_ES\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc400000a)  
*Bad Argument.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER ((CFE\_Status\_t)0xc400000b)  
*Child Task Register Error.*
- #define CFE\_ES\_CDS\_ALREADY\_EXISTS ((CFE\_Status\_t)0x4400000d)  
*CDS Already Exists.*
- #define CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY ((CFE\_Status\_t)0xc400000e)  
*CDS Insufficient Memory.*
- #define CFE\_ES\_CDS\_INVALID\_NAME ((CFE\_Status\_t)0xc400000f)  
*CDS Invalid Name.*
- #define CFE\_ES\_CDS\_INVALID\_SIZE ((CFE\_Status\_t)0xc4000010)  
*CDS Invalid Size.*
- #define CFE\_ES\_CDS\_INVALID ((CFE\_Status\_t)0xc4000012)  
*CDS Invalid.*
- #define CFE\_ES\_CDS\_ACCESS\_ERROR ((CFE\_Status\_t)0xc4000013)  
*CDS Access Error.*
- #define CFE\_ES\_FILE\_IO\_ERR ((CFE\_Status\_t)0xc4000014)  
*File IO Error.*
- #define CFE\_ES\_RST\_ACCESS\_ERR ((CFE\_Status\_t)0xc4000015)  
*Reset Area Access Error.*
- #define CFE\_ES\_ERR\_APP\_REGISTER ((CFE\_Status\_t)0xc4000017)  
*Application Register Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE ((CFE\_Status\_t)0xc4000018)  
*Child Task Delete Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK ((CFE\_Status\_t)0xc4000019)  
*Child Task Delete Passed Main Task.*
- #define CFE\_ES\_CDS\_BLOCK\_CRC\_ERR ((CFE\_Status\_t)0xc400001A)  
*CDS Block CRC Error.*
- #define CFE\_ES\_MUT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001B)  
*Mutex Semaphore Delete Error.*
- #define CFE\_ES\_BIN\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001C)  
*Binary Semaphore Delete Error.*
- #define CFE\_ES\_COUNT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001D)  
*Counting Semaphore Delete Error.*
- #define CFE\_ES\_QUEUE\_DELETE\_ERR ((CFE\_Status\_t)0xc400001E)

- #define CFE\_ES\_FILE\_CLOSE\_ERR ((CFE\_Status\_t)0xc400001F)  
*File Close Error.*
- #define CFE\_ES\_CDS\_WRONG\_TYPE\_ERR ((CFE\_Status\_t)0xc4000020)  
*CDS Wrong Type Error.*
- #define CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR ((CFE\_Status\_t)0xc4000022)  
*CDS Owner Active Error.*
- #define CFE\_ES\_APP\_CLEANUP\_ERR ((CFE\_Status\_t)0xc4000023)  
*Application Cleanup Error.*
- #define CFE\_ES\_TIMER\_DELETE\_ERR ((CFE\_Status\_t)0xc4000024)  
*Timer Delete Error.*
- #define CFE\_ES\_BUFFER\_NOT\_IN\_POOL ((CFE\_Status\_t)0xc4000025)  
*Buffer Not In Pool.*
- #define CFE\_ES\_TASK\_DELETE\_ERR ((CFE\_Status\_t)0xc4000026)  
*Task Delete Error.*
- #define CFE\_ES\_OPERATION\_TIMED\_OUT ((CFE\_Status\_t)0xc4000027)  
*Operation Timed Out.*
- #define CFE\_ES\_LIB\_ALREADY\_LOADED ((CFE\_Status\_t)0x44000028)  
*Library Already Loaded.*
- #define CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED ((CFE\_Status\_t)0x44000029)  
*System Log Message Truncated.*
- #define CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE ((CFE\_Status\_t)0xc400002B)  
*Resource ID is not available.*
- #define CFE\_ES\_POOL\_BLOCK\_INVALID ((CFE\_Status\_t)0xc400002C)  
*Invalid pool block.*
- #define CFE\_ES\_ERR\_DUPLICATE\_NAME ((CFE\_Status\_t)0xc400002E)  
*Duplicate Name Error.*
- #define CFE\_ES\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc400ffff)  
*Not Implemented.*
- #define CFE\_FS\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc6000001)  
*Bad Argument.*
- #define CFE\_FS\_INVALID\_PATH ((CFE\_Status\_t)0xc6000002)  
*Invalid Path.*
- #define CFE\_FS\_FNAME\_TOO\_LONG ((CFE\_Status\_t)0xc6000003)  
*Filename Too Long.*
- #define CFE\_FS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc600ffff)  
*Not Implemented.*
- #define CFE\_SB\_TIME\_OUT ((CFE\_Status\_t)0xca000001)  
*Time Out.*
- #define CFE\_SB\_NO\_MESSAGE ((CFE\_Status\_t)0xca000002)  
*No Message.*
- #define CFE\_SB\_BAD\_ARGUMENT ((CFE\_Status\_t)0xca000003)  
*Bad Argument.*
- #define CFE\_SB\_MAX\_PIPES\_MET ((CFE\_Status\_t)0xca000004)  
*Max Pipes Met.*
- #define CFE\_SB\_PIPE\_CR\_ERR ((CFE\_Status\_t)0xca000005)  
*Pipe Create Error.*

- #define CFE\_SB\_PIPE\_RD\_ERR ((CFE\_Status\_t)0xca000006)  
*Pipe Read Error.*
- #define CFE\_SB\_MSG\_TOO\_BIG ((CFE\_Status\_t)0xca000007)  
*Message Too Big.*
- #define CFE\_SB\_BUF\_ALOC\_ERR ((CFE\_Status\_t)0xca000008)  
*Buffer Allocation Error.*
- #define CFE\_SB\_MAX\_MSGS\_MET ((CFE\_Status\_t)0xca000009)  
*Max Messages Met.*
- #define CFE\_SB\_MAX\_DESTS\_MET ((CFE\_Status\_t)0xca00000a)  
*Max Destinations Met.*
- #define CFE\_SB\_INTERNAL\_ERR ((CFE\_Status\_t)0xca00000c)  
*Internal Error.*
- #define CFE\_SB\_WRONG\_MSG\_TYPE ((CFE\_Status\_t)0xca00000d)  
*Wrong Message Type.*
- #define CFE\_SB\_BUFFER\_INVALID ((CFE\_Status\_t)0xca00000e)  
*Buffer Invalid.*
- #define CFE\_SB\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xca00ffff)  
*Not Implemented.*
- #define CFE\_TBL\_ERR\_INVALID\_HANDLE ((CFE\_Status\_t)0xcc000001)  
*Invalid Handle.*
- #define CFE\_TBL\_ERR\_INVALID\_NAME ((CFE\_Status\_t)0xcc000002)  
*Invalid Name.*
- #define CFE\_TBL\_ERR\_INVALID\_SIZE ((CFE\_Status\_t)0xcc000003)  
*Invalid Size.*
- #define CFE\_TBL\_INFO\_UPDATE\_PENDING ((CFE\_Status\_t)0x4c000004)  
*Update Pending.*
- #define CFE\_TBL\_ERR\_NEVER\_LOADED ((CFE\_Status\_t)0xcc000005)  
*Never Loaded.*
- #define CFE\_TBL\_ERR\_REGISTRY\_FULL ((CFE\_Status\_t)0xcc000006)  
*Registry Full.*
- #define CFE\_TBL\_WARN\_DUPLICATE ((CFE\_Status\_t)0x4c000007)  
*Duplicate Warning.*
- #define CFE\_TBL\_ERR\_NO\_ACCESS ((CFE\_Status\_t)0xcc000008)  
*No Access.*
- #define CFE\_TBL\_ERR\_UNREGISTERED ((CFE\_Status\_t)0xcc000009)  
*Unregistered.*
- #define CFE\_TBL\_ERR\_HANDLES\_FULL ((CFE\_Status\_t)0xcc00000B)  
*Handles Full.*
- #define CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE ((CFE\_Status\_t)0xcc00000C)  
*Duplicate Table With Different Size.*
- #define CFE\_TBL\_ERR\_DUPLICATE\_NOT OWNED ((CFE\_Status\_t)0xcc00000D)  
*Duplicate Table And Not Owned.*
- #define CFE\_TBL\_INFO\_UPDATED ((CFE\_Status\_t)0x4c00000E)  
*Updated.*
- #define CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL ((CFE\_Status\_t)0xcc00000F)  
*No Buffer Available.*
- #define CFE\_TBL\_ERR\_DUMP\_ONLY ((CFE\_Status\_t)0xcc000010)

- Dump Only Error.*
- #define CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE ((CFE\_Status\_t)0xcc000011)  
*Illegal Source Type.*
  - #define CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS ((CFE\_Status\_t)0xcc000012)  
*Load In Progress.*
  - #define CFE\_TBL\_ERR\_FILE\_TOO\_LARGE ((CFE\_Status\_t)0xcc000014)  
*File Too Large.*
  - #define CFE\_TBL\_WARN\_SHORT\_FILE ((CFE\_Status\_t)0x4c000015)  
*Short File Warning.*
  - #define CFE\_TBL\_ERR\_BAD\_CONTENT\_ID ((CFE\_Status\_t)0xcc000016)  
*Bad Content ID.*
  - #define CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING ((CFE\_Status\_t)0x4c000017)  
*No Update Pending.*
  - #define CFE\_TBL\_INFO\_TABLE\_LOCKED ((CFE\_Status\_t)0x4c000018)  
*Table Locked.*
  - #define CFE\_TBL\_INFO\_VALIDATION\_PENDING ((CFE\_Status\_t)0x4c000019)
  - #define CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING ((CFE\_Status\_t)0x4c00001A)
  - #define CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID ((CFE\_Status\_t)0xcc00001B)  
*Bad Subtype ID.*
  - #define CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT ((CFE\_Status\_t)0xcc00001C)  
*File Size Inconsistent.*
  - #define CFE\_TBL\_ERR\_NO\_STD\_HEADER ((CFE\_Status\_t)0xcc00001D)  
*No Standard Header.*
  - #define CFE\_TBL\_ERR\_NO\_TBL\_HEADER ((CFE\_Status\_t)0xcc00001E)  
*No Table Header.*
  - #define CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG ((CFE\_Status\_t)0xcc00001F)  
*Filename Too Long.*
  - #define CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE ((CFE\_Status\_t)0xcc000020)  
*File For Wrong Table.*
  - #define CFE\_TBL\_ERR\_LOAD\_INCOMPLETE ((CFE\_Status\_t)0xcc000021)  
*Load Incomplete.*
  - #define CFE\_TBL\_WARN\_PARTIAL\_LOAD ((CFE\_Status\_t)0x4c000022)  
*Partial Load Warning.*
  - #define CFE\_TBL\_ERR\_PARTIAL\_LOAD ((CFE\_Status\_t)0xcc000023)  
*Partial Load Error.*
  - #define CFE\_TBL\_INFO\_DUMP\_PENDING ((CFE\_Status\_t)0x4c000024)  
*Dump Pending.*
  - #define CFE\_TBL\_ERR\_INVALID\_OPTIONS ((CFE\_Status\_t)0xcc000025)  
*Invalid Options.*
  - #define CFE\_TBL\_WARN\_NOT\_CRITICAL ((CFE\_Status\_t)0x4c000026)  
*Not Critical Warning.*
  - #define CFE\_TBL\_INFO\_RECOVERED\_TBL ((CFE\_Status\_t)0x4c000027)  
*Recovered Table.*
  - #define CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID ((CFE\_Status\_t)0xcc000028)  
*Bad Spacecraft ID.*
  - #define CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID ((CFE\_Status\_t)0xcc000029)  
*Bad Processor ID.*

- #define CFE\_TBL\_MESSAGE\_ERROR ((CFE\_Status\_t)0xcc00002a)  
*Message Error.*
- #define CFE\_TBL\_ERR\_SHORT\_FILE ((CFE\_Status\_t)0xcc00002b)
- #define CFE\_TBL\_ERR\_ACCESS ((CFE\_Status\_t)0xcc00002c)
- #define CFE\_TBL\_BAD\_ARGUMENT ((CFE\_Status\_t)0xcc00002d)  
*Bad Argument.*
- #define CFE\_TBL\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xcc00ffff)  
*Not Implemented.*
- #define CFE\_TIME\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xce00ffff)  
*Not Implemented.*
- #define CFE\_TIME\_INTERNAL\_ONLY ((CFE\_Status\_t)0xce000001)  
*Internal Only.*
- #define CFE\_TIME\_OUT\_OF\_RANGE ((CFE\_Status\_t)0xce000002)  
*Out Of Range.*
- #define CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS ((CFE\_Status\_t)0xce000003)  
*Too Many Sync Callbacks.*
- #define CFE\_TIME\_CALLBACK\_NOT\_REGISTERED ((CFE\_Status\_t)0xce000004)  
*Callback Not Registered.*
- #define CFE\_TIME\_BAD\_ARGUMENT ((CFE\_Status\_t)0xce000005)  
*Bad Argument.*

#### 10.10.1 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 588 of file cfe\_error.h.

##### 10.10.2.2 CFE\_ES\_BAD\_ARGUMENT #define CFE\_ES\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc40000a)

Bad Argument.

Bad parameter passed into an ES API.

Definition at line 399 of file cfe\_error.h.

##### 10.10.2.3 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 527 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 605 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 458 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 415 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 509 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 424 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 450 of file cfe\_error.h.

**10.10.2.10 CFE\_ES\_CDS\_INVALID\_NAME** #define CFE\_ES\_CDS\_INVALID\_NAME (([CFE\\_Status\\_t](#))0xc400000f)  
CDS Invalid Name.

The Application is requesting a CDS Block with an invalid ASCII string name. Either the name is too long (> [CFE\\_MISSION\\_ES\\_CDS\\_MAX\\_NAME\\_LENGTH](#)) or was an empty string.

Definition at line 433 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 442 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 575 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 564 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 536 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 358 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 482 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 366 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 490 of file [cfe\\_error.h](#).

**10.10.2.19 CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK** #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_M←  
AIN\_TASK (([CFE\\_Status\\_t](#))0xc4000019)

Child Task Delete Passed Main Task.

There was an attempt to delete a cFE App Main Task with the [CFE\\_ES\\_DeleteChildTask](#) API.

Definition at line 499 of file [cfe\\_error.h](#).

**10.10.2.20 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 407 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 668 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 391 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 383 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 350 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](#))0xc4000000)

Resource ID is not valid.

This error indicates that the passed in resource identifier (App ID, Lib ID, Counter ID, etc) did not validate.

Definition at line 342 of file cfe\_error.h.

**10.10.2.26 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 375 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 640 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 554 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 466 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 631 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 518 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)0xc400001C)  
Resource ID is not available.

This error indicates that the maximum resource identifiers (App ID, Lib ID, Counter ID, etc) has already been reached and a new ID cannot be allocated.

Definition at line 650 of file cfe\_error.h.

**10.10.2.33 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 679 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 622 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 660 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 545 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 474 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 614 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 597 of file [cfe\\_error.h](#).

**10.10.2.40 CFE\_EVS\_APP\_FILTER\_OVERLOAD** #define CFE\_EVS\_APP\_FILTER\_OVERLOAD (([CFE\\_Status\\_t](#))0xc2000004)  
Application Filter Overload.

Number of Application event filters input upon registration is greater than [CFE\\_PLATFORM\\_EVS\\_MAX\\_EVENT\\_FILTERS](#)

Definition at line 276 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 267 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 258 of file [cfe\\_error.h](#).

**10.10.2.43 CFE\_EVS\_APP\_SQUELCHED** #define CFE\_EVS\_APP\_SQUELCHED (([CFE\\_Status\\_t](#))0xc2000009)  
Event squelched.

Event squelched due to being sent at too high a rate

Definition at line 318 of file [cfe\\_error.h](#).

**10.10.2.44 CFE\_EVS\_EVT\_NOT\_REGISTERED** #define CFE\_EVS\_EVT\_NOT\_REGISTERED (([CFE\\_Status\\_t](#))0xc2000006)  
Event Not Registered.

[CFE\\_EVS\\_ResetFilter](#) EventID argument was not found in any event filter registered by the calling application.

Definition at line 294 of file [cfe\\_error.h](#).

**10.10.2.45 CFE\_EVS\_FILE\_WRITE\_ERROR** #define CFE\_EVS\_FILE\_WRITE\_ERROR (([CFE\\_Status\\_t](#))0xc2000007)  
File Write Error.

A file write error occurred while processing an EVS command

Definition at line 302 of file [cfe\\_error.h](#).

**10.10.2.46 CFE\_EVS\_INVALID\_PARAMETER** #define CFE\_EVS\_INVALID\_PARAMETER (([CFE\\_Status\\_t](#))0xc2000008)  
Invalid Pointer.

Invalid parameter supplied to EVS command

Definition at line 310 of file [cfe\\_error.h](#).

**10.10.2.47 CFE\_EVS\_NOT\_IMPLEMENTED** #define CFE\_EVS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc200ffff)  
Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 329 of file cfe\_error.h.

**10.10.2.48 CFE\_EVS\_RESET\_AREA\_POINTER** #define CFE\_EVS\_RESET\_AREA\_POINTER ((CFE\_Status\_t)0xc2000005)  
Reset Area Pointer Failure.

Could not get pointer to the ES Reset area, so we could not get the pointer to the EVS Log.

Definition at line 285 of file cfe\_error.h.

**10.10.2.49 CFE\_EVS\_UNKNOWN\_FILTER** #define CFE\_EVS\_UNKNOWN\_FILTER ((CFE\_Status\_t)0xc2000001)  
Unknown Filter.

[CFE\\_EVS\\_Register](#) FilterScheme parameter was illegal

Definition at line 250 of file cfe\_error.h.

**10.10.2.50 CFE\_FS\_BAD\_ARGUMENT** #define CFE\_FS\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc6000001)  
Bad Argument.

A parameter given by a caller to a File Services API did not pass validation checks.

Definition at line 692 of file cfe\_error.h.

**10.10.2.51 CFE\_FS\_FNAME\_TOO\_LONG** #define CFE\_FS\_FNAME\_TOO\_LONG ((CFE\_Status\_t)0xc6000003)  
Filename Too Long.

FS filename string is too long

Definition at line 708 of file cfe\_error.h.

**10.10.2.52 CFE\_FS\_INVALID\_PATH** #define CFE\_FS\_INVALID\_PATH ((CFE\_Status\_t)0xc6000002)  
Invalid Path.

FS was unable to extract a filename from a path string

Definition at line 700 of file cfe\_error.h.

**10.10.2.53 CFE\_FS\_NOT\_IMPLEMENTED** #define CFE\_FS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc600ffff)  
Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 719 of file cfe\_error.h.

**10.10.2.54 CFE\_SB\_BAD\_ARGUMENT** #define CFE\_SB\_BAD\_ARGUMENT ((CFE\_Status\_t)0xca000003)  
Bad Argument.

A parameter given by a caller to a Software Bus API did not pass validation checks.

Definition at line 750 of file cfe\_error.h.

**10.10.2.55 CFE\_SB\_BUF\_ALOC\_ERR** #define CFE\_SB\_BUF\_ALOC\_ERR (([CFE\\_Status\\_t](#))0xca000008)

Buffer Allocation Error.

Returned when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter [CFE\\_PLATFORM\\_SB\\_BUF\\_MEMORY\\_BYTES](#) specified in the [cfe\\_platform\\_cfg.h](#) file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Definition at line 808 of file [cfe\\_error.h](#).

**10.10.2.56 CFE\_SB\_BUFFER\_INVALID** #define CFE\_SB\_BUFFER\_INVALID (([CFE\\_Status\\_t](#))0xca00000e)

Buffer Invalid.

This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Definition at line 859 of file [cfe\\_error.h](#).

**10.10.2.57 CFE\_SB\_INTERNAL\_ERR** #define CFE\_SB\_INTERNAL\_ERR (([CFE\\_Status\\_t](#))0xca00000c)

Internal Error.

This error code will be returned by the [CFE\\_SB\\_Subscribe](#) API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Definition at line 840 of file [cfe\\_error.h](#).

**10.10.2.58 CFE\_SB\_MAX\_DESTS\_MET** #define CFE\_SB\_MAX\_DESTS\_MET (([CFE\\_Status\\_t](#))0xca00000a)

Max Destinations Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter [CFE\\_PLATFORM\\_SB\\_MAX\\_DEST\\_PER\\_PKT](#).

Definition at line 830 of file [cfe\\_error.h](#).

**10.10.2.59 CFE\_SB\_MAX\_MSGS\_MET** #define CFE\_SB\_MAX\_MSGS\_MET (([CFE\\_Status\\_t](#))0xca000009)

Max Messages Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate another unique message ID because the platform configuration parameter [CFE\\_PLATFORM\\_SB\\_MAX\\_MSG\\_IDS](#) has been met.

Definition at line 818 of file [cfe\\_error.h](#).

**10.10.2.60 CFE\_SB\_MAX\_PIPES\_MET** #define CFE\_SB\_MAX\_PIPES\_MET (([CFE\\_Status\\_t](#))0xca000004)

Max Pipes Met.

This error code will be returned from [CFE\\_SB\\_CreatePipe](#) when the SB cannot accommodate the request to create a pipe because the maximum number of pipes ([CFE\\_PLATFORM\\_SB\\_MAX\\_PIPES](#)) are in use. This configuration parameter is defined in the [cfe\\_platform\\_cfg.h](#) file.

Definition at line 761 of file [cfe\\_error.h](#).

**10.10.2.61 CFE\_SB\_MSG\_TOO\_BIG** #define CFE\_SB\_MSG\_TOO\_BIG (([CFE\\_Status\\_t](#))0xca000007)

Message Too Big.

The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter [CFE\\_MISSION\\_SB\\_MAX\\_SB\\_MSG\\_SIZE](#) in [cfe\\_mission\\_cfg.h](#)

Definition at line 795 of file [cfe\\_error.h](#).

**10.10.2.62 CFE\_SB\_NO\_MESSAGE** #define CFE\_SB\_NO\_MESSAGE ((CFE\_Status\_t)0xca000002)

No Message.

When "Polling" a pipe for a message in [CFE\\_SB\\_ReceiveBuffer](#), this return value indicates that there was not a message on the pipe.

Definition at line 741 of file cfe\_error.h.

**10.10.2.63 CFE\_SB\_NOT\_IMPLEMENTED** #define CFE\_SB\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xca00ffff)

Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 870 of file cfe\_error.h.

**10.10.2.64 CFE\_SB\_PIPE\_CR\_ERR** #define CFE\_SB\_PIPE\_CR\_ERR ((CFE\_Status\_t)0xca000005)

Pipe Create Error.

The maximum number of queues([OS\\_MAX\\_QUEUES](#)) are in use. Or possibly a lower level problem with creating the underlying queue has occurred such as a lack of memory. If the latter is the problem, the status code displayed in the event must be tracked.

Definition at line 772 of file cfe\_error.h.

**10.10.2.65 CFE\_SB\_PIPE\_RD\_ERR** #define CFE\_SB\_PIPE\_RD\_ERR ((CFE\_Status\_t)0xca000006)

Pipe Read Error.

This return value indicates an error at the Queue read level. This error typically cannot be corrected by the caller. Some possible causes are: queue was not properly initialized or created, the number of bytes read from the queue was not the number of bytes requested in the read. The queue id is invalid. Similar errors regarding the pipe will be caught by higher level code in the Software Bus.

Definition at line 785 of file cfe\_error.h.

**10.10.2.66 CFE\_SB\_TIME\_OUT** #define CFE\_SB\_TIME\_OUT ((CFE\_Status\_t)0xca000001)

Time Out.

In [CFE\\_SB\\_ReceiveBuffer](#), this return value indicates that a packet has not been received in the time given in the "timeout" parameter.

Definition at line 732 of file cfe\_error.h.

**10.10.2.67 CFE\_SB\_WRONG\_MSG\_TYPE** #define CFE\_SB\_WRONG\_MSG\_TYPE ((CFE\_Status\_t)0xca00000d)

Wrong Message Type.

This error code will be returned when a request such as [CFE\\_MSG\\_SetMsgTime](#) is made on a packet that does not include a field for msg time.

Definition at line 849 of file cfe\_error.h.

**10.10.2.68 CFE\_STATUS\_BAD\_COMMAND\_CODE** #define CFE\_STATUS\_BAD\_COMMAND\_CODE ((CFE\_Status\_t)0xc8000004)

Bad Command Code.

This error code will be returned when a message identification process determined that the command code is does not correspond to any known value

Definition at line 182 of file cfe\_error.h.

**10.10.2.69 CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL** #define CFE\_STATUS\_EXTERNAL\_RESOURCE\_FA←  
IL ((CFE\_Status\_t) 0xc8000005)

External failure.

This error indicates that the operation failed for some reason outside the scope of CFE. The real failure may have been in OSAL, PSP, or another dependent library.

Details of the original failure should be written to syslog and/or a system event before returning this error.

Definition at line 194 of file cfe\_error.h.

**10.10.2.70 CFE\_STATUS\_INCORRECT\_STATE** #define CFE\_STATUS\_INCORRECT\_STATE ((int32) 0xc8000009)

Cannot process request at this time.

The system is not currently in the correct state to accept the request at this time.

Definition at line 227 of file cfe\_error.h.

**10.10.2.71 CFE\_STATUS\_NO\_COUNTER\_INCREMENT** #define CFE\_STATUS\_NO\_COUNTER\_INCREMENT ((CFE\_Status\_t) 0x48000000)

No Counter Increment.

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

Definition at line 155 of file cfe\_error.h.

**10.10.2.72 CFE\_STATUS\_NOT\_IMPLEMENTED** #define CFE\_STATUS\_NOT\_IMPLEMENTED ((CFE\_Status\_t) 0xc800ffff)

Not Implemented.

Current version does not have the function or the feature of the function implemented. This could be due to either an early build for this platform or the platform does not support the specified feature.

Definition at line 238 of file cfe\_error.h.

**10.10.2.73 CFE\_STATUS\_RANGE\_ERROR** #define CFE\_STATUS\_RANGE\_ERROR ((int32) 0xc8000008)

Request or input value is out of range.

A message, table, or function call input contained a value that was outside the acceptable range, and the request was rejected.

Definition at line 219 of file cfe\_error.h.

**10.10.2.74 CFE\_STATUS\_REQUEST\_ALREADY\_PENDING** #define CFE\_STATUS\_REQUEST\_ALREADY\_PENDI←  
NG ((int32) 0xc8000006)

Request already pending.

Commands or requests are already pending or the pending request limit has been reached. No more requests can be made until the current request(s) complete.

Definition at line 203 of file cfe\_error.h.

**10.10.2.75 CFE\_STATUS\_UNKNOWN\_MSG\_ID** #define CFE\_STATUS\_UNKNOWN\_MSG\_ID ((CFE\_Status\_t) 0xc8000003)

Unknown Message ID.

This error code will be returned when a message identification process determined that the message ID does not correspond to a known value

Definition at line 173 of file cfe\_error.h.

**10.10.2.76 CFE\_STATUS\_VALIDATION\_FAILURE** #define CFE\_STATUS\_VALIDATION\_FAILURE ((int32)0xc8000007)  
Request or input value failed basic structural validation.

A message or table input was not in the proper format to be understood and processed by an application, and was rejected.

Definition at line 211 of file cfe\_error.h.

**10.10.2.77 CFE\_STATUS\_WRONG\_MSG\_LENGTH** #define CFE\_STATUS\_WRONG\_MSG\_LENGTH ((CFE\_Status\_t)0xc8000002)  
Wrong Message Length.

This error code will be returned when a message validation process determined that the message length is incorrect

Definition at line 164 of file cfe\_error.h.

**10.10.2.78 CFE\_SUCCESS** #define CFE\_SUCCESS ((CFE\_Status\_t)0)  
Successful execution.

Operation was performed successfully

Definition at line 147 of file cfe\_error.h.

**10.10.2.79 CFE\_TBL\_BAD\_ARGUMENT** #define CFE\_TBL\_BAD\_ARGUMENT ((CFE\_Status\_t)0xcc00002d)  
Bad Argument.

A parameter given by a caller to a Table API did not pass validation checks.

Definition at line 1281 of file cfe\_error.h.

**10.10.2.80 CFE\_TBL\_ERR\_ACCESS** #define CFE\_TBL\_ERR\_ACCESS ((CFE\_Status\_t)0xcc00002c)  
Error code indicating that the TBL file could not be opened by the OS.

Definition at line 1272 of file cfe\_error.h.

**10.10.2.81 CFE\_TBL\_ERR\_BAD\_CONTENT\_ID** #define CFE\_TBL\_ERR\_BAD\_CONTENT\_ID ((CFE\_Status\_t)0xcc000016)  
Bad Content ID.

The calling Application called [CFE\\_TBL\\_Load](#) with a filename that specified a file whose content ID was not that of a table image.

Definition at line 1064 of file cfe\_error.h.

**10.10.2.82 CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID** #define CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID ((CFE\_Status\_t)0xcc000029)  
Bad Processor ID.

The selected table file failed validation for Processor ID. The platform configuration file has verification of table files enabled for Processor ID and an attempt was made to load a table with an invalid Processor ID in the table file header.

Definition at line 1252 of file cfe\_error.h.

**10.10.2.83 CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID** #define CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID ((CFE\_Status\_t)0xcc000028)  
Bad Spacecraft ID.

The selected table file failed validation for Spacecraft ID. The platform configuration file has verification of table files enabled for Spacecraft ID and an attempt was made to load a table with an invalid Spacecraft ID in the table file header.

Definition at line 1241 of file cfe\_error.h.

**10.10.2.84 CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID** #define CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID ((CFE\_Status\_t)0xcc00001B)  
Bad Subtype ID.

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.  
Definition at line 1105 of file cfe\_error.h.

**10.10.2.85 CFE\_TBL\_ERR\_DUMP\_ONLY** #define CFE\_TBL\_ERR\_DUMP\_ONLY ((CFE\_Status\_t)0xcc000010)  
Dump Only Error.

The calling Application has attempted to perform a load on a table that was created with "Dump Only" attributes.  
Definition at line 1016 of file cfe\_error.h.

**10.10.2.86 CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE** #define CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE ((CFE\_Status\_t)0xcc00000C)  
Duplicate Table With Different Size.

An application attempted to register a table with the same name as a table that is already in the registry. The size of the new table is different from the size already in the registry.

Definition at line 977 of file cfe\_error.h.

**10.10.2.87 CFE\_TBL\_ERR\_DUPLICATE\_NOT OWNED** #define CFE\_TBL\_ERR\_DUPLICATE\_NOT OWNED ((CFE\_Status\_t)0xcc00000D)  
Duplicate Table And Not Owned.

An application attempted to register a table with the same name as a table that is already in the registry. The previously registered table is owned by a different application.

Definition at line 987 of file cfe\_error.h.

**10.10.2.88 CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE** #define CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE ((CFE\_Status\_t)0xcc00000E)  
File For Wrong Table.

The calling Application tried to load a table using a file whose header indicated that it was for a different table.  
Definition at line 1149 of file cfe\_error.h.

**10.10.2.89 CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT** #define CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT ((CFE\_Status\_t)0xcc00001C)  
File Size Inconsistent.

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.  
Definition at line 1114 of file cfe\_error.h.

**10.10.2.90 CFE\_TBL\_ERR\_FILE\_TOO\_LARGE** #define CFE\_TBL\_ERR\_FILE\_TOO\_LARGE ((CFE\_Status\_t)0xcc000014)  
File Too Large.

The calling Application called [CFE\\_TBL\\_Load](#) with a filename that specified a file that contained more data than the size of the table OR which contained more data than specified in the table header.

Definition at line 1044 of file cfe\_error.h.

**10.10.2.91 CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG** #define CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG ((CFE\_Status\_t)0xcc00001F)  
Filename Too Long.

The calling Application tried to load a table using a filename that was too long.  
Definition at line 1140 of file cfe\_error.h.

**10.10.2.92 CFE\_TBL\_ERR\_HANDLES\_FULL** #define CFE\_TBL\_ERR\_HANDLES\_FULL ((CFE\_Status\_t)0xcc00000B)  
Handles Full.

An application attempted to create a table and the Table Handle Array already used all CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES in it.

Definition at line 967 of file cfe\_error.h.

**10.10.2.93 CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE** #define CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE ((CFE\_Status\_t)0xcc000011)  
Illegal Source Type.

The calling Application called [CFE\\_TBL\\_Load](#) with an illegal value for the second parameter.

Definition at line 1025 of file cfe\_error.h.

**10.10.2.94 CFE\_TBL\_ERR\_INVALID\_HANDLE** #define CFE\_TBL\_ERR\_INVALID\_HANDLE ((CFE\_Status\_t)0xcc000001)  
Invalid Handle.

The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Definition at line 884 of file cfe\_error.h.

**10.10.2.95 CFE\_TBL\_ERR\_INVALID\_NAME** #define CFE\_TBL\_ERR\_INVALID\_NAME ((CFE\_Status\_t)0xcc000002)  
Invalid Name.

The calling Application attempted to register a table whose name length exceeded the platform configuration value of [CFE\\_MISSION\\_TBL\\_MAX\\_NAME\\_LENGTH](#) or was zero characters long.

Definition at line 894 of file cfe\_error.h.

**10.10.2.96 CFE\_TBL\_ERR\_INVALID\_OPTIONS** #define CFE\_TBL\_ERR\_INVALID\_OPTIONS ((CFE\_Status\_t)0xcc000025)  
Invalid Options.

The calling Application has used an illegal combination of table options. A summary of the illegal combinations are as follows:

#CFE\_TBL\_OPT\_USR\_DEF\_ADDR cannot be combined with any of the following:

1. [CFE\\_TBL\\_OPT\\_DBL\\_BUFFER](#)
2. [CFE\\_TBL\\_OPT\\_LOAD\\_DUMP](#)
3. [CFE\\_TBL\\_OPT\\_CRITICAL](#)

#CFE\_TBL\_OPT\_DBL\_BUFFER cannot be combined with the following:

1. [CFE\\_TBL\\_OPT\\_USR\\_DEF\\_ADDR](#)
2. [CFE\\_TBL\\_OPT\\_DUMP\\_ONLY](#)

Definition at line 1206 of file cfe\_error.h.

**10.10.2.97 CFE\_TBL\_ERR\_INVALID\_SIZE** #define CFE\_TBL\_ERR\_INVALID\_SIZE ((CFE\_Status\_t)0xcc000003)  
Invalid Size.

The calling Application attempted to register a table: a) that was a double buffered table with size greater than [CFE\\_PLATFORM\\_TBL\\_MAX\\_DBL\\_TABLE\\_SIZE](#) b) that was a single buffered table with size greater than [CFE\\_PLATFORM\\_TBL\\_MAX\\_SNGL\\_TABLE\\_SIZE](#) c) that had a size of zero

Definition at line 905 of file cfe\_error.h.

**10.10.2.98 CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS** #define CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS ((CFE\_Status\_t)0xcc000012)  
Load In Progress.

The calling Application called [CFE\\_TBL\\_Load](#) when another Application was trying to load the table.

Definition at line 1034 of file cfe\_error.h.

**10.10.2.99 CFE\_TBL\_ERR\_LOAD\_INCOMPLETE** #define CFE\_TBL\_ERR\_LOAD\_INCOMPLETE ((CFE\_Status\_t)0xcc000021)  
Load Incomplete.

The calling Application tried to load a table file whose header claimed the load was larger than what was actually read from the file.

Definition at line 1158 of file cfe\_error.h.

**10.10.2.100 CFE\_TBL\_ERR\_NEVER\_LOADED** #define CFE\_TBL\_ERR\_NEVER\_LOADED ((CFE\_Status\_t)0xcc000005)  
Never Loaded.

Table has not been loaded with data.

Definition at line 921 of file cfe\_error.h.

**10.10.2.101 CFE\_TBL\_ERR\_NO\_ACCESS** #define CFE\_TBL\_ERR\_NO\_ACCESS ((CFE\_Status\_t)0xcc000008)  
No Access.

The calling application either failed when calling [CFE\\_TBL\\_Register](#), failed when calling [CFE\\_TBL\\_Share](#) or forgot to call either one.

Definition at line 949 of file cfe\_error.h.

**10.10.2.102 CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL** #define CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL ((CFE\_Status\_t)0xcc00000F)  
No Buffer Available.

The calling Application has tried to allocate a working buffer but none were available.

Definition at line 1007 of file cfe\_error.h.

**10.10.2.103 CFE\_TBL\_ERR\_NO\_STD\_HEADER** #define CFE\_TBL\_ERR\_NO\_STD\_HEADER ((CFE\_Status\_t)0xcc00001D)  
No Standard Header.

The calling Application tried to access a table file whose standard cFE File Header was the wrong size, etc.

Definition at line 1122 of file cfe\_error.h.

**10.10.2.104 CFE\_TBL\_ERR\_NO\_TBL\_HEADER** #define CFE\_TBL\_ERR\_NO\_TBL\_HEADER ((CFE\_Status\_t)0xcc00001E)  
No Table Header.

The calling Application tried to access a table file whose standard cFE Table File Header was the wrong size, etc.

Definition at line 1131 of file cfe\_error.h.

**10.10.2.105 CFE\_TBL\_ERR\_PARTIAL\_LOAD** #define CFE\_TBL\_ERR\_PARTIAL\_LOAD ((CFE\_Status\_t)0xcc000023)  
Partial Load Error.

The calling Application tried to load a table file whose header claimed the load did not start with the first byte and the table image had NEVER been loaded before. Partial loads are not allowed on uninitialized tables. It should be noted that [CFE\\_TBL\\_WARN\\_SHORT\\_FILE](#) also indicates a partial load.

Definition at line 1180 of file cfe\_error.h.

**10.10.2.106 CFE\_TBL\_ERR\_REGISTRY\_FULL** #define CFE\_TBL\_ERR\_REGISTRY\_FULL (([CFE\\_Status\\_t](#))0xcc000006)  
Registry Full.

An application attempted to create a table and the Table registry already contained [CFE\\_PLATFORM\\_TBL\\_MAX\\_NUM\\_TABLES](#) in it.

Definition at line 930 of file [cfe\\_error.h](#).

**10.10.2.107 CFE\_TBL\_ERR\_SHORT\_FILE** #define CFE\_TBL\_ERR\_SHORT\_FILE (([CFE\\_Status\\_t](#))0xcc00002b)

Error code indicating that the TBL file is shorter than indicated in the file header.

Definition at line 1266 of file [cfe\\_error.h](#).

**10.10.2.108 CFE\_TBL\_ERR\_UNREGISTERED** #define CFE\_TBL\_ERR\_UNREGISTERED (([CFE\\_Status\\_t](#))0xcc000009)  
Unregistered.

The calling application is trying to access a table that has been unregistered.

Definition at line 958 of file [cfe\\_error.h](#).

**10.10.2.109 CFE\_TBL\_INFO\_DUMP\_PENDING** #define CFE\_TBL\_INFO\_DUMP\_PENDING (([CFE\\_Status\\_t](#))0x4c000024)  
Dump Pending.

The calling Application should call [CFE\\_TBL\\_Manage](#) for the specified table. The ground has requested a dump of the Dump-Only table and needs to synchronize with the owning application.

Definition at line 1190 of file [cfe\\_error.h](#).

**10.10.2.110 CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING** #define CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING (([CFE\\_Status\\_t](#))0x4c000017)  
No Update Pending.

The calling Application has attempted to update a table without a pending load.

Definition at line 1072 of file [cfe\\_error.h](#).

**10.10.2.111 CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING** #define CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDI←  
NG (([CFE\\_Status\\_t](#))0x4c00001A)

No Validation Pending

The calling Application tried to validate a table that did not have a validation request pending.

Definition at line 1096 of file [cfe\\_error.h](#).

**10.10.2.112 CFE\_TBL\_INFO\_RECOVERED\_TBL** #define CFE\_TBL\_INFO\_RECOVERED\_TBL (([CFE\\_Status\\_t](#))0x4c000027)  
Recovered Table.

The calling Application registered a critical table whose previous contents were discovered in the Critical Data Store.

The discovered contents were copied back into the newly registered table as the table's initial contents.

**NOTE:** In this situation, the contents of the table are **NOT** validated using the table's validation function.

Definition at line 1230 of file [cfe\\_error.h](#).

**10.10.2.113 CFE\_TBL\_INFO\_TABLE\_LOCKED** #define CFE\_TBL\_INFO\_TABLE\_LOCKED (([CFE\\_Status\\_t](#))0x4c000018)  
Table Locked.

The calling Application tried to update a table that is locked by another user.

Definition at line 1080 of file [cfe\\_error.h](#).

**10.10.2.114 CFE\_TBL\_INFO\_UPDATE\_PENDING** #define CFE\_TBL\_INFO\_UPDATE\_PENDING (([CFE\\_Status\\_t](#))0x4c000004)  
Update Pending.

The calling Application has identified a table that has a load pending.

Definition at line 913 of file cfe\_error.h.

**10.10.2.115 CFE\_TBL\_INFO\_UPDATED** #define CFE\_TBL\_INFO\_UPDATED (([CFE\\_Status\\_t](#))0x4c00000E)  
Updated.

The calling Application has identified a table that has been updated.

**NOTE:** This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.

Definition at line 998 of file cfe\_error.h.

**10.10.2.116 CFE\_TBL\_INFO\_VALIDATION\_PENDING** #define CFE\_TBL\_INFO\_VALIDATION\_PENDING (([CFE\\_Status\\_t](#))0x4c000010)  
Validation Pending

The calling Application should call [CFE\\_TBL\\_Validate](#) for the specified table.

Definition at line 1088 of file cfe\_error.h.

**10.10.2.117 CFE\_TBL\_MESSAGE\_ERROR** #define CFE\_TBL\_MESSAGE\_ERROR (([CFE\\_Status\\_t](#))0xcc00002a)  
Message Error.

Error code indicating that the TBL command was not processed successfully and that the error counter should be incremented.

Definition at line 1260 of file cfe\_error.h.

**10.10.2.118 CFE\_TBL\_NOT\_IMPLEMENTED** #define CFE\_TBL\_NOT\_IMPLEMENTED (([CFE\\_Status\\_t](#))0xcc00ffff)  
Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 1292 of file cfe\_error.h.

**10.10.2.119 CFE\_TBL\_WARN\_DUPLICATE** #define CFE\_TBL\_WARN\_DUPLICATE (([CFE\\_Status\\_t](#))0x4c000007)  
Duplicate Warning.

This is an error that the registration is trying to replace an existing table with the same name. The previous table stays in place and the new table is rejected.

Definition at line 940 of file cfe\_error.h.

**10.10.2.120 CFE\_TBL\_WARN\_NOT\_CRITICAL** #define CFE\_TBL\_WARN\_NOT\_CRITICAL (([CFE\\_Status\\_t](#))0x4c000026)  
Not Critical Warning.

The calling Application attempted to register a table as "Critical". Table Services failed to create an appropriate Critical Data Store (See System Log for reason) to save the table contents. The table will be treated as a normal table from now on.

Definition at line 1217 of file cfe\_error.h.

**10.10.2.121 CFE\_TBL\_WARN\_PARTIAL\_LOAD** #define CFE\_TBL\_WARN\_PARTIAL\_LOAD (([CFE\\_Status\\_t](#))0x4c000022)  
Partial Load Warning.

The calling Application tried to load a table file whose header claimed the load did not start with the first byte. It should be noted that [CFE\\_TBL\\_WARN\\_SHORT\\_FILE](#) also indicates a partial load.

Definition at line 1168 of file `cfe_error.h`.

**10.10.2.122 CFE\_TBL\_WARN\_SHORT\_FILE** `#define CFE_TBL_WARN_SHORT_FILE ((CFE_Status_t)0x4c000015)`  
Short File Warning.

The calling Application called [CFE\\_TBL\\_Load](#) with a filename that specified a file that started with the first byte of the table but contained less data than the size of the table. It should be noted that [CFE\\_TBL\\_WARN\\_PARTIAL\\_LOAD](#) also indicates a partial load (one that starts at a non-zero offset).

Definition at line 1055 of file `cfe_error.h`.

**10.10.2.123 CFE\_TIME\_BAD\_ARGUMENT** `#define CFE_TIME_BAD_ARGUMENT ((CFE_Status_t)0xce000005)`  
Bad Argument.

A parameter given by a caller to a TIME Services API did not pass validation checks.

Definition at line 1364 of file `cfe_error.h`.

**10.10.2.124 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 1355 of file `cfe_error.h`.

**10.10.2.125 CFE\_TIME\_INTERNAL\_ONLY** `#define CFE_TIME_INTERNAL_ONLY ((CFE_Status_t)0xce000001)`  
Internal Only.

One of the TIME Services API functions to set the time with data from an external time source has been called, but TIME Services has been commanded to not accept external time data. However, the command is still a signal for the Time Server to generate a "time at the tone" command packet using internal data.

Definition at line 1319 of file `cfe_error.h`.

**10.10.2.126 CFE\_TIME\_NOT\_IMPLEMENTED** `#define CFE_TIME_NOT_IMPLEMENTED ((CFE_Status_t)0xce00ffff)`  
Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 1307 of file `cfe_error.h`.

**10.10.2.127 CFE\_TIME\_OUT\_OF\_RANGE** `#define CFE_TIME_OUT_OF_RANGE ((CFE_Status_t)0xce000002)`  
Out Of Range.

One of the TIME Services API functions to set the time with data from an external time source has been called, but TIME Services has determined that the new time data is invalid. However, the command is still a signal for the Time Server to generate a "time at the tone" command packet using internal data.

Note that the test for invalid time update data only occurs if TIME Services has previously been commanded to set the clock state to "valid".

Definition at line 1334 of file `cfe_error.h`.

**10.10.2.128 CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS** #define CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS  
KS ((CFE\_Status\_t)0xce000003)

Too Many Sync Callbacks.

An attempt to register too many cFE Time Services Synchronization callbacks has been made. Only one callback function is allowed per application. It is expected that the application itself will distribute the single callback to child threads as needed.

Definition at line 1345 of file cfe\_error.h.

## 10.11 cFE Resource ID APIs

### Functions

- `CFE_Status_t CFE_ES_AppID_ToIndex (CFE_ES_AppId_t AppID, uint32 *Idx)`  
*Obtain an index value correlating to an ES Application ID.*
- `int32 CFE_ES_LibID_ToIndex (CFE_ES_LibId_t LibId, uint32 *Idx)`  
*Obtain an index value correlating to an ES Library ID.*
- `CFE_Status_t CFE_ES_TaskID_ToIndex (CFE_ES_TaskId_t TaskID, uint32 *Idx)`  
*Obtain an index value correlating to an ES Task ID.*
- `CFE_Status_t CFE_ES_CounterID_ToIndex (CFE_ES_CounterId_t CounterId, uint32 *Idx)`  
*Obtain an index value correlating to an ES Counter ID.*

#### 10.11.1 Detailed Description

#### 10.11.2 Function Documentation

**10.11.2.1 CFE\_ES\_AppID\_ToIndex()** `CFE_Status_t CFE_ES_AppID_ToIndex (`  
`CFE_ES_AppId_t AppID,`  
`uint32 * Idx )`

Obtain an index value correlating to an ES Application ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] application IDs will never overlap, but the index of an application and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

#### Note

There is no inverse of this function - indices cannot be converted back to the original AppID value. The caller should retain the original ID for future use.

#### Parameters

in	<i>AppID</i>	Application ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

#### Returns

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

#### Return values

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

**10.11.2.2 CFE\_ES\_CounterID\_ToIndex()** `CFE_Status_t CFE_ES_CounterID_ToIndex (`  
`CFE_ES_CounterId_t CounterId,`  
`uint32 * Idx )`

Obtain an index value correlating to an ES Counter ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Counter IDs will never overlap, but the index of a Counter and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

#### Note

There is no inverse of this function - indices cannot be converted back to the original CounterID value. The caller should retain the original ID for future use.

#### Parameters

in	<i>Counter</i> <i>Id</i>	Counter ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

#### Returns

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

#### Return values

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

**10.11.2.3 CFE\_ES\_LibID\_ToIndex()** `int32 CFE_ES_LibID_ToIndex (`  
    `CFE_ES_LibId_t LibId,`  
    `uint32 * Idx )`

Obtain an index value correlating to an ES Library ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Library IDs will never overlap, but the index of an Library and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

#### Note

There is no inverse of this function - indices cannot be converted back to the original LibID value. The caller should retain the original ID for future use.

#### Parameters

in	<i>Lib</i> <i>Id</i>	Library ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

#### Returns

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

## Return values

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

```
10.11.2.4 CFE_ES_TaskID_ToIndex() CFE_Status_t CFE_ES_TaskID_ToIndex (
    CFE_ES_TaskId_t TaskID,
    uint32 * Idx )
```

Obtain an index value correlating to an ES Task ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Task IDs will never overlap, but the index of a Task and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

## Note

There is no inverse of this function - indices cannot be converted back to the original TaskID value. The caller should retain the original ID for future use.

## Parameters

in	<i>TaskID</i>	Task ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

## Returns

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

## Return values

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

## 10.12 cFE Entry/Exit APIs

### Functions

- void [CFE\\_ES\\_Main](#) (uint32 StartType, uint32 StartSubtype, uint32 ModelId, const char \*StartFilePath)  
*cFE Main Entry Point used by Board Support Package to start cFE*
- [CFE\\_Status\\_t CFE\\_ES\\_ResetCFE](#) (uint32 ResetType)  
*Reset the cFE Core and all cFE Applications.*

#### 10.12.1 Detailed Description

#### 10.12.2 Function Documentation

**10.12.2.1 CFE\_ES\_Main()** void CFE\_ES\_Main (

```
    uint32 StartType,
    uint32 StartSubtype,
    uint32 ModelId,
    const char * StartFilePath )
```

cFE Main Entry Point used by Board Support Package to start cFE

#### Description

cFE main entry point. This is the entry point into the cFE software. It is called only by the Board Support Package software.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>StartType</i>	Identifies whether this was a <a href="#">CFE_PSP_RST_TYPE_POWERON</a> or <a href="#">CFE_PSP_RST_TYPE_PROCESSOR</a> .
in	<i>StartSubtype</i>	Specifies, in more detail, what caused the <i>StartType</i> identified above. See <a href="#">CFE_PSP_RST_SUBTYPE_POWER_CYCLE</a> for possible examples.
in	<i>ModelId</i>	Identifies the source of the Boot as determined by the BSP.
in	<i>StartFilePath</i>	Identifies the startup file to use to initialize the cFE apps.

#### See also

[CFE\\_ES\\_ResetCFE](#)

**10.12.2.2 CFE\_ES\_ResetCFE()** [CFE\\_Status\\_t CFE\\_ES\\_ResetCFE](#) (

```
    uint32 ResetType )
```

Reset the cFE Core and all cFE Applications.

#### Description

This API causes an immediate reset of the cFE Kernel and all cFE Applications. The caller can specify whether the reset should clear all memory ([CFE\\_PSP\\_RST\\_TYPE\\_POWERON](#)) or try to retain volatile memory areas ([CFE\\_PSP\\_RST\\_TYPE\\_PROCESSOR](#)).

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>ResetType</i>	Identifies the type of reset desired. Allowable settings are: <ul style="list-style-type: none"><li>• <a href="#">CFE_PSP_RST_TYPE_POWERON</a> - Causes all memory to be cleared</li><li>• <a href="#">CFE_PSP_RST_TYPE_PROCESSOR</a> - Attempts to retain volatile disk, critical data store and user reserved memory.</li></ul>
----	------------------	---

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_ES_NOT_IMPLEMENTED</a>	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	<code>AppID</code>	Identifies the application to be reset.
----	--------------------	---

#### Returns

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

#### Return values

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

#### See also

[CFE\\_ES\\_RestartApp](#), [CFE\\_ES\\_ReloadApp](#)

##### 10.13.2.2 `CFE_ES_ReloadApp()` `CFE_Status_t CFE_ES_ReloadApp (` `CFE_ES_AppId_t AppID,` `const char * AppFileName )`

Reload a single cFE Application.

### Description

This API causes a cFE Application to be stopped and restarted from the specified file.

### Assumptions, External Events, and Notes:

The filename is checked for existence prior to load. A missing file will be reported and the reload operation will be aborted prior to unloading the app.

Goes through the standard CFE\_ES\_CleanUpApp which unloads, then attempts a load using the specified file name. In the event that an application cannot be reloaded due to a corrupt file, the application may no longer be reloaded when given a valid load file (it has been deleted and no longer exists). To recover, the application may be started by loading the application via the ES\_STARTAPP command ([CFE\\_ES\\_START\\_APP\\_CC](#)).

### Parameters

in	<i>AppID</i>	Identifies the application to be reset.
in	<i>AppFileName</i>	Identifies the new file to start (must not be null)

### Returns

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

### Return values

<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_FILE_IO_ERR</a>	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	<i>AppID</i>	Identifies the application to be reset.
----	--------------	---

**Returns**

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

**Return values**

<a href="#"><i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i></a>	Resource ID is not valid.
<a href="#"><i>CFE_ES_FILE_IO_ERR</i></a>	File IO Error.
<a href="#"><i>CFE_SUCCESS</i></a>	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	<a href="#">ExitStatus</a>	Acceptable values are: <ul style="list-style-type: none"><li>• <a href="#">CFE_ES_RunStatus_APP_EXIT</a> - Indicates that the Application wants to exit normally.</li><li>• <a href="#">CFE_ES_RunStatus_APP_ERROR</a> - Indicates that the Application is quitting with an error.</li><li>• <a href="#">CFE_ES_RunStatus_CORE_APP_INIT_ERROR</a> - Indicates that the Core Application could not Init.</li><li>• <a href="#">CFE_ES_RunStatus_CORE_APP_RUNTIME_ERROR</a> - Indicates that the Core Application had a runtime failure.</li></ul>
----	----------------------------	--

##### See also

[CFE\\_ES\\_RunLoop](#)

Referenced by CS\_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\_RESET commands) or if a system error occurs requiring the app to be shut down administratively, this function returns "false" and optionally sets the "RunStatus" output to further indicate the specific application state.

If "RunStatus" is passed as non-NULL, it should point to a local status variable containing the requested status to ES. Normally, this should be initialized to [CFE\\_ES\\_RunStatus\\_APP\\_RUN](#) during application start up, and should remain as this value during normal operation.

If "RunStatus" is set to [CFE\\_ES\\_RunStatus\\_APP\\_EXIT](#) or [CFE\\_ES\\_RunStatus\\_APP\\_ERROR](#) on input, this acts as a shutdown request - [CFE\\_ES\\_RunLoop\(\)](#) function will return "false", and a shutdown will be initiated similar to if ES had been externally commanded to shut down the app.

If "RunStatus" is not used, it should be passed as NULL. In this mode, only the boolean return value is relevant, which will indicate if an externally-initiated shutdown request is pending.

#### Parameters

in, out	<i>RunStatus</i>	Optional pointer to a variable containing the desired run status
---------	------------------	--

#### Returns

Boolean indicating application should continue running

**Return values**

<i>true</i>	Application should continue running
<i>false</i>	Application should not continue running

**See also**

[CFE\\_ES\\_ExitApp](#)

Referenced by CS\_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**

<i>in</i>	<i>TimeOutMilliseconds</i>	The timeout value in Milliseconds. This parameter must be at least 1000. Lower values will be rounded up. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.
-----------	----------------------------	---

**See also**

[CFE\\_ES\\_RunLoop](#)

Referenced by CS\_AppMain().

**10.14.2.5 CFE\_ES\_WaitForSystemState()** `CFE_Status_t CFE_ES_WaitForSystemState (`  
`uint32 MinSystemState,`  
`uint32 TimeOutMilliseconds )`

Allow an Application to Wait for a minimum global system state.

**Description**

This is the API that allows an app to wait for the rest of the apps to complete a given stage of initialization before continuing.

This gives finer grained control than [CFE\\_ES\\_WaitForStartupSync](#)

**Assumptions, External Events, and Notes:**

This API assumes that the caller has also been initialized sufficiently to satisfy the global system state it is waiting for, and the apps own state will be updated accordingly.

**Parameters**

in	<i>MinSystemState</i>	Determine the state of the App
in	<i>TimeOutMilliseconds</i>	The timeout value in Milliseconds. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_RunLoop](#)

## 10.15 cFE Information APIs

### Functions

- `int32 CFE_ES_GetResetType (uint32 *ResetSubtypePtr)`  
*Return the most recent Reset Type.*
- `CFE_Status_t CFE_ES_GetAppID (CFE_ES_AppId_t *AppIdPtr)`  
*Get an Application ID for the calling Application.*
- `CFE_Status_t CFE_ES_GetTaskID (CFE_ES_TaskId_t *TaskIdPtr)`  
*Get the task ID of the calling context.*
- `CFE_Status_t CFE_ES_GetAppIDByName (CFE_ES_AppId_t *AppIdPtr, const char *AppName)`  
*Get an Application ID associated with a specified Application name.*
- `CFE_Status_t CFE_ES_GetLibIDByName (CFE_ES_LibId_t *LibIdPtr, const char *LibName)`  
*Get a Library ID associated with a specified Library name.*
- `CFE_Status_t CFE_ES_GetAppName (char *AppName, CFE_ES_AppId_t AppId, size_t BufferLength)`  
*Get an Application name for a specified Application ID.*
- `CFE_Status_t CFE_ES_GetLibName (char *LibName, CFE_ES_LibId_t LibId, size_t BufferLength)`  
*Get a Library name for a specified Library ID.*
- `CFE_Status_t CFE_ES_GetAppInfo (CFE_ES_AppInfo_t *AppInfo, CFE_ES_AppId_t AppId)`  
*Get Application Information given a specified App ID.*
- `CFE_Status_t CFE_ES_GetTaskInfo (CFE_ES_TaskInfo_t *TaskInfo, CFE_ES_TaskId_t TaskId)`  
*Get Task Information given a specified Task ID.*
- `int32 CFE_ES_GetLibInfo (CFE_ES_AppInfo_t *LibInfo, CFE_ES_LibId_t LibId)`  
*Get Library Information given a specified Resource ID.*
- `int32 CFE_ES_GetModuleInfo (CFE_ES_AppInfo_t *ModuleInfo, CFE_Resourceld_t Resourceld)`  
*Get Information given a specified Resource ID.*

### 10.15.1 Detailed Description

### 10.15.2 Function Documentation

**10.15.2.1 CFE\_ES\_GetAppID()** `CFE_Status_t CFE_ES_GetAppID (`  
`CFE_ES_AppId_t * AppIdPtr )`

Get an Application ID for the calling Application.

#### Description

This routine retrieves the cFE Application ID for the calling Application.

#### Assumptions, External Events, and Notes:

NOTE: All tasks associated with the Application would return the same Application ID.

#### Parameters

<code>out</code>	<code>AppIdPtr</code>	Pointer to variable that is to receive the Application's ID (must not be null). <code>*AppIdPtr</code> will be set to the application ID of the calling Application.
------------------	-----------------------	--

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetResetType](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppName](#), [CFE\\_ES\\_GetTaskInfo](#)

Referenced by [CS\\_ProcessNewTablesDefinitionTable\(\)](#).

**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**

<i>out</i>	<i>AppIdPtr</i>	Pointer to variable that is to receive the Application's ID (must not be null).
<i>in</i>	<i>AppName</i>	Pointer to null terminated character string containing an Application name (must not be null).

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetAppID](#), [CFE\\_ES\\_GetAppName](#), [CFE\\_ES\\_GetAppInfo](#)

Referenced by [CS\\_ComputeApp\(\)](#).

```
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

<code>out</code>	<code>AppInfo</code>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
<code>in</code>	<code>AppId</code>	ID of application to obtain information about

#### Returns

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

#### Return values

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

#### See also

[CFE\\_ES\\_GetAppID](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppName](#)

```
10.15.2.4 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**

<i>out</i>	<i>AppName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the appropriate Application name.
<i>in</i>	<i>AppId</i>	Application ID of Application whose name is being requested.
<i>in</i>	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <i>AppName</i> buffer. This routine will truncate the name to this length, if necessary.

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetAppID](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppInfo](#)

Referenced by `CS_ProcessNewTablesDefinitionTable()`.

**10.15.2.5 CFE\_ES\_GetLibIDByName()** [CFE\\_Status\\_t](#) `CFE_ES_GetLibIDByName (`  
`CFE\_ES\_LibId\_t * LibIdPtr,`  
`const char * LibName )`

Get a Library ID associated with a specified Library name.

**Description**

This routine retrieves the cFE Library ID associated with a specified Library name.

**Assumptions, External Events, and Notes:**

None

**Parameters**

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

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetLibName](#)

Referenced by CS\_ComputeApp().

```
10.15.2.6 CFE_ES_GetLibInfo() int32 CFE_ES_GetLibInfo (
    CFE_ES_AppInfo_t * LibInfo,
    CFE_ES_LibId_t LibId )
```

Get Library Information given a specified Resource ID.

**Description**

This routine retrieves the information about a Library associated with a specified ID. The information includes all of the information ES maintains for this resource type ( documented in the CFE\_ES\_AppInfo\_t type ).

This shares the same output structure as CFE\_ES\_GetAppInfo, such that informational commands can be executed against either applications or libraries. When applied to a library, the task information in the structure will be omitted, as libraries do not have tasks associated.

**Assumptions, External Events, and Notes:**

None

**Parameters**

out	<i>LibInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	<i>LibId</i>	ID of application to obtain information about

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetLibIDByName](#), [CFE\\_ES\\_GetLibName](#)

```
10.15.2.7 CFE_ES_GetLibName() CFE_Status_t CFE_ES_GetLibName (
    char * LibName,
    CFE_ES_LibId_t LibId,
    size_t BufferLength )
```

Get a Library name for a specified Library ID.

### Description

This routine retrieves the cFE Library name associated with a specified Library ID.

### Assumptions, External Events, and Notes:

In the case of a failure ([CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID](#)), an empty string is returned.

### Parameters

out	<i>LibName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Library name.
in	<i>LibId</i>	Library ID of Library whose name is being requested.
in	<i>BufferLength</i>	The maximum number of characters (must not be zero), including the null terminator, that can be put into the <i>LibName</i> buffer. This routine will truncate the name to this length, if necessary.

### Returns

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

### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	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**

<i>out</i>	<i>ModuleInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
<i>in</i>	<i>ResourceId</i>	ID of application or library to obtain information about

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetLibInfo](#), [CFE\\_ES\\_GetApplInfo](#)

Referenced by `CS_ComputeApp()`.

**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**

<i>in,out</i>	<i>ResetSubtypePtr</i>	Pointer to <code>uint32</code> type variable in which the Reset Sub-Type will be stored. The caller can set this pointer to NULL if the Sub-Type is of no interest. <i>ResetSubtypePtr</i> If the provided pointer was not NULL, the Reset Sub-Type is stored at the given address. For a list of possible Sub-Type values, see " <a href="#">Reset Sub-Types</a> ".
---------------	------------------------	---

**Returns**

Processor reset type

**Return values**

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

**See also**

[CFE\\_ES\\_GetAppID](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppName](#), [CFE\\_ES\\_GetTaskInfo](#)

**10.15.2.10 CFE\_ES\_GetTaskID()** `CFE_Status_t` CFE\_ES\_GetTaskID (   
     `CFE_ES_TaskId_t` \* `TaskIdPtr` )

Get the task ID of the calling context.

**Description**

This retrieves the current task context from OSAL

**Assumptions, External Events, and Notes:**

Applications which desire to call other CFE ES services such as CFE\_ES\_TaskGetInfo() should use this API rather than getting the ID from OSAL directly via [OS\\_TaskGetId\(\)](#).

**Parameters**

<code>out</code>	<code>TaskIdPtr</code>	Pointer to variable that is to receive the ID (must not be null). Will be set to the ID of the calling task.
------------------	------------------------	--

**Returns**

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

**Return values**

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

**10.15.2.11 CFE\_ES\_GetTaskInfo()** `CFE_Status_t` CFE\_ES\_GetTaskInfo (   
     `CFE_ES_TaskInfo_t` \* `TaskInfo`,   
     `CFE_ES_TaskId_t` `TaskId` )

Get Task Information given a specified Task ID.

**Description**

This routine retrieves the information about a Task associated with a specified Task ID. The information includes Task Name, and Parent/Creator Application ID.

**Assumptions, External Events, and Notes:**

None

**Parameters**

out	<i>TaskInfo</i>	Pointer to a 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	<i>TaskId</i>	Application ID of Application whose name is being requested.

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetTaskID](#), [CFE\\_ES\\_GetTaskIDByName](#), [CFE\\_ES\\_GetTaskName](#)

## 10.16 cFE Child Task APIs

### Functions

- `CFE_Status_t CFE_ES_CreateChildTask (CFE_ES_TaskId_t *TaskIdPtr, const char *TaskName, CFE_ES_ChildTaskMainFuncPtr FunctionPtr, CFE_ES_StackPointer_t StackPtr, size_t StackSize, CFE_ES_TaskPriority_Atom_t Priority, uint32 Flags)`  
*Creates a new task under an existing Application.*
- `CFE_Status_t CFE_ES_GetTaskIDByName (CFE_ES_TaskId_t *TaskIdPtr, const char *TaskName)`  
*Get a Task ID associated with a specified Task name.*
- `CFE_Status_t CFE_ES_GetTaskName (char *TaskName, CFE_ES_TaskId_t TaskId, size_t BufferLength)`  
*Get a Task name for a specified Task ID.*
- `CFE_Status_t CFE_ES_DeleteChildTask (CFE_ES_TaskId_t TaskId)`  
*Deletes a task under an existing Application.*
- `void CFE_ES_ExitChildTask (void)`  
*Exits a child task.*

### 10.16.1 Detailed Description

### 10.16.2 Function Documentation

**10.16.2.1 CFE\_ES\_CreateChildTask()** `CFE_Status_t CFE_ES_CreateChildTask (`  
`CFE_ES_TaskId_t * TaskIdPtr,`  
`const char * TaskName,`  
`CFE_ES_ChildTaskMainFuncPtr_t FunctionPtr,`  
`CFE_ES_StackPointer_t StackPtr,`  
`size_t StackSize,`  
`CFE_ES_TaskPriority_Atom_t Priority,`  
`uint32 Flags )`

Creates a new task under an existing Application.

#### Description

This routine creates a new task (a separate execution thread) owned by the calling Application.

#### Assumptions, External Events, and Notes:

None

#### Parameters

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

**Parameters**

in	<i>Priority</i>	The priority for the new task. Lower numbers are higher priority, with 0 being the highest priority.
in	<i>Flags</i>	Reserved for future expansion.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_CHILD_TASK_CREATE</a>	Child Task Create Error.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.

**See also**

[CFE\\_ES\\_DeleteChildTask](#), [CFE\\_ES\\_ExitChildTask](#)

Referenced by CS\_OneShotCmd(), CS\_RecomputeBaselineAppCmd(), CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), and CS\_RecomputeBaselineTablesCmd().

### 10.16.2.2 CFE\_ES\_DeleteChildTask()

```
CFE_Status_t CFE_ES_DeleteChildTask (
    CFE_ES_TaskId_t TaskId )
```

Deletes a task under an existing Application.

**Description**

This routine deletes a task under an Application specified by the `TaskId` obtained when the child task was created using the [CFE\\_ES\\_CreateChildTask](#) API.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TaskId</i>	The task ID previously obtained when the Child Task was created with the <a href="#">CFE_ES_CreateChildTask</a> API.
----	---------------	--

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

**Return values**

<a href="#"><code>CFE_ES_ERR_CHILD_TASK_DELETE</code></a>	(return value only verified in coverage test) Child Task Delete Error.
<a href="#"><code>CFE_ES_ERR_CHILD_TASK_DELETE_MAIN_TASK</code></a>	Child Task Delete Passed Main Task.
<a href="#"><code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code></a>	Resource ID is not valid.

**See also**

[CFE\\_ES\\_CreateChildTask](#), [CFE\\_ES\\_ExitChildTask](#)

Referenced by `CS_CancelOneShotCmd()`.

**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 `CS_OneShotChildTask()`, `CS_RecomputeAppChildTask()`, `CS_RecomputeEepromMemoryChildTask()`, and `CS_RecomputeTablesChildTask()`.

**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**

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

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_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**

<i>out</i>	<i>TaskName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Task name.
<i>in</i>	<i>TaskId</i>	Task ID of Task whose name is being requested.
<i>in</i>	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <i>TaskName</i> buffer. This routine will truncate the name to this length, if necessary.

**Returns**

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

**Return values**

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

See also

[CFE\\_ES\\_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,  
[CFE\\_ES\\_CrcType\\_Enum\\_t](#) TypeCRC)  
*Calculate a CRC on a block of memory.*
- void [CFE\\_ES\\_ProcessAsyncEvent](#) (void)  
*Notification that an asynchronous event was detected by the underlying OS/PSP.*

### 10.17.1 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,  
    [CFE\\_ES\\_CrcType\\_Enum\\_t](#) TypeCRC )

Calculate a CRC on a block of memory.

##### Description

This routine calculates a cyclic redundancy check (CRC) on a block of memory. The CRC algorithm used is determined by the last parameter.

##### Assumptions, External Events, and Notes:

None

**Parameters**

in	<i>DataPtr</i>	Pointer to the base of the memory block.
in	<i>DataLength</i>	The number of bytes in the memory block.
in	<i>InputCRC</i>	A starting value for use in the CRC calculation. This parameter allows the user to calculate the CRC of non-contiguous blocks as a single value. Nominally, the user should set this value to zero.
in	<i>TypeCRC</i>	<p>One of the following CRC algorithm selections:</p> <ul style="list-style-type: none"> <li>• CFE_ES_CrcType_CRC_8 - (Not currently implemented)</li> <li>• CFE_ES_CrcType_CRC_16 - CRC-16/ARC Polynomial: 0x8005 Initialization: 0x0000 Reflect Input/Output: true XorOut: 0x0000</li> <li>• CFE_ES_CrcType_CRC_32 - (not currently implemented)</li> </ul>

**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 CS\_ComputeApp(), CS\_ComputeEepromMemory(), CS\_ComputeTables(), and CS\_OneShotChildTask().

**10.17.2.3 CFE\_ES\_ProcessAsyncEvent()** `void CFE_ES_ProcessAsyncEvent (`  
 `void )`

Notification that an asynchronous event was detected by the underlying OS/PSP.

**Description**

This hook routine is called from the PSP when an exception or other asynchronous system event occurs

**Assumptions, External Events, and Notes:**

The PSP must guarantee that this function is only invoked from a context which may use OSAL primitives. In general this means that it shouldn't be *directly* invoked from an ISR/signal context.

**10.17.2.4 CFE\_ES\_WriteToSysLog()** `CFE_Status_t CFE_ES_WriteToSysLog (`  
 `const char * SpecStringPtr,`  
 `... )`

Write a string to the cFE System Log.

**Description**

This routine writes a formatted string to the cFE system log. This can be used to record very low-level errors that can't be reported using the Event Services. This function is used in place of printf for flight software. It should be used for significant startup events, critical errors, and conditionally compiled debug software.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>SpecStringPtr</i>	The format string for the log message (must not be null). This is similar to the format string for a printf() call.
----	----------------------	---

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_SYS_LOG_FULL</i>	System Log Full.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

Referenced by CS\_AppInit(), and CS\_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 \(char \*BlockName, CFE\_ES\_CDSHandle\_t BlockId, size\_t BufferLength\)`](#)  
*Get a Block name for a specified Block ID.*
- [`CFE\_Status\_t CFE\_ES\_CopyToCDS \(CFE\_ES\_CDSHandle\_t Handle, const void \*DataToCopy\)`](#)  
*Save a block of data in the Critical Data Store (CDS)*
- [`CFE\_Status\_t CFE\_ES\_RestoreFromCDS \(void \*RestoreToMemory, CFE\_ES\_CDSHandle\_t Handle\)`](#)  
*Recover a block of data from the Critical Data Store (CDS)*

#### 10.18.1 Detailed Description

#### 10.18.2 Function Documentation

**10.18.2.1 `CFE_ES_CopyToCDS()`** [`CFE\_Status\_t CFE\_ES\_CopyToCDS \(`](#)  
[`CFE\_ES\_CDSHandle\_t Handle,`](#)  
[`const void \* DataToCopy \)`](#)

Save a block of data in the Critical Data Store (CDS)

##### Description

This routine copies a specified block of memory into the Critical Data Store that had been previously registered via [`CFE\_ES\_RegisterCDS`](#). The block of memory to be copied must be at least as big as the size specified when registering the CDS.

##### Assumptions, External Events, and Notes:

None

##### Parameters

in	<code>Handle</code>	The handle of the CDS block that was previously obtained from <a href="#"><code>CFE_ES_RegisterCDS</code></a> .
in	<code>DataToCopy</code>	A Pointer to the block of memory to be copied into the CDS (must not be null).

##### Returns

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

##### Return values

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

**See also**

[CFE\\_ES\\_RegisterCDS](#), [CFE\\_ES\\_RestoreFromCDS](#)

Referenced by CS\_CreateRestoreStatesFromCDS(), and CS\_UpdateCDS().

**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	<i>BlockIdPtr</i>	Pointer to variable that is to receive the CDS Block ID (must not be null).
in	<i>BlockName</i>	Pointer to null terminated character string containing a CDS Block name (must not be null).

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_NAME_NOT_FOUND</i>	Resource Name Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_ES_NOT_IMPLEMENTED</i>	The processor does not support a Critical Data Store.

**See also**

[CFE\\_ES\\_GetCDSBlockName](#)

**10.18.2.3 CFE\_ES\_GetCDSBlockName()** `CFE_Status_t CFE_ES_GetCDSBlockName (`  
 `char * BlockName,`  
 `CFE_ES_CDSHandle_t BlockId,`  
 `size_t BufferLength )`

Get a Block name for a specified Block ID.

**Description**

This routine retrieves the cFE Block name associated with a specified Block ID.

**Assumptions, External Events, and Notes:**

In the case of a failure ([CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID](#)), an empty string is returned.

**Parameters**

out	<i>BlockName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the CDS Block name.
in	<i>BlockId</i>	Block ID/Handle of CDS registry entry whose name is being requested.
in	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <i>BlockName</i> buffer. This routine will truncate the name to this length, if necessary.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_ES_NOT_IMPLEMENTED</a>	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**

<b>out</b>	<i>CDSHandlePtr</i>	Pointer Application's variable that will contain the CDS Memory Block Handle (must not be null). HandlePtr is the handle of the CDS block that can be used in <a href="#">CFE_ES_CopyToCDS</a> and <a href="#">CFE_ES_RestoreFromCDS</a> .
<b>in</b>	<i>BlockSize</i>	The number of bytes needed in the CDS (must not be zero).
<b>in</b>	<i>Name</i>	A pointer to a character string (must not be null) containing an application unique name of <a href="#">CFE_MISSION_ES_CDS_MAX_NAME_LENGTH</a> characters or less.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	The memory block was successfully created in the CDS.
<a href="#">CFE_ES_NOT_IMPLEMENTED</a>	The processor does not support a Critical Data Store.
<a href="#">CFE_ES_CDS_ALREADY_EXISTS</a>	CDS Already Exists.
<a href="#">CFE_ES_CDS_INVALID_SIZE</a>	CDS Invalid Size.
<a href="#">CFE_ES_CDS_INVALID_NAME</a>	CDS Invalid Name.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_ES_CDS_INVALID</a>	(return value only verified in coverage test) CDS Invalid.

**See also**

[CFE\\_ES\\_CopyToCDS](#), [CFE\\_ES\\_RestoreFromCDS](#)

Referenced by [CS\\_CreateRestoreStatesFromCDS\(\)](#).

**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**

<b>in</b>	<i>Handle</i>	The handle of the CDS block that was previously obtained from <a href="#">CFE_ES_RegisterCDS</a> .
<b>out</b>	<i>RestoreToMemory</i>	A Pointer to the block of memory (must not be null) that is to be restored with the contents of the CDS. *RestoreToMemory is the contents of the specified CDS.

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_CDS_BLOCK_CRC_ERR</i>	(return value only verified in coverage test) CDS Block CRC Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

**See also**

[CFE\\_ES\\_RegisterCDS](#), [CFE\\_ES\\_CopyToCDS](#)

Referenced by `CS_CreateRestoreStatesFromCDS()`.

## 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

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

#### Returns

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

#### Return values

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

#### See also

[CFE\\_ES\\_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	<i>BufPtr</i>	A pointer to the Application's pointer (must not be null) in which will be stored the address of the allocated memory buffer. <i>*BufPtr</i> is the address of the requested buffer.
in	<i>Handle</i>	The handle to the memory pool as returned by <a href="#">CFE_ES_PoolCreate</a> or <a href="#">CFE_ES_PoolCreateNoSem</a> .
in	<i>Size</i>	The size of the buffer requested. NOTE: The size allocated may be larger.

#### Returns

Bytes Allocated, or error code cFE Return Code Defines

#### Return values

<code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code>	Resource ID is not valid.
<code>CFE_ES_ERR_MEM_BLOCK_SIZE</code>	Memory Block Size Error.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.

#### See also

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_PoolCreateEx](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#), [CFE\\_ES\\_GetPoolBufInfo](#)

```
10.19.2.3 CFE_ES_GetPoolBufInfo() CFE_Status_t CFE_ES_GetPoolBufInfo (
```

```
CFE_ES_MemHandle_t Handle,
CFE_ES_MemPoolBuf_t BufPtr )
```

Gets info on a buffer previously allocated via [CFE\\_ES\\_GetPoolBuf](#).

#### Description

This routine gets info on a buffer in the memory pool.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>Handle</i>	The handle to the memory pool as returned by <a href="#">CFE_ES_PoolCreate</a> or <a href="#">CFE_ES_PoolCreateNoSem</a> .
in	<i>BufPtr</i>	A pointer to the memory buffer to provide status for (must not be null).

#### Returns

Size of the buffer if successful, or status code if not successful, see [cFE Return Code Defines](#)

#### Return values

<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_ES_BUFFER_NOT_IN_POOL</a>	Buffer Not In Pool.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	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**

<i>out</i>	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
<i>in</i>	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The <a href="#">CFE_ES_STATIC_POOL_TYPE</a> macro may be used to assist in creating properly aligned memory pools.
<i>in</i>	<i>Size</i>	The size of the pool of memory (must not be zero). Note that this must be an integral multiple of the memory alignment of the processor architecture.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	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**

<i>out</i>	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
<i>in</i>	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The <a href="#">CFE_ES_STATIC_POOL_TYPE</a> macro may be used to assist in creating properly aligned memory pools.

**Parameters**

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

**Returns**

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

**Return values**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.
<code>CFE_ES_NO_RESOURCE_IDS_AVAILABLE</code>	Resource ID is not available.
<code>CFE_STATUS_EXTERNAL_RESOURCE_FAIL</code>	(return value only verified in coverage test) External failure.

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#)

**10.19.2.6 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	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
in	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The <a href="#">CFE_ES_STATIC_POOL_TYPE</a> macro may be used to assist in creating properly aligned memory pools.
in	<i>Size</i>	The size of the pool of memory (must not be zero). Note that this must be an integral multiple of the memory alignment of the processor architecture.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateEx](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#)

**10.19.2.7 CFE\_ES\_PoolDelete()** `int32 CFE_ES_PoolDelete (`  
`CFE_ES_MemHandle_t PoolID )`

Deletes a memory pool that was previously created.

**Description**

This routine removes the pool ID and frees the global table entry for future re-use.

**Assumptions, External Events, and Notes:**

All buffers associated with the pool become invalid after this call. The application should ensure that buffers/references to the pool are returned before deleting the pool.

**Parameters**

in	<i>PoolID</i>	The ID of the pool to delete
----	---------------	------------------------------

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#)

**10.19.2.8 CFE\_ES\_PutPoolBuf()** `int32 CFE_ES_PutPoolBuf (`  
 `CFE_ES_MemHandle_t Handle,`  
 `CFE_ES_MemPoolBuf_t BufPtr )`

Releases a buffer from the memory pool that was previously allocated via [CFE\\_ES\\_GetPoolBuf](#).

**Description**

This routine releases a buffer back into the memory pool.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Handle</i>	The handle to the memory pool as returned by <a href="#">CFE_ES_PoolCreate</a> or <a href="#">CFE_ES_PoolCreateNoSem</a> .
in	<i>BufPtr</i>	A pointer to the memory buffer to be released (must not be null).

**Returns**

Bytes released, or error code cFE Return Code Defines

**Return values**

<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_ES_BUFFER_NOT_IN_POOL</a>	Buffer Not In Pool.
<a href="#">CFE_ES_POOL_BLOCK_INVALID</a>	Invalid pool block.

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_PoolCreateEx](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#),  
[CFE\\_ES\\_GetPoolBufInfo](#)

## 10.20 cFE Performance Monitor APIs

### Macros

- `#define CFE_ES_PerfLogEntry(id) (CFE_ES_PerfLogAdd(id, 0))`  
*Entry marker for use with Software Performance Analysis Tool.*
- `#define CFE_ES_PerfLogExit(id) (CFE_ES_PerfLogAdd(id, 1))`  
*Exit marker for use with Software Performance Analysis Tool.*

### Functions

- `void CFE_ES_PerfLogAdd (uint32 Marker, uint32 EntryExit)`  
*Adds a new entry to the data buffer.*

#### 10.20.1 Detailed Description

#### 10.20.2 Macro Definition Documentation

##### 10.20.2.1 CFE\_ES\_PerfLogEntry `#define CFE_ES_PerfLogEntry(` `id ) (CFE_ES_PerfLogAdd(id, 0))`

Entry marker for use with Software Performance Analysis Tool.

#### Description

This macro logs the entry or start event/marker for the specified entry `id`. This macro, in conjunction with the `CFE_ES_PerfLogExit`, is used by the Software Performance Analysis tool (see section 5.15).

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>id</code>	Identifier of the specific event or marker.
----	-----------------	---

#### See also

`CFE_ES_PerfLogExit`, `CFE_ES_PerfLogAdd`

Definition at line 1471 of file `cfe_es.h`.

##### 10.20.2.2 CFE\_ES\_PerfLogExit `#define CFE_ES_PerfLogExit(` `id ) (CFE_ES_PerfLogAdd(id, 1))`

Exit marker for use with Software Performance Analysis Tool.

#### Description

This macro logs the exit or end event/marker for the specified entry `id`. This macro, in conjunction with the `CFE_ES_PerfLogEntry`, is used by the Software Performance Analysis tool (see section 5.15).

#### Assumptions, External Events, and Notes:

None

**Parameters**

in	<i>id</i>	Identifier of the specific event or marker.
----	-----------	---

**See also**

[CFE\\_ES\\_PerfLogEntry](#), [CFE\\_ES\\_PerfLogAdd](#)

Definition at line 1490 of file `cfe_es.h`.

### 10.20.3 Function Documentation

**10.20.3.1 CFE\_ES\_PerfLogAdd()** `void CFE_ES_PerfLogAdd (`  
`uint32 Marker,`  
`uint32 EntryExit )`

Adds a new entry to the data buffer.

Function called by [CFE\\_ES\\_PerfLogEntry](#) and [CFE\\_ES\\_PerfLogExit](#) macros

**Description**

This function logs the entry and exit marker for the specified *id*. This function is used by the Software Performance Analysis tool (see section 5.15).

**Assumptions, External Events, and Notes:**

This function implements a circular buffer using an array. DataStart points to first stored entry DataEnd points to next available entry if DataStart == DataEnd then the buffer is either empty or full depending on the value of the DataCount Time is stored as 2 32 bit integers, (TimerLower32, TimerUpper32): TimerLower32 is the current value of the hardware timer register. TimerUpper32 is the number of times the timer has rolled over.

**Parameters**

in	<i>Marker</i>	Identifier of the specific event or marker.
in	<i>EntryExit</i>	Used to specify Entry(0) or Exit(1)

**See also**

[CFE\\_ES\\_PerfLogEntry](#), [CFE\\_ES\\_PerfLogExit](#)

## 10.21 cFE Generic Counter APIs

### Functions

- `CFE_Status_t CFE_ES_RegisterGenCounter (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`  
*Register a generic counter.*
- `CFE_Status_t CFE_ES_DeleteGenCounter (CFE_ES_CounterId_t CounterId)`  
*Delete a generic counter.*
- `CFE_Status_t CFE_ES_IncrementGenCounter (CFE_ES_CounterId_t CounterId)`  
*Increments the specified generic counter.*
- `CFE_Status_t CFE_ES_SetGenCount (CFE_ES_CounterId_t CounterId, uint32 Count)`  
*Set the specified generic counter.*
- `CFE_Status_t CFE_ES_GetGenCount (CFE_ES_CounterId_t CounterId, uint32 *Count)`  
*Get the specified generic counter count.*
- `CFE_Status_t CFE_ES_GetGenCounterIDByName (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`  
*Get the Id associated with a generic counter name.*
- `CFE_Status_t CFE_ES_GetGenCounterName (char *CounterName, CFE_ES_CounterId_t CounterId, size_t BufferLength)`  
*Get a Counter name for a specified Counter ID.*

#### 10.21.1 Detailed Description

#### 10.21.2 Function Documentation

##### 10.21.2.1 CFE\_ES\_DeleteGenCounter() `CFE_Status_t CFE_ES_DeleteGenCounter (CFE_ES_CounterId_t CounterId )`

Delete a generic counter.

#### Description

This routine deletes a previously registered generic counter.

#### Assumptions, External Events, and Notes:

None.

#### Parameters

in	<code>CounterId</code>	The Counter Id of the newly created counter.
----	------------------------	--

#### Returns

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

#### Return values

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

**See also**

[CFE\\_ES\\_IncrementGenCounter](#), [CFE\\_ES\\_RegisterGenCounter](#), [CFE\\_ES\\_SetGenCount](#), [CFE\\_ES\\_GetGenCount](#),  
[CFE\\_ES\\_GetGenCounterIDByName](#)

**10.21.2.2 CFE\_ES\_GetGenCount()** `CFE_Status_t CFE_ES_GetGenCount (`

```
    CFE_ES_CounterId_t CounterId,  
    uint32 * Count )
```

Get the specified generic counter count.

**Description**

This routine gets the value of a generic counter.

**Assumptions, External Events, and Notes:**

None.

**Parameters**

in	<i>CounterId</i>	The Counter to get the value from.
out	<i>Count</i>	Buffer to store value of the Counter (must not be null).

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_RegisterGenCounter](#), [CFE\\_ES\\_DeleteGenCounter](#), [CFE\\_ES\\_SetGenCount](#), [CFE\\_ES\\_IncrementGenCounter](#),  
[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	<i>CounterIdPtr</i>	Pointer to variable that is to receive the Counter's ID (must not be null).
in	<i>CounterName</i>	Pointer to null terminated character string containing a Counter name (must not be null).

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetGenCounterName](#)

**10.21.2.4 CFE\_ES\_GetGenCounterName()** *CFE\_Status\_t* CFE\_ES\_GetGenCounterName (

```
    char * CounterName,  
    CFE_ES_CounterId_t CounterId,  
    size_t BufferLength )
```

Get a Counter name for a specified Counter ID.

**Description**

This routine retrieves the cFE Counter name associated with a specified Counter ID.

**Assumptions, External Events, and Notes:**

In the case of a failure ([CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID](#)), an empty string is returned.

**Parameters**

out	<i>CounterName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Counter name.
in	<i>CounterId</i>	ID of Counter whose name is being requested.
in	<i>BufferLength</i>	The maximum number of characters, including the null terminator (must not be zero), that can be put into the <i>CounterName</i> buffer. This routine will truncate the name to this length, if necessary.

**Returns**

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

**Return values**

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

**Return values**

<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	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**

<a href="#">in</a>	<i>CounterId</i>	The Counter to be incremented.
--------------------	------------------	--------------------------------

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_ES\\_RegisterGenCounter](#), [CFE\\_ES\\_DeleteGenCounter](#), [CFE\\_ES\\_SetGenCount](#), [CFE\\_ES\\_GetGenCount](#), [CFE\\_ES\\_GetGenCounterIDByName](#)

**10.21.2.6 CFE\_ES\_RegisterGenCounter()** [CFE\\_Status\\_t](#) CFE\_ES\_RegisterGenCounter ( [CFE\\_ES\\_CounterId\\_t](#) \* CounterIdPtr, [const char](#) \* CounterName )

Register a generic counter.

**Description**

This routine registers a generic thread-safe counter which can be used for inter-task management.

**Assumptions, External Events, and Notes:**

The initial value of all newly registered counters is 0.

**Parameters**

out	<i>CounterIdPtr</i>	Buffer to store the Counter Id of the newly created counter (must not be null).
in	<i>CounterName</i>	The Name of the generic counter (must not be null).

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_ES_ERR_DUPLICATE_NAME</i>	Duplicate Name Error.
<i>CFE_ES_NO_RESOURCE_IDS_AVAILABLE</i>	Resource ID is not available.

**See also**

[CFE\\_ES\\_IncrementGenCounter](#), [CFE\\_ES\\_DeleteGenCounter](#), [CFE\\_ES\\_SetGenCount](#), [CFE\\_ES\\_GetGenCount](#), [CFE\\_ES\\_GetGenCounterIDByName](#)

**10.21.2.7 CFE\_ES\_SetGenCount()** *CFE\_Status\_t* CFE\_ES\_SetGenCount (   
     *CFE\_ES\_CounterId\_t* *CounterId*,  
     *uint32* *Count* )

Set the specified generic counter.

**Description**

This routine sets the specified generic counter to the specified value.

**Assumptions, External Events, and Notes:**

None.

**Parameters**

in	<i>CounterId</i>	The Counter to be set.
in	<i>Count</i>	The new value of the Counter.

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_RegisterGenCounter](#), [CFE\\_ES\\_DeleteGenCounter](#), [CFE\\_ES\\_IncrementGenCounter](#), [CFE\\_ES\\_GetGenCount](#),  
[CFE\\_ES\\_GetGenCounterIDByName](#)

## 10.22 cFE Registration APIs

### Functions

- **CFE\_Status\_t CFE\_EVS\_Register** (const void \*Filters, uint16 NumEventFilters, uint16 FilterScheme)  
*Register an application for receiving event services.*

#### 10.22.1 Detailed Description

#### 10.22.2 Function Documentation

##### 10.22.2.1 CFE\_EVS\_Register() [CFE\\_Status\\_t](#) CFE\_EVS\_Register (

```
    const void * Filters,
    uint16 NumEventFilters,
    uint16 FilterScheme )
```

Register an application for receiving event services.

#### Description

This routine registers an application with event services and allocates/initializes the internal data structures used to support this application's events. An application may not send events unless it has called this routine. The routine also accepts a filter array structure for applications requiring event filtering. In the current implementation of the EVS, only the binary filtering scheme is supported. See section TBD of the cFE Application Programmer's Guide for a description of the behavior of binary filters. Applications may call [CFE\\_EVS\\_Register](#) more than once, but each call will wipe out all filters registered by previous calls (filter registration is NOT cumulative).

#### Assumptions, External Events, and Notes:

Note: Event filters can be added, deleted or modified by ground commands. All filtering schemes include a default setting that results in no filtering (such as [CFE\\_EVS\\_NO\\_FILTER](#) for binary filters).

**Filter Scheme:** Binary

**Code:** CFE\_EVS\_EventFilter\_BINARY

**Filter Structure:**

```
typedef struct CFE_EVS_BinFilter {
    uint16 EventID,
    uint16 Mask ;
} CFE_EVS_BinFilter_t;
```

#### Parameters

in	<i>Filters</i>	Pointer to an array of event message filters, or NULL if no filtering is desired. The structure of an event message filter depends on the FilterScheme selected. (see Filter Schemes mentioned above)
in	<i>NumEventFilters</i>	The number of event message filters included in this call. This must be less than or equal to the maximum number of events allowed per application ( <a href="#">CFE_PLATFORM_EVS_MAX_EVENT_FILTERS</a> ).
in	<i>FilterScheme</i>	The event filtering scheme that this application will use. For the first implementation of the event services, only filter type <a href="#">CFE_EVS_EventFilter_BINARY</a> will be supported.

#### Returns

Execution status below or from [CFE\\_ES\\_GetAppID](#), see [cFE Return Code Defines](#)

**Return values**

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

Referenced by CS\_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	<code>EventID</code>	A numeric literal used to uniquely identify an application event. The <code>EventID</code> is defined and supplied by the application sending the event.
in	<code>EventType</code>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"><li>• <code>CFE_EVS_EventType_DEBUG</code></li><li>• <code>CFE_EVS_EventType_INFORMATION</code></li><li>• <code>CFE_EVS_EventType_ERROR</code></li><li>• <code>CFE_EVS_EventType_CRITICAL</code></li></ul>

**Parameters**

in	<b>Spec</b>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter <code>CFE_MISSION_EVS_MAX_MESSAGE_LENGTH</code> . Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.
----	-------------	--

**Returns**

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

**Return values**

<a href="#"><code>CFE_SUCCESS</code></a>	Successful execution.
<a href="#"><code>CFE_EVS_APP_NOT_REGISTERED</code></a>	Application Not Registered.
<a href="#"><code>CFE_EVS_APP_ILLEGAL_APP_ID</code></a>	Illegal Application ID.
<a href="#"><code>CFE_EVS_INVALID_PARAMETER</code></a>	Invalid Pointer.

**See also**

[CFE\\_EVS\\_SendEventWithAppID](#), [CFE\\_EVS\\_SendTimedEvent](#)

Referenced by `CS_AppInit()`, `CS_AppMain()`, `CS_AppPipe()`, `CS_BackgroundApp()`, `CS_BackgroundCfeCore()`, `CS_BackgroundCheckCycle()`, `CS_BackgroundEeprom()`, `CS_BackgroundMemory()`, `CS_BackgroundOS()`, `CS_BackgroundTables()`, `CS_CancelOneShotCmd()`, `CS_CheckRecomputeOneshot()`, `CS_ComputeApp()`, `CS_ComputeTables()`, `CS_CreateRestoreStatesFromCDS()`, `CS_DisableAllCSCmd()`, `CS_DisableAppCmd()`, `CS_DisableCfeCoreCmd()`, `CS_DisableEepromCmd()`, `CS_DisableEntryIDEepromCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_DisableMemoryCmd()`, `CS_DisableNameAppCmd()`, `CS_DisableNameTablesCmd()`, `CS_DisableOSCmd()`, `CS_DisableTablesCmd()`, `CS_EnableAllCSCmd()`, `CS_EnableAppCmd()`, `CS_EnableCfeCoreCmd()`, `CS_EnableEepromCmd()`, `CS_EnableEntryIDEepromCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_EnableMemoryCmd()`, `CS_EnableNameAppCmd()`, `CS_EnableNameTablesCmd()`, `CS_EnableOSCmd()`, `CS_EnableTablesCmd()`, `CS_GetEntryIDEepromCmd()`, `CS_GetEntryIDMemoryCmd()`, `CS_HandleRoutineTableUpdates()`, `CS_HandleTableUpdate()`, `CS_HousekeepingCmd()`, `CS_InitAllTables()`, `CS_InitSegments()`, `CS_NoopCmd()`, `CS_OneShotChildTask()`, `CS_OneShotCmd()`, `CS_ProcessCmd()`, `CS_ProcessNewAppDefinitionTable()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, `CS_ProcessNewTablesDefinitionTable()`, `CS_RecomputeAppChildTask()`, `CS_RecomputeBaselineAppCmd()`, `CS_RecomputeBaselineCfeCoreCmd()`, `CS_RecomputeBaselineEepromCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeBaselineOSCmd()`, `CS_RecomputeBaselineTablesCmd()`, `CS_RecomputeEepromMemoryChildTask()`, `CS_RecomputeTablesChildTask()`, `CS_ReportBaselineAppCmd()`, `CS_ReportBaselineCfeCoreCmd()`, `CS_ReportBaselineEntryIDEepromCmd()`, `CS_ReportBaselineEntryIDMemoryCmd()`, `CS_ReportBaselineOSCmd()`, `CS_ReportBaselineTablesCmd()`, `CS_ResetCmd()`, `CS_SblInit()`, `CS_TableInit()`, `CS_UpdateCDS()`, `CS_ValidateAppChecksumDefinitionTable()`, `CS_ValidateEepromChecksumDefinitionTable()`, `CS_ValidateMemoryChecksumDefinitionTable()`, `CS_ValidateTablesChecksumDefinitionTable()`, and `CS_VerifyCmdLength()`.

**10.23.2.2 CFE\_EVS\_SendEventWithAppID()**    `CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithAppID (`

```

    uint16 EventID,
    uint16 EventType,
    CFE_ES_AppId_t AppID,
```

```
    const char * Spec,  
    ... )
```

Generate a software event given the specified Application ID.

#### Description

This routine generates a software event message. If the EventID is not filtered, the event will be sent as a software bus message, optionally logged in the local event log, and optionally sent as an ASCII text string out the enabled output port(s). Note that this function should really only be used from within an API in order to preserve the context of an Application's event. In general, [CFE\\_EVS\\_SendEvent](#) should be used.

#### Assumptions, External Events, and Notes:

The Application ID must correspond to a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE\\_ES\\_WriteToSysLog](#) can be used for reporting.

#### Parameters

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The <i>EventID</i> is defined and supplied by the application sending the event.
in	<i>EventType</i>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"><li>• <a href="#">CFE_EVS_EventType_DEBUG</a></li><li>• <a href="#">CFE_EVS_EventType_INFORMATION</a></li><li>• <a href="#">CFE_EVS_EventType_ERROR</a></li><li>• <a href="#">CFE_EVS_EventType_CRITICAL</a></li></ul>
in	<i>AppID</i>	The Application ID from which the event message should appear.
in	<i>Spec</i>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter <a href="#">CFE_MISSION_EVS_MAX_MESSAGE_LENGTH</a> . 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

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_EVS_APP_NOT_REGISTERED</a>	Application Not Registered.
<a href="#">CFE_EVS_APP_ILLEGAL_APP_ID</a>	Illegal Application ID.
<a href="#">CFE_EVS_INVALID_PARAMETER</a>	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(Event (`

```
    CFE_TIME_SysTime_t Time,
    uint16 EventID,
    uint16 EventType,
    const char * Spec,
    ...
)
```

Generate a software event with a specific time tag.

**Description**

This routine is the same as [CFE\\_EVS\\_SendEvent](#) except that the caller specifies the event time instead of having the EVS use the current spacecraft time. This routine should be used in situations where an error condition is detected at one time, but the event message is reported at a later time.

**Assumptions, External Events, and Notes:**

This API only works within the context of a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE\\_ES\\_WriteToSysLog](#) can be used for reporting.

**Parameters**

in	<i>Time</i>	The time to include in the event. This will usually be a time returned by the function <a href="#">CFE_TIME_GetTime</a> .
in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The <i>EventID</i> is defined and supplied by the application sending the event.
in	<i>EventType</i>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"> <li>• <a href="#">CFE_EVS_EventType_DEBUG</a></li> <li>• <a href="#">CFE_EVS_EventType_INFORMATION</a></li> <li>• <a href="#">CFE_EVS_EventType_ERROR</a></li> <li>• <a href="#">CFE_EVS_EventType_CRITICAL</a></li> </ul>
in	<i>Spec</i>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter <a href="#">CFE_MISSION_EVS_MAX_MESSAGE_LENGTH</a> . 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**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_EVS_APP_NOT_REGISTERED</i>	Application Not Registered.
<i>CFE_EVS_APP_ILLEGAL_APP_ID</i>	Illegal Application ID.
<i>CFE_EVS_INVALID_PARAMETER</i>	Invalid Pointer.

**See also**

[CFE\\_EVS\\_SendEvent](#), [CFE\\_EVS\\_SendEventWithAppID](#)

## 10.24 cFE Reset Event Filter APIs

### Functions

- [CFE\\_Status\\_t CFE\\_EVS\\_ResetFilter \(uint16 EventID\)](#)  
*Resets the calling application's event filter for a single event ID.*
- [CFE\\_Status\\_t CFE\\_EVS\\_ResetAllFilters \(void\)](#)  
*Resets all of the calling application's event filters.*

#### 10.24.1 Detailed Description

#### 10.24.2 Function Documentation

##### 10.24.2.1 CFE\_EVS\_ResetAllFilters() [CFE\\_Status\\_t CFE\\_EVS\\_ResetAllFilters \(void \)](#)

Resets all of the calling application's event filters.

#### Description

This routine resets all the calling application's event filter counters to zero, providing a quick and convenient method for resetting event filters.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status below or from [CFE\\_ES\\_GetAppID](#), see [cFE Return Code Defines](#)

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_EVS_APP_NOT_REGISTERED</a>	Application Not Registered.
<a href="#">CFE_EVS_APP_ILLEGAL_APP_ID</a>	Illegal Application ID.

#### See also

[CFE\\_EVS\\_ResetFilter](#)

##### 10.24.2.2 CFE\_EVS\_ResetFilter() [CFE\\_Status\\_t CFE\\_EVS\\_ResetFilter \(uint16 EventID \)](#)

Resets the calling application's event filter for a single event ID.

#### Description

Resets the filter such that the next event is treated like the first. For example, if the filter was set to only send the first event, the next event following the reset would be sent.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The Event ID is defined and supplied by the application sending the event.
----	----------------	--

**Returns**

Execution status below or from [CFE\\_ES\\_GetAppID](#), see [cFE Return Code Defines](#)

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_EVS_APP_NOT_REGISTERED</a>	Application Not Registered.
<a href="#">CFE_EVS_APP_ILLEGAL_APP_ID</a>	Illegal Application ID.
<a href="#">CFE_EVS_EVT_NOT_REGISTERED</a>	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	<code>Hdr</code>	Pointer to a variable of type <code>CFE_FS_Header_t</code> that will be cleared and initialized
in	<code>Description</code>	Initializes Header's Description (must not be null)
in	<code>SubType</code>	Initializes Header's SubType

#### See also

[CFE\\_FS\\_WriteHeader](#)

**10.25.2.2 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**

<i>out</i>	<i>Hdr</i>	Pointer to a variable of type <a href="#">CFE_FS_Header_t</a> (must not be null) that will be filled with the contents of the Standard cFE File Header. *Hdr is the contents of the Standard cFE File Header for the specified file.
<i>in</i>	<i>FileDes</i>	File Descriptor obtained from a previous call to <a href="#">OS_OpenCreate</a> that is associated with the file whose header is to be read.

**Returns**

Bytes read or error status from OSAL

**Return values**

<a href="#">CFE_FS_BAD_ARGUMENT</a>	Bad Argument.
-------------------------------------	---------------

**Note**

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in [cFE Return Code Defines](#).

**See also**

[CFE\\_FS\\_WriteHeader](#)

**10.25.2.3 CFE\_FS\_SetTimestamp()** [CFE\\_Status\\_t](#) CFE\_FS\_SetTimestamp (   
     *osal\_id\_t* *FileDes*,  
     [CFE\\_TIME\\_SysTime\\_t](#) *NewTimestamp* )

Modifies the Time Stamp field in the Standard cFE File Header for the specified file.

**Description**

This API will modify the [timestamp](#) found in the Standard cFE File Header of the specified file. The timestamp will be replaced with the time specified by the caller.

**Assumptions, External Events, and Notes:**

1. The File has already been successfully opened using [OS\\_OpenCreate](#) and the caller has a legitimate File Descriptor.
2. The *NewTimestamp* field has been filled appropriately by the Application.
3. File offset behavior: Agnostic on entry since it will move the offset, on success the offset will be at the end of the time stamp, undefined offset behavior for error cases.

**Parameters**

in	<i>FileDes</i>	File Descriptor obtained from a previous call to <a href="#">OS_OpenCreate</a> that is associated with the file whose header is to be read.
in	<i>NewTimestamp</i>	A <a href="#">CFE_TIME_SysTime_t</a> 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**

<a href="#">CFE_STATUS_EXTERNAL_RESOURCE_FAIL</a>	(return value only verified in coverage test) External failure.
<a href="#">CFE_SUCCESS</a>	Successful execution.

**Note**

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in [cFE Return Code Defines](#).

**10.25.2.4 CFE\_FS\_WriteHeader()** [CFE\\_Status\\_t](#) CFE\_FS\_WriteHeader (   
     [osal\\_id\\_t](#) *FileDes*,  
     [CFE\\_FS\\_Header\\_t](#) \* *Hdr* )

Write the specified Standard cFE File Header to the specified file.

**Description**

This API will output the specified [CFE\\_FS\\_Header\\_t](#) variable, with some fields automatically updated, to the specified file as the Standard cFE File Header. This API will automatically populate the following fields in the specified [CFE\\_FS\\_Header\\_t](#):

1. [ContentType](#) - Filled with 0x63464531 ('cFE1')
2. [Length](#) - Filled with the sizeof(CFE\_FS\_Header\_t)
3. [SpacecraftID](#) - Filled with the Spacecraft ID
4. [ProcessorID](#) - Filled with the Processor ID
5. [ApplicationID](#) - Filled with the Application ID
6. [TimeSeconds](#) - Filled with the Time, in seconds, as obtained by [CFE\\_TIME\\_GetTime](#)
7. [TimeSubSeconds](#) - Filled with the Time, subseconds, as obtained by [CFE\\_TIME\\_GetTime](#)

**Assumptions, External Events, and Notes:**

1. The File has already been successfully opened using [OS\\_OpenCreate](#) and the caller has a legitimate File Descriptor.
2. The SubType field has been filled appropriately by the Application.
3. The Description field has been filled appropriately by the Application.
4. File offset behavior: Agnostic on entry since it will move the offset to the start of the file, on success the offset will be at the end of the header, undefined offset behavior for error cases.

**Parameters**

in	<i>FileDes</i>	File Descriptor obtained from a previous call to <a href="#">OS_OpenCreate</a> that is associated with the file whose header is to be read.
out	<i>Hdr</i>	Pointer to a variable of type <a href="#">CFE_FS_Header_t</a> (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**

<a href="#">CFE_FS_BAD_ARGUMENT</a>	Bad Argument.
-------------------------------------	---------------

**Note**

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in [cFE Return Code Defines](#).

**See also**

[CFE\\_FS\\_ReadHeader](#)

## 10.26 cFE File Utility APIs

### Functions

- `const char * CFE_FS_GetDefaultMountPoint (CFE_FS_FileCategory_t FileCategory)`  
*Get the default virtual mount point for a file category.*
- `const char * CFE_FS_GetDefaultExtension (CFE_FS_FileCategory_t FileCategory)`  
*Get the default filename extension for a file category.*
- `int32 CFE_FS_ParseInputFileNameEx (char *OutputBuffer, const char *InputBuffer, size_t OutputBufSize, size_t InputBufSize, const char *DefaultInput, const char *DefaultPath, const char *DefaultExtension)`  
*Parse a filename input from an input buffer into a local buffer.*
- `int32 CFE_FS_ParseInputFileName (char *OutputBuffer, const char *InputName, size_t OutputBufSize, CFE_FS_FileCategory_t FileCategory)`  
*Parse a filename string from the user into a local buffer.*
- `CFE_Status_t CFE_FS_ExtractFilenameFromPath (const char *OriginalPath, char *FileNameOnly)`  
*Extracts the filename from a unix style path and filename string.*
- `int32 CFE_FS_BackgroundFileDumpRequest (CFE_FS_FileWriteMetaData_t *Meta)`  
*Register a background file dump request.*
- `bool CFE_FS_BackgroundFileDumplsPending (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\_BackgroundFileDumplsPending()** `bool CFE_FS_BackgroundFileDumpIsPending (const CFE_FS_FileWriteMetaData_t * Meta )`

Query if a background file write request is currently pending.

#### Description

This returns "true" while the request is on the background work queue. This returns "false" once the request is complete and removed from the queue.

#### Assumptions, External Events, and Notes:

None

#### Parameters

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

#### Returns

boolean value indicating if request is already pending

#### Return values

<code>true</code>	if request is pending
<code>false</code>	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

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

#### Returns

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

#### Return values

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

**10.26.2.3 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

<code>in</code>	<code>OriginalPath</code>	The original path (must not be null)
<code>out</code>	<code>FileNameOnly</code>	The filename that is extracted from the path (must not be null)

**Returns**

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

**Return values**

<code>CFE_FS_BAD_ARGUMENT</code>	Bad Argument.
<code>CFE_FS_FNAME_TOO_LONG</code>	Filename Too Long.
<code>CFE_FS_INVALID_PATH</code>	Invalid Path.
<code>CFE_SUCCESS</code>	Successful execution.

**10.26.2.4 CFE\_FS\_GetDefaultExtension()** `const char* CFE_FS_GetDefaultExtension ( CFE_FS_FileCategory_t FileCategory )`

Get the default filename extension for a file category.

Certain file types may have an extension that varies from system to system. This is primarily an issue for application modules which are ".so" on Linux systems, ".dll" on Windows, ".o" on VxWorks, ".obj" on RTEMS, and so on.

This uses a combination of compile-time configuration and hints from the build environment to get the default/expected extension for a given file category.

**Returns**

String containing the extension

**Return values**

<code>NULL</code>	if no default extension is known for the given file category
-------------------	--

**10.26.2.5 CFE\_FS\_GetDefaultMountPoint()** `const char* CFE_FS_GetDefaultMountPoint ( CFE_FS_FileCategory_t FileCategory )`

Get the default virtual mount point for a file category.

Certain classes of files generally reside in a common directory, mainly either the persistent storage (/cf typically) or ram disk (/ram typically).

Ephemeral status files are generally in the ram disk while application modules and scripts are generally in the persistent storage.

This returns the expected directory for a given class of files in the form of a virtual OSAL mount point string.

**Returns**

String containing the mount point

**Return values**

<code>NULL</code>	if no mount point is known for the given file category
-------------------	--

**10.26.2.6 CFE\_FS\_ParseInputFileName()** `int32 CFE_FS_ParseInputFileName ( char * OutputBuffer,`

```
const char * InputName,
size_t OutputBufSize,
CFE_FS_FileCategory_t FileCategory)
```

Parse a filename string from the user into a local buffer.

#### Description

Simplified API for [CFE\\_FS\\_ParseInputFileNameEx\(\)](#) where input is always known to be a non-empty, null terminated string and the fixed-length input buffer not needed. For instance this may be used where the input is a fixed string from cfe\_platform\_cfg.h or similar.

#### Assumptions, External Events, and Notes:

The parameters are organized such that this is basically like strncpy() with an extra argument, and existing file name accesses which use a direct copy can easily change to use this instead.

#### See also

[CFE\\_FS\\_ParseInputFileNameEx\(\)](#)

#### Parameters

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

#### Returns

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

---

```
10.26.2.7 CFE_FS_ParseInputFileNameEx() int32 CFE_FS_ParseInputFileNameEx (
    char * OutputBuffer,
    const char * InputBuffer,
    size_t OutputBufSize,
    size_t InputBufSize,
    const char * DefaultInput,
    const char * DefaultPath,
    const char * DefaultExtension)
```

Parse a filename input from an input buffer into a local buffer.

#### Description

This provides a more user friendly way to specify file names, using default values for the path and extension, which can vary from system to system.

If InputBuffer is null or its length is zero, then DefaultInput is used as if it was the content of the input buffer.  
If either the pathname or extension is missing from the input, it will be added from defaults, with the complete fully-qualified filename stored in the output buffer.

**Assumptions, External Events, and Notes:**

1. The paths and filenames used here are the standard unix style filenames separated by "/" (path) and "." (extension) characters.
2. Input Buffer has a fixed max length. Parsing will not exceed InputBufSize, and does not need to be null terminated. However parsing will stop at the first null char, when the input is shorter than the maximum.

**Parameters**

out	<i>OutputBuffer</i>	Buffer to store result (must not be null).
in	<i>InputBuffer</i>	A input buffer that may contain a file name (e.g. from command) (must not be null).
in	<i>OutputBufSize</i>	Maximum Size of output buffer (must not be zero).
in	<i>InputBufSize</i>	Maximum Size of input buffer.
in	<i>DefaultInput</i>	Default value to use for input if InputBffer is empty
in	<i>DefaultPath</i>	Default value to use for pathname if omitted from input
in	<i>DefaultExtension</i>	Default value to use for extension if omitted from input

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

<a href="#">CFE_FS_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_FS_FNAME_TOO_LONG</a>	Filename Too Long.
<a href="#">CFE_FS_INVALID_PATH</a>	Invalid Path.
<a href="#">CFE_SUCCESS</a>	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.*
- `CFE_Status_t CFE_MSG_UpdateHeader (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)`  
*Set/compute all dynamically-updated headers on a message.*

#### 10.27.1 Detailed Description

#### 10.27.2 Function Documentation

##### 10.27.2.1 CFE\_MSG\_Init() `CFE_Status_t CFE_MSG_Init (`

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_SB_MsgId_t MsgId,  
    CFE_MSG_Size_t Size )
```

Initialize a message.

#### Description

This routine initialize a message. The entire message is set to zero (based on size), defaults are set, then the size and bits from MsgId are set.

#### Parameters

out	<code>MsgPtr</code>	A pointer to the buffer that contains the message (must not be null).
in	<code>MsgId</code>	MsgId that corresponds to message
in	<code>Size</code>	Total size of the message (used to set length field)

#### Returns

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

#### Return values

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

Referenced by CS\_SbInit().

##### 10.27.2.2 CFE\_MSG\_UpdateHeader() `CFE_Status_t CFE_MSG_UpdateHeader (`

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_SequenceCount_t SeqCnt )
```

Set/compute all dynamically-updated headers on a message.

### Description

This routine updates all dynamic header fields on a message, and is typically invoked via SB just prior to broadcasting the message. Dynamic headers include are values that should be computed/updated per message, including:

- the sequence number
- the timestamp, if present
- any error control or checksum fields, if present

The MSG module implementation determines which header fields meet this criteria and how they should be computed.

### Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>SeqCnt</i>	The current sequence number from the message route

### Returns

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

### Return values

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

## 10.28 cFE Message Primary Header APIs

### Functions

- `CFE_Status_t CFE_MSG_GetSize (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t *Size)`  
*Gets the total size of a message.*
- `CFE_Status_t CFE_MSG_SetSize (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t Size)`  
*Sets the total size of a message.*
- `CFE_Status_t CFE_MSG.GetType (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t *Type)`  
*Gets the message type.*
- `CFE_Status_t CFE_MSG_SetType (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t Type)`  
*Sets the message type.*
- `CFE_Status_t CFE_MSG_GetHeaderVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t *Version)`  
*Gets the message header version.*
- `CFE_Status_t CFE_MSG_SetHeaderVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t Version)`  
*Sets the message header version.*
- `CFE_Status_t CFE_MSG_GetHasSecondaryHeader (const CFE_MSG_Message_t *MsgPtr, bool *HasSecondary)`  
*Gets the message secondary header boolean.*
- `CFE_Status_t CFE_MSG_SetHasSecondaryHeader (CFE_MSG_Message_t *MsgPtr, bool HasSecondary)`  
*Sets the message secondary header boolean.*
- `CFE_Status_t CFE_MSG_GetApld (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t *Apld)`  
*Gets the message application ID.*
- `CFE_Status_t CFE_MSG_SetApld (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t Apld)`  
*Sets the message application ID.*
- `CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)`  
*Gets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)`  
*Sets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)`  
*Gets the message sequence count.*
- `CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)`  
*Sets the message sequence count.*
- `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)`  
*Gets the next sequence count value (rolls over if appropriate)*

#### 10.28.1 Detailed Description

#### 10.28.2 Function Documentation

```
10.28.2.1 CFE_MSG_GetApId() CFE_Status_t CFE_MSG_GetApId (
    const CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_ApId_t * ApId )
```

Gets the message application ID.

#### Description

This routine gets the message application ID.

#### Parameters

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

#### Returns

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

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

```
10.28.2.2 CFE_MSG_GetHasSecondaryHeader() CFE_Status_t CFE_MSG_GetHasSecondaryHeader (
    const CFE_MSG_Message_t * MsgPtr,
    bool * HasSecondary )
```

Gets the message secondary header boolean.

#### Description

This routine gets the message secondary header boolean.

#### Parameters

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

#### Returns

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

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
out	<i>Version</i>	Header version (must not be null)

#### Returns

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

#### Return values

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

**10.28.2.4 CFE\_MSG\_GetNextSequenceCount()** `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (`  
    `CFE_MSG_SequenceCount_t SeqCnt )`

Gets the next sequence count value (rolls over if appropriate)

#### Description

Abstract method to get the next valid sequence count value. Will roll over to zero for any input value greater than or equal to the maximum possible sequence count value given the field in the header.

#### Parameters

in	<i>SeqCnt</i>	Sequence count
----	---------------	----------------

#### Returns

The next valid sequence count value

**10.28.2.5 CFE\_MSG\_GetSegmentationFlag()** `CFE_Status_t CFE_MSG_GetSegmentationFlag (`  
    `const CFE_MSG_Message_t * MsgPtr,`  
    `CFE_MSG_SegmentationFlag_t * SegFlag )`

Gets the message segmentation flag.

#### Description

This routine gets the message segmentation flag

**Parameters**

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

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_MSG_BAD_ARGUMENT</i></a>	Error - bad argument.

**10.28.2.6 CFE\_MSG\_GetSequenceCount()** [\*CFE\\_Status\\_t\*](#) *CFE\_MSG\_GetSequenceCount* (

```
const CFE\_MSG\_Message\_t * MsgPtr,
CFE\_MSG\_SequenceCount\_t * SeqCnt )
```

Gets the message sequence count.

**Description**

This routine gets the message sequence count.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_MSG_BAD_ARGUMENT</i></a>	Error - bad argument.

**10.28.2.7 CFE\_MSG.GetSize()** [\*CFE\\_Status\\_t\*](#) *CFE\_MSG\_GetSize* (

```
const CFE\_MSG\_Message\_t * MsgPtr,
CFE\_MSG\_Size\_t * Size )
```

Gets the total size of a message.

**Description**

This routine gets the total size of the message.

**Parameters**

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

**Returns**

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

**Return values**

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

Referenced by CS\_BackgroundCheckCycle(), CS\_HousekeepingCmd(), and CS\_VerifyCmdLength().

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

**Returns**

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

**Return values**

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

**10.28.2.9 CFE\_MSG\_SetApId()** *CFE\_Status\_t* CFE\_MSG\_SetApId (

```
CFE_MSG_Message_t * MsgPtr,
```

```
CFE_MSG_ApId_t ApId )
```

Sets the message application ID.

**Description**

This routine sets the message application ID. Typically set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

**Parameters**

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

**Returns**

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

**Return values**

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

**10.28.2.10 CFE\_MSG\_SetHasSecondaryHeader()** *CFE\_Status\_t* CFE\_MSG\_SetHasSecondaryHeader (

```
    CFE_MSG_Message_t * MsgPtr,
    bool HasSecondary )
```

Sets the message secondary header boolean.

**Description**

This routine sets the message secondary header boolean. Typically only set within message initialization and not used by APPs.

**Parameters**

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

**Returns**

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

**Return values**

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

**10.28.2.11 CFE\_MSG\_SetHeaderVersion()** *CFE\_Status\_t* CFE\_MSG\_SetHeaderVersion (

```
    CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_HeaderVersion_t Version )
```

Sets the message header version.

**Description**

This routine sets the message header version. Typically only set within message initialization and not used by APPs.

**Parameters**

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message.
in	<i>Version</i>	Header version

**Returns**

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

**Return values**

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

**10.28.2.12 CFE\_MSG\_SetSegmentationFlag()** *CFE\_Status\_t* CFE\_MSG\_SetSegmentationFlag (   
     *CFE\_MSG\_Message\_t* \* *MsgPtr*,  
     *CFE\_MSG\_SegmentationFlag\_t* *SegFlag* )

Sets the message segmentation flag.

**Description**

This routine sets the message segmentation flag.

**Parameters**

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

**Returns**

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

**Return values**

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

**Returns**

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

**Return values**

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

**Returns**

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

**Return values**

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

**10.28.2.15 CFE\_MSG\_SetType()** *CFE\_Status\_t* CFE\_MSG\_SetType (

```
CFE_MSG_Message_t * MsgPtr,
```

```
CFE_MSG_Type_t Type )
```

Sets the message type.

**Description**

This routine sets the message type.

**Parameters**

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

**Returns**

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

**Return values**

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

## 10.29 cFE Message Extended Header APIs

### Functions

- `CFE_Status_t CFE_MSG_GetEDSVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t *Version)`  
*Gets the message EDS version.*
- `CFE_Status_t CFE_MSG_SetEDSVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t Version)`  
*Sets the message EDS version.*
- `CFE_Status_t CFE_MSG_GetEndian (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t *Endian)`  
*Gets the message endian.*
- `CFE_Status_t CFE_MSG_SetEndian (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t Endian)`  
*Sets the message endian.*
- `CFE_Status_t CFE_MSG_GetPlaybackFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t *PlayFlag)`  
*Gets the message playback flag.*
- `CFE_Status_t CFE_MSG_SetPlaybackFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t PlayFlag)`  
*Sets the message playback flag.*
- `CFE_Status_t CFE_MSG_GetSubsystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t *Subsystem)`  
*Gets the message subsystem.*
- `CFE_Status_t CFE_MSG_SetSubsystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t Subsystem)`  
*Sets the message subsystem.*
- `CFE_Status_t CFE_MSG_GetSystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t *System)`  
*Gets the message system.*
- `CFE_Status_t CFE_MSG_SetSystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t System)`  
*Sets the message system.*

### 10.29.1 Detailed Description

### 10.29.2 Function Documentation

**10.29.2.1 CFE\_MSG\_GetEDSVersion()** `CFE_Status_t CFE_MSG_GetEDSVersion (`  
`const CFE_MSG_Message_t * MsgPtr,`  
`CFE_MSG_EDSVersion_t * Version )`

Gets the message EDS version.

#### Description

This routine gets the message EDS version.

#### Parameters

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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**

<a href="#">in</a>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<a href="#">out</a>	<i>Subsystem</i>	Subsystem (must not be null)

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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**

<a href="#">in</a>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<a href="#">out</a>	<i>System</i>	System (must not be null)

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

**10.29.2.8 CFE\_MSG\_SetPlaybackFlag()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetPlaybackFlag (

[CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr*,  
[CFE\\_MSG\\_PlaybackFlag\\_t](#) *PlayFlag* )

Sets the message playback flag.

**Description**

This routine sets the message playback flag.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

**10.29.2.9 CFE\_MSG\_SetSubsystem()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetSubsystem (

[CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr*,  
[CFE\\_MSG\\_Subsystem\\_t](#) *Subsystem* )

Sets the message subsystem.

**Description**

This routine sets the message subsystem. Some bits may be set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	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**

<a href="#">in, out</a>	<a href="#">MsgPtr</a>	A pointer to the buffer that contains the message (must not be null).
<a href="#">in</a>	<a href="#">System</a>	System

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

## 10.30 cFE Message Secondary Header APIs

### Functions

- `CFE_Status_t CFE_MSG_GenerateChecksum (CFE_MSG_Message_t *MsgPtr)`  
*Calculates and sets the checksum of a message.*
- `CFE_Status_t CFE_MSG_ValidateChecksum (const CFE_MSG_Message_t *MsgPtr, bool *isValid)`  
*Validates the checksum of a message.*
- `CFE_Status_t CFE_MSG_SetFcnCode (CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t FcnCode)`  
*Sets the function code field in a message.*
- `CFE_Status_t CFE_MSG_GetFcnCode (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t *FcnCode)`  
*Gets the function code field from a message.*
- `CFE_Status_t CFE_MSG_GetMsgTime (const CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t *Time)`  
*Gets the time field from a message.*
- `CFE_Status_t CFE_MSG_SetMsgTime (CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t NewTime)`  
*Sets the time field in a message.*

#### 10.30.1 Detailed Description

#### 10.30.2 Function Documentation

**10.30.2.1 CFE\_MSG\_GenerateChecksum()** `CFE_Status_t CFE_MSG_GenerateChecksum (`  
`CFE_MSG_Message_t * MsgPtr )`

Calculates and sets the checksum of a message.

##### Description

This routine calculates the checksum of a message according to an implementation-defined algorithm. Then, it sets the checksum field in the message with the calculated value. The contents and location of this field will depend on the underlying implementation of messages. It may be a checksum, a CRC, or some other algorithm.

##### Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a checksum field, then this routine will return `CFE_MSG_WRONG_MSG_TYPE`

##### Parameters

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

##### Returns

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

##### Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.
<code>CFE_MSG_WRONG_MSG_TYPE</code>	Error - wrong type.

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

#### Returns

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

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.
<a href="#">CFE_MSG_WRONG_MSG_TYPE</a>	Error - wrong type.

Referenced by CS\_BackgroundCheckCycle(), CS\_HousekeepingCmd(), CS\_ProcessCmd(), and CS\_VerifyCmdLength().

```
10.30.2.3 CFE_MSG_GetMsgTime() CFE_Status_t CFE_MSG_GetMsgTime (
    const CFE_MSG_Message_t * MsgPtr,
    CFE_TIME_SysTime_t * Time )
```

Gets the time field from a message.

#### Description

This routine gets the time from a message.

#### Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a time field, then this routine will set Time to zero and return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#)
- Note default implementation of command messages do not have a time field.

**Parameters**

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

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

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

**10.30.2.4 CFE\_MSG\_SetFcnCode()** *CFE\_Status\_t* CFE\_MSG\_SetFcnCode (

```
    CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_FcnCode_t FcnCode )
```

Sets the function code field in a message.

**Description**

This routine sets the function code of a message.

**Assumptions, External Events, and Notes:**

- If the underlying implementation of messages does not include a function code field, then this routine will do nothing to the message contents and will return *CFE\_MSG\_WRONG\_MSG\_TYPE*.

**Parameters**

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

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

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

**10.30.2.5 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	<i>MsgPtr</i>	A pointer to the message (must not be null).
in	<i>NewTime</i>	The time to include in the message. This will usually be a time from <a href="#">CFE_TIME_GetTime</a> .

#### Returns

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

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.
<a href="#">CFE_MSG_WRONG_MSG_TYPE</a>	Error - wrong type.

### 10.30.2.6 CFE\_MSG\_ValidateChecksum()

```
CFE_Status_t CFE_MSG_ValidateChecksum (
    const CFE_MSG_Message_t * MsgPtr,
    bool * IsValid )
```

Validates the checksum of a message.

#### Description

This routine validates the checksum of a message according to an implementation-defined algorithm.

#### Assumptions, External Events, and Notes:

- If the underlying implementation of messages does not include a checksum field, then this routine will return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#) and set the IsValid parameter false.

#### Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null). This must point to the first byte of the message header.
----	---------------	--

**Parameters**

<code>out</code>	<code>isValid</code>	Checksum validation result (must not be null) <ul style="list-style-type: none"><li>• true - valid</li><li>• false - invalid or not supported/implemented</li></ul>
------------------	----------------------	---

**Returns**

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

**Return values**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_MSG_BAD_ARGUMENT</code>	Error - bad argument.
<code>CFE_MSG_WRONG_MSG_TYPE</code>	Error - wrong type.

## 10.31 cFE Message Id APIs

### Functions

- `CFE_Status_t CFE_MSG_GetMsgId (const CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t *MsgId)`  
*Gets the message id from a message.*
- `CFE_Status_t CFE_MSG_SetMsgId (CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId)`  
*Sets the message id bits in a message.*
- `CFE_Status_t CFE_MSG.GetTypeFromMsgId (CFE_SB_MsgId_t MsgId, CFE_MSG_Type_t *Type)`  
*Gets message type using message ID.*

#### 10.31.1 Detailed Description

#### 10.31.2 Function Documentation

##### 10.31.2.1 CFE\_MSG\_GetMsgId() `CFE_Status_t CFE_MSG_GetMsgId (`

```
    const CFE_MSG_Message_t * MsgPtr,
    CFE_SB_MsgId_t * MsgId )
```

Gets the message id from a message.

#### Description

This routine gets the message id from a message. The message id is a hash of bits in the message header, used by the software bus for routing. Message id needs to be unique for each endpoint in the system.

#### Parameters

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

#### Returns

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

#### Return values

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

Referenced by CS\_AppPipe(), CS\_BackgroundCheckCycle(), CS\_HousekeepingCmd(), CS\_ProcessCmd(), and CS\_VerifyCmdLength().

##### 10.31.2.2 CFE\_MSG.GetTypeFromMsgId() `CFE_Status_t CFE_MSG.GetTypeFromMsgId (`

```
    CFE_SB_MsgId_t MsgId,
    CFE_MSG_Type_t * Type )
```

Gets message type using message ID.

**Description**

This routine gets the message type using the message ID

**Parameters**

in	<i>MsgId</i>	Message id
out	<i>Type</i>	Message type (must not be null)

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

**10.31.2.3 CFE\_MSG\_SetMsgId()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetMsgId (

```
    CFE_MSG_Message_t * MsgPtr,
    CFE_SB_MsgId_t MsgId )
```

Sets the message id bits in a message.

**Description**

This routine sets the message id bits in a message. The message id is a hash of bits in the message header, used by the software bus for routing. Message id needs to be unique for each endpoint in the system.

**Note**

This API only sets the bits in the header that make up the message ID. No other values in the header are modified.

The user should ensure that this function is only called with a valid MsgId parameter value. If called with an invalid value, the results are implementation-defined. The implementation may or may not return the error code [CFE\\_MSG\\_BAD\\_ARGUMENT](#) in this case.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

## 10.32 cFE Message Checking APIs

### Functions

- `CFE_Status_t CFE_MSG_Verify (const CFE_MSG_Message_t *MsgPtr, bool *VerifyStatus)`  
*Checks message headers against expected values.*

#### 10.32.1 Detailed Description

#### 10.32.2 Function Documentation

**10.32.2.1 CFE\_MSG\_Verify()** `CFE_Status_t CFE_MSG_Verify (`  
    `const CFE_MSG_Message_t * MsgPtr,`  
    `bool * VerifyStatus )`

Checks message headers against expected values.

#### Description

This routine validates that any error-control field(s) in the message header matches the expected value.

The specific function of this API is entirely dependent on the header fields and may be a no-op if no error checking is implemented. In that case, it will always output "true".

#### Parameters

in	<code>MsgPtr</code>	Message Pointer (must not be null)
out	<code>VerifyStatus</code>	Output variable to be set to verification result (must not be null)

#### Returns

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

#### Return values

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

## 10.33 cFE Pipe Management APIs

### Functions

- `CFE_Status_t CFE_SB_CreatePipe (CFE_SB_Pipeld_t *PipeldPtr, uint16 Depth, const char *PipeName)`  
*Creates a new software bus pipe.*
- `CFE_Status_t CFE_SB_DeletePipe (CFE_SB_Pipeld_t Pipeld)`  
*Delete a software bus pipe.*
- `CFE_Status_t CFE_SB_Pipeld_ToIndex (CFE_SB_Pipeld_t Pipeld, uint32 *Idx)`  
*Obtain an index value correlating to an SB Pipe ID.*
- `CFE_Status_t CFE_SB_SetPipeOpts (CFE_SB_Pipeld_t Pipeld, uint8 Opts)`  
*Set options on a pipe.*
- `CFE_Status_t CFE_SB_GetPipeOpts (CFE_SB_Pipeld_t Pipeld, uint8 *OptsPtr)`  
*Get options on a pipe.*
- `CFE_Status_t CFE_SB_GetPipeName (char *PipeNameBuf, size_t PipeNameSize, CFE_SB_Pipeld_t Pipeld)`  
*Get the pipe name for a given id.*
- `CFE_Status_t CFE_SB_GetPipeldByName (CFE_SB_Pipeld_t *PipeldPtr, const char *PipeName)`  
*Get pipe id by pipe name.*

#### 10.33.1 Detailed Description

#### 10.33.2 Function Documentation

**10.33.2.1 CFE\_SB\_CreatePipe()** `CFE_Status_t CFE_SB_CreatePipe (`  
`CFE_SB_Pipeld_t * PipeIdPtr,`  
`uint16 Depth,`  
`const char * PipeName )`

Creates a new software bus pipe.

#### Description

This routine creates and initializes an input pipe that the calling application can use to receive software bus messages. By default, no messages are routed to the new pipe. So, the application must use `CFE_SB_Subscribe` to specify which messages it wants to receive on this pipe.

#### Assumptions, External Events, and Notes:

None

#### Parameters

out	<code>PipeldPtr</code>	A pointer to a variable of type <code>CFE_SB_Pipeld_t</code> (must not be null), which will be filled in with the pipe ID information by the <code>CFE_SB_CreatePipe</code> routine. <code>*PipeldPtr</code> is the identifier for the created pipe.
in	<code>Depth</code>	The maximum number of messages that will be allowed on this pipe at one time.
in	<code>PipeName</code>	A string (must not be null) to be used to identify this pipe in error messages and routing information telemetry. The string must be no longer than <code>OS_MAX_API_NAME</code> (including terminator). Longer strings will be truncated.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_SB_MAX_PIPES_MET</a>	Max Pipes Met.
<a href="#">CFE_SB_PIPE_CR_ERR</a>	Pipe Create Error.

**See also**

[CFE\\_SB\\_DeletePipe](#) [CFE\\_SB\\_GetPipeOpts](#) [CFE\\_SB\\_SetPipeOpts](#) [CFE\\_SB\\_GetPipeIdByName](#)

Referenced by [CS\\_SbInit\(\)](#).

**10.33.2.2 CFE\_SB\_DeletePipe()** [CFE\\_Status\\_t](#) [CFE\\_SB\\_DeletePipe](#) (

[CFE\\_SB\\_PipeId\\_t](#) *PipeId* )

Delete a software bus pipe.

**Description**

This routine deletes an input pipe and cleans up all data structures associated with the pipe. All subscriptions made for this pipe by calls to [CFE\\_SB\\_Subscribe](#) will be automatically removed from the SB routing tables. Any messages in the pipe will be discarded.

Applications should not call this routine for all of their SB pipes as part of their orderly shutdown process, as the pipe will be deleted by the support framework at the appropriate time.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>PipeId</i>	The pipe ID (obtained previously from <a href="#">CFE_SB_CreatePipe</a> ) of the pipe to be deleted.
----	---------------	--

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_SB\\_CreatePipe](#) [CFE\\_SB\\_GetPipeOpts](#) [CFE\\_SB\\_SetPipeOpts](#) [CFE\\_SB\\_GetPipeIdByName](#)

**10.33.2.3 CFE\_SB\_GetPipeIdByName()** `CFE_Status_t` CFE\_SB\_GetPipeIdByName (   
`CFE_SB_PipeId_t * PipeIdPtr,`  
`const char * PipeName )`

Get pipe id by pipe name.

#### Description

This routine finds the pipe id for a pipe name.

#### Parameters

in	<i>PipeName</i>	The name of the pipe (must not be null).
out	<i>PipeIdPtr</i>	The Pipeld for that name (must not be null).

#### Returns

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

#### Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_SB_BAD_ARGUMENT</code>	Bad Argument.

#### See also

[CFE\\_SB\\_CreatePipe](#) [CFE\\_SB\\_DeletePipe](#) [CFE\\_SB\\_SetPipeOpts](#) [CFE\\_SB\\_PIPEOPTS\\_IGNOREMINE](#)

**10.33.2.4 CFE\_SB\_GetPipeName()** `CFE_Status_t` CFE\_SB\_GetPipeName (   
`char * PipeNameBuf,`  
`size_t PipeNameSize,`  
`CFE_SB_PipeId_t PipeId )`

Get the pipe name for a given id.

#### Description

This routine finds the pipe name for a pipe id.

#### Parameters

out	<i>PipeNameBuf</i>	The buffer to receive the pipe name (must not be null).
in	<i>PipeNameSize</i>	The size (in chars) of the PipeName buffer (must not be zero).
in	<i>Pipeld</i>	The Pipeld for that name.

#### Returns

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_SB\\_CreatePipe](#) [CFE\\_SB\\_DeletePipe](#) [CFE\\_SB\\_SetPipeOpts](#) [CFE\\_SB\\_GetPipeIdByName](#)

**10.33.2.5 CFE\_SB\_GetPipeOpts()** [CFE\\_Status\\_t](#) CFE\_SB\_GetPipeOpts (   
 [CFE\\_SB\\_PipeId\\_t](#) PipeId,   
 [uint8](#) \* OptsPtr )

Get options on a pipe.

**Description**

This routine gets the current options on a pipe.

**Parameters**

in	<i>PipeId</i>	The pipe ID of the pipe to get options from.
out	<i>OptsPtr</i>	A bit field of options: <a href="#">cFE SB Pipe options</a> (must not be null)

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_SB\\_CreatePipe](#) [CFE\\_SB\\_DeletePipe](#) [CFE\\_SB\\_SetPipeOpts](#) [CFE\\_SB\\_GetPipeIdByName](#) [CFE\\_SB\\_PIPEOPTS\\_IGNOREMIN](#)

**10.33.2.6 CFE\_SB\_PipeId\_ToIndex()** [CFE\\_Status\\_t](#) CFE\_SB\_PipeId\_ToIndex (   
 [CFE\\_SB\\_PipeId\\_t](#) PipeID,   
 [uint32](#) \* Idx )

Obtain an index value correlating to an SB Pipe ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] application IDs will never overlap, but the index of a pipe ID and an app ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

**Note**

There is no inverse of this function - indices cannot be converted back to the original PipeID value. The caller should retain the original ID for future use.

**Parameters**

in	<i>PipeID</i>	Pipe ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i></a>	Resource ID is not valid.

**10.33.2.7 CFE\_SB\_SetPipeOpts()** [\*CFE\\_Status\\_t\*](#) CFE\_SB\_SetPipeOpts (

```
    CFE\_SB\_PipeId\_t PipeId,
    uint8 Opts )
```

Set options on a pipe.

**Description**

This routine sets (or clears) options to alter the pipe's behavior. Options are (re)set every call to this routine.

**Parameters**

in	<i>PipeId</i>	The pipe ID of the pipe to set options on.
in	<i>Opts</i>	A bit field of options: <a href="#">cFE SB Pipe options</a>

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_SB_BAD_ARGUMENT</i></a>	Bad Argument.

**See also**

[CFE\\_SB\\_CreatePipe](#) [CFE\\_SB\\_DeletePipe](#) [CFE\\_SB\\_GetPipeOpts](#) [CFE\\_SB\\_GetPipeIdByName](#) [CFE\\_SB\\_PIPEOPTS\\_IGNOREMIN](#)

## 10.34 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.*

#### 10.34.1 Detailed Description

#### 10.34.2 Function Documentation

**10.34.2.1 CFE\_SB\_Subscribe()** `CFE_Status_t CFE_SB_Subscribe (`  
    `CFE_SB_MsgId_t MsgId,`  
    `CFE_SB_PipeId_t PipeId )`

Subscribe to a message on the software bus with default parameters.

##### Description

This routine adds the specified pipe to the destination list for the specified message ID. This is the same as `CFE_SB_SubscribeEx` with the Quality field set to `CFE_SB_DEFAULT_QOS` and `MsgLim` set to `CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT` (4).

##### Assumptions, External Events, and Notes:

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

##### Parameters

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

##### Returns

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_SB_MAX_MSGS_MET</i>	(return value only verified in coverage test) Max Messages Met.
<i>CFE_SB_MAX_DESTS_MET</i>	Max Destinations Met.
<i>CFE_SB_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_SB_BUF_ALOC_ERR</i>	(return value only verified in coverage test) Buffer Allocation Error.

**See also**

[CFE\\_SB\\_SubscribeEx](#), [CFE\\_SB\\_SubscribeLocal](#), [CFE\\_SB\\_Unsubscribe](#), [CFE\\_SB\\_UnsubscribeLocal](#)

Referenced by CS\_SbInit().

**10.34.2.2 CFE\_SB\_SubscribeEx() *CFE\_Status\_t* CFE\_SB\_SubscribeEx (**

```
    CFE_SB_MsgId_t MsgId,
    CFE_SB_PipeId_t PipeId,
    CFE_SB_Qos_t Quality,
    uint16 MsgLim )
```

Subscribe to a message on the software bus.

**Description**

This routine adds the specified pipe to the destination list associated with the specified message ID.

**Assumptions, External Events, and Notes:**

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

**Parameters**

in	<i>MsgId</i>	The message ID of the message to be subscribed to.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should be sent to.
in	<i>Quality</i>	The requested Quality of Service (QoS) required of the messages. Most callers will use <a href="#">CFE_SB_DEFAULT_QOS</a> for this parameter.
in	<i>MsgLim</i>	The maximum number of messages with this Message ID to allow in this pipe at the same time.

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_SB_MAX_MSGS_MET</i>	(return value only verified in coverage test) Max Messages Met.
<i>CFE_SB_MAX_DESTS_MET</i>	Max Destinations Met.
<i>CFE_SB_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_SB_BUF_ALOC_ERR</i>	(return value only verified in coverage test) Buffer Allocation Error.

**See also**

[CFE\\_SB\\_Subscribe](#), [CFE\\_SB\\_SubscribeLocal](#), [CFE\\_SB\\_Unsubscribe](#), [CFE\\_SB\\_UnsubscribeLocal](#)

**10.34.2.3 CFE\_SB\_SubscribeLocal()** `CFE_Status_t CFE_SB_SubscribeLocal (`

```
    CFE_SB_MsgId_t MsgId,  
    CFE_SB_PipeId_t PipeId,  
    uint16 MsgLim )
```

Subscribe to a message while keeping the request local to a cpu.

**Description**

This routine adds the specified pipe to the destination list for the specified message ID. This is similar to [CFE\\_SB\\_SubscribeEx](#) with the Quality field set to [CFE\\_SB\\_DEFAULT\\_QOS](#) and MsgLim set to [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_MSG\\_LIMIT](#), but will not report the subscription.

Software Bus Network (SBN) application is an example use case, where local subscriptions should not be reported to peers.

**Assumptions, External Events, and Notes:**

- This API is typically only used by Software Bus Network (SBN) Application

**Parameters**

in	<i>MsgId</i>	The message ID of the message to be subscribed to.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should be sent to.
in	<i>MsgLim</i>	The maximum number of messages with this Message ID to allow in this pipe at the same time.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_MAX_MSGS_MET</a>	(return value only verified in coverage test) Max Messages Met.
<a href="#">CFE_SB_MAX_DESTS_MET</a>	Max Destinations Met.
<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_SB_BUF_ALLOC_ERR</a>	(return value only verified in coverage test) Buffer Allocation Error.

**See also**

[CFE\\_SB\\_Subscribe](#), [CFE\\_SB\\_SubscribeEx](#), [CFE\\_SB\\_Unsubscribe](#), [CFE\\_SB\\_UnsubscribeLocal](#)

**10.34.2.4 CFE\_SB\_Unsubscribe()** `CFE_Status_t CFE_SB_Unsubscribe (`

```
    CFE_SB_MsgId_t MsgId,  
    CFE_SB_PipeId_t PipeId )
```

Remove a subscription to a message on the software bus.

**Description**

This routine removes the specified pipe from the destination list for the specified message ID.

**Assumptions, External Events, and Notes:**

If the Pipe is not subscribed to MsgId, the CFE\_SB\_UNSUB\_NO\_SUBS\_EID event will be generated and [CFE\\_SUCCESS](#) will be returned

**Parameters**

in	<i>MsgId</i>	The message ID of the message to be unsubscribed.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should no longer be sent to.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_SB\\_Subscribe](#), [CFE\\_SB\\_SubscribeEx](#), [CFE\\_SB\\_SubscribeLocal](#), [CFE\\_SB\\_UnsubscribeLocal](#)

**10.34.2.5 CFE\_SB\_UnsubscribeLocal()** [CFE\\_Status\\_t](#) CFE\_SB\_UnsubscribeLocal (   
*CFE\_SB\_MsgId\_t MsgId,*  
*CFE\_SB\_PipeId\_t PipeId* )

Remove a subscription to a message on the software bus on the current CPU.

**Description**

This routine removes the specified pipe from the destination list for the specified message ID on the current CPU.

**Assumptions, External Events, and Notes:**

This API is typically only used by Software Bus Network (SBN) Application. If the Pipe is not subscribed to MsgId, the CFE\_SB\_UNSUB\_NO\_SUBS\_EID event will be generated and [CFE\\_SUCCESS](#) will be returned

**Parameters**

in	<i>MsgId</i>	The message ID of the message to be unsubscribed.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should no longer be sent to.

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_SB_BAD_ARGUMENT</i></a>	Bad Argument.

**See also**

[CFE\\_SB\\_Subscribe](#), [CFE\\_SB\\_SubscribeEx](#), [CFE\\_SB\\_SubscribeLocal](#), [CFE\\_SB\\_Unsubscribe](#)

## 10.35 cFE Send/Receive Message APIs

### Functions

- `CFE_Status_t CFE_SB_TransmitMsg (const CFE_MSG_Message_t *MsgPtr, bool UpdateHeader)`  
*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.*

#### 10.35.1 Detailed Description

#### 10.35.2 Function Documentation

**10.35.2.1 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

<code>in, out</code>	<code>BufPtr</code>	A pointer to the software bus buffer to receive to (must not be null). Typically a caller declares a ptr of type <code>CFE_SB_Buffer_t</code> (i.e. <code>CFE_SB_Buffer_t *Ptr</code> ) then gives the address of that pointer ( <code>&amp;Ptr</code> ) as this parameter. After a successful receipt of a message, <code>*BufPtr</code> will point to the first byte of the software bus buffer. This should be used as a read-only pointer (in systems with an MMU, writes to this pointer may cause a memory protection fault). The <code>*BufPtr</code> is valid only until the next call to <code>CFE_SB_ReceiveBuffer</code> for the same pipe.
<code>in</code>	<code>PipeId</code>	The pipe ID of the pipe containing the message to be obtained.
<code>in</code>	<code>TimeOut</code>	The number of milliseconds to wait for a new message if the pipe is empty at the time of the call. This can also be set to <code>CFE_SB_POLL</code> for a non-blocking receive or <code>CFE_SB_PEND_FOREVER</code> to wait forever for a message to arrive.

#### Returns

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

#### Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_SB_BAD_ARGUMENT</code>	Bad Argument.

**Return values**

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

Referenced by CS\_AppMain().

```
10.35.2.2 CFE_SB_TransmitMsg() CFE_Status_t CFE_SB_TransmitMsg (
    const CFE_MSG_Message_t * MsgPtr,
    bool UpdateHeader )
```

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.

In general, the "UpdateHeader" parameter should be passed as "true" if the message was newly constructed by the sender and is being sent for the first time. When forwarding a message that originated from an external entity (e.g. messages passing through CI or SBN), the parameter should be passed as "false" to not overwrite existing data.

**Assumptions, External Events, and Notes:**

- This routine will not normally wait for the receiver tasks to process the message before returning control to the caller's task.
- However, if a higher priority task is pending and subscribed to this message, that task may get to run before returning control to the caller.
- In previous versions of CFE, the boolean parameter referred to the sequence number header of telemetry messages only. This has been extended to apply more generically to any headers, as determined by the CFE MSG implementation.

**Parameters**

in	<i>MsgPtr</i>	A pointer to the message to be sent (must not be null). This must point to the first byte of the message header.
in	<i>UpdateHeader</i>	Update the headers of the message

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_SB_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_SB_MSG_TOO_BIG</i>	Message Too Big.
<i>CFE_SB_BUF_ALOC_ERR</i>	(return value only verified in coverage test) Buffer Allocation Error.

Referenced by CS\_HousekeepingCmd().

## 10.36 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 \(CFE\\_SB\\_Buffer\\_t \\*BufPtr, bool UpdateHeader\)](#)  
*Transmit a buffer.*

#### 10.36.1 Detailed Description

#### 10.36.2 Function Documentation

##### 10.36.2.1 CFE\_SB\_AllocateMessageBuffer() [CFE\\_SB\\_Buffer\\_t\\* CFE\\_SB\\_AllocateMessageBuffer \( size\\_t MsgSize \)](#)

Get a buffer pointer to use for "zero copy" SB sends.

#### Description

This routine can be used to get a pointer to one of the software bus' internal memory buffers that are used for sending messages. The caller can use this memory buffer to build an SB message, then send it using the [CFE\\_SB\\_TransmitBuffer\(\)](#) function. This interface avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer.

#### Assumptions, External Events, and Notes:

1. The pointer returned by [CFE\\_SB\\_AllocateMessageBuffer\(\)](#) is only good for one call to [CFE\\_SB\\_TransmitBuffer\(\)](#).
2. Once a buffer has been successfully transmitted (as indicated by a successful return from [CFE\\_SB\\_TransmitBuffer\(\)](#)) the buffer becomes owned by the SB application. It will automatically be freed by SB once all recipients have finished reading it.
3. Applications must not de-reference the message pointer (for reading or writing) after the call to [CFE\\_SB\\_TransmitBuffer\(\)](#).
4. If [CFE\\_SB\\_ReleaseMessageBuffer](#) should be used only if a message is not transmitted

#### Parameters

in	<i>MsgSize</i>	The size of the SB message buffer the caller wants (including the SB message header).
----	----------------	---

#### Returns

A pointer to a memory buffer that message data can be written to for use with [CFE\\_SB\\_TransmitBuffer\(\)](#).

##### 10.36.2.2 CFE\_SB\_ReleaseMessageBuffer() [CFE\\_Status\\_t CFE\\_SB\\_ReleaseMessageBuffer \( CFE\\_SB\\_Buffer\\_t \\* BufPtr \)](#)

Release an unused "zero copy" buffer pointer.

### Description

This routine can be used to release a pointer to one of the software bus' internal memory buffers.

### Assumptions, External Events, and Notes:

1. This function is not needed for normal "zero copy" transfers. It is needed only for cleanup when an application gets a pointer using [CFE\\_SB\\_AllocateMessageBuffer\(\)](#), but (due to some error condition) never uses that pointer in a call to [CFE\\_SB\\_TransmitBuffer\(\)](#).

### Parameters

in	<i>BufPtr</i>	A pointer to the SB internal buffer (must not be null). This must be a pointer returned by a call to <a href="#">CFE_SB_AllocateMessageBuffer()</a> , but never used in a call to <a href="#">CFE_SB_TransmitBuffer()</a> .
----	---------------	---

### Returns

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

### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_BUFFER_INVALID</a>	Buffer Invalid.

**10.36.2.3 CFE\_SB\_TransmitBuffer()** [CFE\\_Status\\_t](#) CFE\_SB\_TransmitBuffer (   
     [CFE\\_SB\\_Buffer\\_t](#) \* *BufPtr*,  
     bool *UpdateHeader* )

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.

In general, the "UpdateHeader" parameter should be passed as "true" if the message was newly constructed by the sender and is being sent for the first time. When forwarding a message that originated from an external entity (e.g. messages passing through CI or SBN), the parameter should be passed as "false" to not overwrite existing data.

### Assumptions, External Events, and Notes:

1. A handle returned by [CFE\\_SB\\_AllocateMessageBuffer](#) is "consumed" by a *successful* call to [CFE\\_SB\\_TransmitBuffer](#).
2. If this function returns [CFE\\_SUCCESS](#), this indicates the zero copy handle is now owned by software bus, and is no longer owned by the calling application, and should not be re-used.
3. However if this function fails (returns any error status) it does not change the state of the buffer at all, meaning the calling application still owns it. (a failure means the buffer is left in the same state it was before the call).
4. Applications should be written as if [CFE\\_SB\\_AllocateMessageBuffer](#) is equivalent to a `malloc()` and a successful call to [CFE\\_SB\\_TransmitBuffer](#) is equivalent to a `free()`.

5. Applications must not de-reference the message pointer (for reading or writing) after a successful call to [CFE\\_SB\\_TransmitBuffer](#).
6. This function will increment and apply the internally tracked sequence counter if set to do so.

**Parameters**

in	<i>BufPtr</i>	A pointer to the buffer to be sent (must not be null).
in	<i>UpdateHeader</i>	Update the headers of the message

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_SB_MSG_TOO_BIG</a>	Message Too Big.

## 10.37 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.*

#### 10.37.1 Detailed Description

#### 10.37.2 Function Documentation

##### 10.37.2.1 CFE\_SB\_GetUserData() `void* CFE_SB_GetUserData (` `CFE_MSG_Message_t * MsgPtr )`

Get a pointer to the user data portion of a software bus message.

#### Description

This routine returns a pointer to the user data portion of a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function and avoid hard coding offsets into their SB message buffers.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>MsgPtr</code>	A pointer to the buffer that contains the software bus message (must not be null).
----	---------------------	--

#### Returns

A pointer to the first byte of user data within the software bus message.

##### 10.37.2.2 CFE\_SB\_GetUserDataLength() `size_t CFE_SB_GetUserDataLength (` `const CFE_MSG_Message_t * MsgPtr )`

Gets the length of user data in a software bus message.

**Description**

This routine returns the size of the user data in a software bus message.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
----	---------------	---

**Returns**

The size (in bytes) of the user data in the software bus message.

**Return values**

0	if an error occurs, such as if the <i>MsgPtr</i> argument is not valid.
---	---

```
10.37.2.3 CFE_SB_MessageStringGet() int32 CFE_SB_MessageStringGet (
    char * DestStringPtr,
    const char * SourceStringPtr,
    const char * DefaultString,
    size_t DestMaxSize,
    size_t SourceMaxSize )
```

Copies a string out of a software bus message.

**Description**

Strings within software bus messages have a defined/fixed maximum length, and may not necessarily be null terminated within the message. This presents a possible issue when using the C library functions to copy strings out of a message.

This function should replace use of C library functions such as strcpy/strncpy when copying strings out of software bus messages to local storage buffers.

Up to [SourceMaxSize] or [DestMaxSize-1] (whichever is smaller) characters will be copied from the source buffer to the destination buffer, and a NUL termination character will be written to the destination buffer as the last character.

If the *DefaultString* pointer is non-NULL, it will be used in place of the source string if the source is an empty string. This is typically a string constant that comes from the platform configuration, allowing default values to be assumed for fields that are unspecified.

IMPORTANT - the default string, if specified, must be null terminated. This will be the case if a string literal is passed in (the typical/expected use case).

If the default is NULL, then only the source string will be copied, and the result will be an empty string if the source was empty.

If the destination buffer is too small to store the entire string, it will be truncated, but it will still be null terminated.

**Parameters**

out	<i>DestStringPtr</i>	Pointer to destination buffer (must not be null)
-----	----------------------	--

**Parameters**

in	<i>SourceStringPtr</i>	Pointer to source buffer (component of SB message definition)
in	<i>DefaultString</i>	Default string to use if source is empty
in	<i>DestMaxSize</i>	Size of destination storage buffer (must not be zero)
in	<i>SourceMaxSize</i>	Size of source buffer as defined by the message definition

**Returns**

Number of characters copied or error code, see [cFE Return Code Defines](#)

**Return values**

<a href="#"><i>CFE_SB_BAD_ARGUMENT</i></a>	Bad Argument.
--	---------------

Referenced by CS\_ProcessNewAppDefinitionTable(), and CS\_ProcessNewTablesDefinitionTable().

**10.37.2.4 CFE\_SB\_MessageStringSet()** `int32 CFE_SB_MessageStringSet (`  
 `char * DestStringPtr,`  
 `const char * SourceStringPtr,`  
 `size_t DestMaxSize,`  
 `size_t SourceMaxSize )`

Copies a string into a software bus message.

**Description**

Strings within software bus messages have a defined/fixed maximum length, and may not necessarily be null terminated within the message. This presents a possible issue when using the C library functions to copy strings out of a message.

This performs a very similar function to "strncpy()" except that the sizes of *both* buffers are passed in. Neither buffer is required to be null-terminated, but copying will stop after the first termination character is encountered.

If the destination buffer is not completely filled by the source data (such as if the supplied string was shorter than the allotted length) the destination buffer will be padded with NUL characters up to the size of the buffer, similar to what strncpy() does. This ensures that the entire destination buffer is set.

**Note**

If the source string buffer is already guaranteed to be null terminated, then there is no difference between the C library "strncpy()" function and this implementation. It is only necessary to use this when termination of the source buffer is not guaranteed.

**Parameters**

out	<i>DestStringPtr</i>	Pointer to destination buffer (component of SB message definition) (must not be null)
in	<i>SourceStringPtr</i>	Pointer to source buffer (must not be null)
in	<i>DestMaxSize</i>	Size of destination buffer as defined by the message definition
in	<i>SourceMaxSize</i>	Size of source buffer

**Returns**

Number of characters copied or error code, see [cFE Return Code Defines](#)

**Return values**

<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.
-------------------------------------	---------------

**10.37.2.5 CFE\_SB\_SetUserDataLength()** `void CFE_SB_SetUserDataLength (`

```
    CFE_MSG_Message_t * MsgPtr,  
    size_t DataLength )
```

Sets the length of user data in a software bus message.

**Description**

This routine sets the field in the SB message header that determines the size of the user data in a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function rather than trying to poke a length value directly into their SB message buffers.

**Assumptions, External Events, and Notes:**

- You must set a valid message ID in the SB message header before calling this function.

**Parameters**

in	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
in	<i>DataLength</i>	The length to set (size of the user data, in bytes).

**10.37.2.6 CFE\_SB\_TimeStampMsg()** `void CFE_SB_TimeStampMsg (`

```
    CFE_MSG_Message_t * MsgPtr )
```

Sets the time field in a software bus message with the current spacecraft time.

**Description**

This routine sets the time of a software bus message with the current spacecraft time. This will be the same time that is returned by the function [CFE\\_TIME\\_GetTime](#).

**Assumptions, External Events, and Notes:**

- If the underlying implementation of software bus messages does not include a time field, then this routine will do nothing.

**Parameters**

in	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
----	---------------	---

Referenced by CS\_HousekeepingCmd().

## 10.38 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`.*

#### 10.38.1 Detailed Description

#### 10.38.2 Function Documentation

**10.38.2.1 `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

<code>true</code>	Message ID is within the valid range
<code>false</code>	Message ID is not within the valid range

**10.38.2.2 `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

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

Definition at line 778 of file cfe\_sb.h.

References CFE\_SB\_MSGID\_UNWRAP\_VALUE.

**10.38.2.3 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 809 of file cfe\_sb.h.

References CFE\_SB\_MSGID\_UNWRAP\_VALUE.

Referenced by CS\_AppPipe(), CS\_BackgroundCheckCycle(), CS\_HousekeepingCmd(), CS\_ProcessCmd(), and CS\_VerifyCmdLength().

**10.38.2.4 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 838 of file cfe\_sb.h.

References CFE\_SB\_MSGID\_C.

Referenced by CS\_SblInit().

## 10.39 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.*

### 10.39.1 Detailed Description

### 10.39.2 Macro Definition Documentation

#### 10.39.2.1 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.

## 10.40 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.*

### 10.40.1 Detailed Description

### 10.40.2 Function Documentation

**10.40.2.1 CFE\_TBL\_Register()** `CFE_Status_t CFE_TBL_Register (`  
`CFE_TBL_Handle_t * TblHandlePtr,`  
`const char * Name,`  
`size_t Size,`  
`uint16 TblOptionFlags,`  
`CFE_TBL_CallbackFuncPtr_t TblValidationFuncPtr )`

Register a table with cFE to obtain Table Management Services.

#### Description

When an application is created and initialized, it is responsible for creating its table images via the TBL API. The application must inform the Table Service of the table name, table size and selection of optional table features.

#### Assumptions, External Events, and Notes:

Note: This function call can block. Therefore, interrupt service routines should NOT create their own tables. An application should create any table(s) and provide the handle(s) to the interrupt service routine.

#### Parameters

out	<code>TblHandlePtr</code>	a pointer to a <code>CFE_TBL_Handle_t</code> type variable (must not be null) that will be assigned the table's handle. The table handle is required for other API calls when accessing the data contained in the table. <code>*TblHandlePtr</code> is the handle used to identify table to cFE when performing Table operations. This value is returned at address specified by <code>TblHandlePtr</code> .
in	<code>Name</code>	The raw table name. This name will be combined with the name of the application to produce a name of the form "AppName.RawTableName". This application specific name will be used in commands for modifying or viewing the contents of the table.
in	<code>Size</code>	The size, in bytes, of the table to be created (must not be zero). This is the size that will be allocated as a shared memory resource between the Table Management Service and the calling application.

## Parameters

in	<i>TblOptionFlags</i>	<p>Flag bits indicating selected options for table. A bitwise OR of the following option flags:</p> <ul style="list-style-type: none"> <li>• <a href="#">CFE_TBL_OPT_DEFAULT</a> - The default setting for table options is a combination of <a href="#">CFE_TBL_OPT_SNGL_BUFFER</a> and <a href="#">CFE_TBL_OPT_LOAD_DUMP</a>. See below for a description of these two options. This option is mutually exclusive with the <a href="#">CFE_TBL_OPT_DBL_BUFFER</a>, <a href="#">CFE_TBL_OPT_DUMP_ONLY</a> and <a href="#">CFE_TBL_OPT_USR_DEF_ADDR</a> options.</li> <li>• <a href="#">CFE_TBL_OPT_SNGL_BUFFER</a> - 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 <a href="#">CFE_TBL_OPT_DBL_BUFFER</a> option</li> <li>• <a href="#">CFE_TBL_OPT_DBL_BUFFER</a> - 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 <a href="#">CFE_TBL_OPT_SNGL_BUFFER</a> and <a href="#">CFE_TBL_OPT_DEFAULT</a> option.</li> <li>• <a href="#">CFE_TBL_OPT_LOAD_DUMP</a> - 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 <a href="#">CFE_TBL_OPT_DUMP_ONLY</a> option.</li> <li>• <a href="#">CFE_TBL_OPT_DUMP_ONLY</a> - 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 <a href="#">CFE_TBL_GetAddress</a> API function. This option is mutually exclusive with the <a href="#">CFE_TBL_OPT_LOAD_DUMP</a> and <a href="#">CFE_TBL_OPT_DEFAULT</a> options. If the Application wishes to specify their own block of memory as the Dump Only table, they need to also include the <a href="#">CFE_TBL_OPT_USR_DEF_ADDR</a> option explained below.</li> <li>• <a href="#">CFE_TBL_OPT_NOT_USR_DEF</a> - 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 <a href="#">CFE_TBL_OPT_USR_DEF_ADDR</a> option.</li> <li>• <a href="#">CFE_TBL_OPT_USR_DEF_ADDR</a> - 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 <a href="#">CFE_TBL_Load</a> function. This option implies the <a href="#">CFE_TBL_OPT_DUMP_ONLY</a> and the <a href="#">CFE_TBL_OPT_SNGL_BUFFER</a> options and is mutually exclusive of the <a href="#">CFE_TBL_OPT_DBL_BUFFER</a> option.</li> <li>• <a href="#">CFE_TBL_OPT_CRITICAL</a> - 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 <a href="#">CFE_TBL_OPT_USR_DEF_ADDR</a> and <a href="#">CFE_TBL_OPT_DUMP_ONLY</a> 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</li> </ul>

### Parameters

in	<i>TblValidationFuncPtr</i>	<p>is a pointer to a function that will be executed in the context of the Table Management Service when the contents of a table need to be validated. If set to NULL, then the Table Management Service will assume any data is valid. If the value is not NULL, it must be a pointer to a function with the following prototype:</p> <pre><b>int32 CallbackFunc(void *TblPtr);</b></pre> <p>where</p> <p><b>TblPtr</b> will be a pointer to the table data that is to be verified. When the function returns <a href="#">CFE_SUCCESS</a>, 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 <b>must</b> return either <a href="#">CFE_SUCCESS</a> or a negative number (whose value is at the developer's discretion). The validation function will be executed in the Application's context so that Event Messages describing the validation failure are possible from within the function.</p>
----	-----------------------------	---

### Returns

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

### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_INFO_RECOVERED_TBL</a>	Recovered Table.
<a href="#">CFE_TBL_ERR_DUPLICATE_DIFF_SIZE</a>	Duplicate Table With Different Size.
<a href="#">CFE_TBL_ERR_DUPLICATE_NOT OWNED</a>	Duplicate Table And Not Owned.
<a href="#">CFE_TBL_ERR_REGISTRY_FULL</a>	Registry Full.
<a href="#">CFE_TBL_ERR_HANDLES_FULL</a>	Handles Full.
<a href="#">CFE_TBL_ERR_INVALID_SIZE</a>	Invalid Size.
<a href="#">CFE_TBL_ERR_INVALID_NAME</a>	Invalid Name.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_TBL_ERR_INVALID_OPTIONS</a>	Invalid Options.
<a href="#">CFE_TBL_WARN_DUPLICATE</a>	Duplicate Warning.
<a href="#">CFE_TBL_WARN_NOT_CRITICAL</a>	Not Critical Warning.

### See also

[CFE\\_TBL\\_Unregister](#), [CFE\\_TBL\\_Share](#)

Referenced by [CS\\_TableInit\(\)](#).

```
10.40.2.2 CFE_TBL_Share() CFE\_Status\_t CFE_TBL_Share (
    CFE\_TBL\_Handle\_t * TblHandlePtr,
    const char * TblName )
```

Obtain handle of table registered by another application.

### Description

After a table has been created, other applications can gain access to that table via the table handle. In order for two or more applications to share a table, the applications that do not create the table must obtain the handle using this function.

### Assumptions, External Events, and Notes:

None

### Parameters

<i>out</i>	<i>TblHandlePtr</i>	A pointer to a <a href="#">CFE_TBL_Handle_t</a> type variable (must not be null) that will be assigned the table's handle. The table handle is required for other API calls when accessing the data contained in the table. *TblHandlePtr is the handle used to identify table to cFE when performing Table operations. This value is returned at the address specified by TblHandlePtr.
<i>in</i>	<i>TblName</i>	The application specific name of the table of the form "AppName.RawTableName", where RawTableName is the name specified in the <a href="#">CFE_TBL_Register</a> 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

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_ERR_HANDLES_FULL</a>	Handles Full.
<a href="#">CFE_TBL_ERR_INVALID_NAME</a>	Invalid Name.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_BAD_ARGUMENT</a>	Bad Argument.

### See also

[CFE\\_TBL\\_Unregister](#), [CFE\\_TBL\\_Register](#)

Referenced by `CS_AttemptTableReshare()`, and `CS_ComputeTables()`.

### 10.40.2.3 CFE\_TBL\_Unregister()

```
CFE_Status_t CFE_TBL_Unregister (
    CFE_TBL_Handle_t TblHandle )
```

Unregister a table.

### Description

When an application is being removed from the system, ES will clean up/free all the application related resources including tables so apps are not required to call this function.

A valid use-case for this API is to unregister a shared table if access is no longer needed or the owning application was removed from the system (CS app is an example).

Typically apps should only register tables during initialization and registration/unregistration by the owning application during operation should be avoided. If unavoidable, special care needs to be taken (especially for shared tables) to avoid race conditions due to competing requests from multiple tasks.

Note the table will not be removed from memory until all table access links have been removed (registration and all shared access).

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , that identifies the Table to be unregistered.
----	------------------	--

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.

**See also**

[CFE\\_TBL\\_Share](#), [CFE\\_TBL\\_Register](#)

Referenced by `CS_ComputeTables()`, and `CS_HandleTableUpdate()`.

## 10.41 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.*

#### 10.41.1 Detailed Description

#### 10.41.2 Function Documentation

**10.41.2.1 `CFE_TBL_DumpToBuffer()`** `CFE_Status_t CFE_TBL_DumpToBuffer (`  
`CFE_TBL_Handle_t TblHandle )`

Copies the contents of a Dump Only Table to a shared buffer.

##### Description

Typically, apps should just call `CFE_TBL_Manage` as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just a dump should be performed.

##### Assumptions, External Events, and Notes:

If the table does not have a dump pending status, nothing will occur (no error, no dump)

##### Parameters

in	<code>TblHandle</code>	Handle of Table to be dumped.
----	------------------------	-------------------------------

##### Returns

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

##### Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code>	Resource ID is not valid.
<code>CFE_TBL_ERR_NO_ACCESS</code>	No Access.
<code>CFE_TBL_ERR_INVALID_HANDLE</code>	Invalid Handle.
<code>CFE_TBL_INFO_DUMP_PENDING</code>	Dump Pending.

**See also**

[CFE\\_TBL\\_Manage](#)

**10.41.2.2 CFE\_TBL\_Load()** `CFE_Status_t CFE_TBL_Load (`  
 `CFE_TBL_Handle_t TblHandle,`  
 `CFE_TBL_SrcEnum_t SrcType,`  
 `const void * SrcDataPtr )`

Load a specified table with data from specified source.

**Description**

Once an application has created a table ([CFE\\_TBL\\_Register](#)), it must provide the values that initialize the contents of that table. The application accomplishes this with one of two different TBL API calls. This function call initializes the table with values that are held in a data structure.

**Assumptions, External Events, and Notes:**

This function call can block. Therefore, interrupt service routines should NOT initialize their own tables. An application should initialize any table(s) prior to providing the handle(s) to the interrupt service routine.

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , that identifies the Table to be loaded.
in	<i>SrcType</i>	Flag indicating the nature of the given <i>SrcDataPtr</i> below. This value can be any one of the following: <ul style="list-style-type: none"> <li>• <a href="#">CFE_TBL_SRC_FILE</a> - File source When this option is selected, the <i>SrcDataPtr</i> will be interpreted as a pointer to a null terminated character string. The string should specify the full path and filename of the file containing the initial data contents of the table.</li> <li>• <a href="#">CFE_TBL_SRC_ADDRESS</a> - Address source When this option is selected, the <i>SrcDataPtr</i> will be interpreted as a pointer to a memory location that is the beginning of the initialization data for loading the table OR, in the case of a "user defined" dump only table, the address of the active table itself. The block of memory is assumed to be of the same size specified in the <a href="#">CFE_TBL_Register</a> function Size parameter.</li> </ul>
in	<i>SrcDataPtr</i>	Pointer (must not be null) to either a character string specifying a filename or a memory address of a block of binary data to be loaded into a table or, if the table was registered with the <a href="#">CFE_TBL_OPT_USR_DEF_ADDR</a> option, the address of the active table buffer.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.

**Return values**

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

**See also**

[CFE\\_TBL\\_Update](#), [CFE\\_TBL\\_Validate](#), [CFE\\_TBL\\_Manage](#)

Referenced by `CS_TableInit()`.

**10.41.2.3 CFE\_TBL\_Manage()** `CFE_Status_t CFE_TBL_Manage (`  
`CFE_TBL_Handle_t TblHandle )`

Perform standard operations to maintain a table.

**Description**

Applications should call this API periodically to process pending requests for update, validation, or dump to buffer. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , that identifies the Table to be managed.
----	------------------	---

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATED</i>	Updated.

**Return values**

<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.
<a href="#">CFE_TBL_INFO_DUMP_PENDING</a>	Dump Pending.
<a href="#">CFE_TBL_INFO_UPDATE_PENDING</a>	Update Pending.
<a href="#">CFE_TBL_INFO_VALIDATION_PENDING</a>	

**See also**

[CFE\\_TBL\\_Update](#), [CFE\\_TBL\\_Validate](#), [CFE\\_TBL\\_Load](#), [CFE\\_TBL\\_DumpToBuffer](#)

Referenced by `CS_HandleTableUpdate()`.

**10.41.2.4 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	<code>TblHandle</code>	Handle of Table that was modified.
----	------------------------	------------------------------------

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.

**See also**

[CFE\\_TBL\\_Manage](#)

Referenced by `CS_DisableEntryIDEEPROMCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_DisableNameAppCmd()`, `CS_DisableNameTablesCmd()`, `CS_EnableEntryIDEEPROMCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_Enable`

NameAppCmd(), CS\_EnableNameTablesCmd(), CS\_RecomputeAppChildTask(), CS\_RecomputeEepromMemoryChildTask(), and CS\_RecomputeTablesChildTask().

#### 10.41.2.5 CFE\_TBL\_Update() `CFE_Status_t CFE_TBL_Update ( CFE_TBL_Handle_t TblHandle )`

Update contents of a specified table, if an update is pending.

##### Description

Typically, apps should just call [CFE\\_TBL\\_Manage](#) as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just an update should be performed.

##### Assumptions, External Events, and Notes:

None

##### Parameters

in	<i>TblHandle</i>	Handle, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , that identifies the Table to be updated.
----	------------------	---

##### Returns

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

##### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_INFO_NO_UPDATE_PENDING</a>	No Update Pending.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.

##### See also

[CFE\\_TBL\\_Load](#), [CFE\\_TBL\\_Validate](#), [CFE\\_TBL\\_Manage](#)

#### 10.41.2.6 CFE\_TBL\_Validate() `CFE_Status_t CFE_TBL_Validate ( CFE_TBL_Handle_t TblHandle )`

Perform steps to validate the contents of a table image.

##### Description

Typically, apps should just call [CFE\\_TBL\\_Manage](#) as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just a validation should be performed.

##### Assumptions, External Events, and Notes:

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , that identifies the Table to be managed.
----	------------------	---

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_INFO_NO_VALIDATION_PENDING</a>	
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.

**See also**

[CFE\\_TBL\\_Update](#), [CFE\\_TBL\\_Manage](#), [CFE\\_TBL\\_Load](#)

## 10.42 cFE Access Table Content APIs

### Functions

- **`CFE_Status_t CFE_TBL_GetAddress`** (`void **TblPtr, CFE_TBL_Handle_t TblHandle`)
 

*Obtain the current address of the contents of the specified table.*
- **`CFE_Status_t CFE_TBL_ReleaseAddress`** (`CFE_TBL_Handle_t TblHandle`)
 

*Release previously obtained pointer to the contents of the specified table.*
- **`CFE_Status_t CFE_TBL_GetAddresses`** (`void **TblPtrs[], uint16 NumTables, const CFE_TBL_Handle_t TblHandles[]`)
 

*Obtain the current addresses of an array of specified tables.*
- **`CFE_Status_t CFE_TBL_ReleaseAddresses`** (`uint16 NumTables, const CFE_TBL_Handle_t TblHandles[]`)
 

*Release the addresses of an array of specified tables.*

### 10.42.1 Detailed Description

### 10.42.2 Function Documentation

**10.42.2.1 `CFE_TBL_GetAddress()`** `CFE_Status_t CFE_TBL_GetAddress (`  
`void ** TblPtr,`  
`CFE_TBL_Handle_t TblHandle )`

Obtain the current address of the contents of the specified table.

#### Description

When a table has been created and initialized, it is available to any application that can identify it with its unique handle. In order to view the data contained in the table, an application must call this function or `CFE_TBL_GetAddresses`.

#### Assumptions, External Events, and Notes:

1. This call can be a blocking call when the table is not double buffered and is shared with another application of lower priority that just happens to be in the middle of a table update of the specific table. If this occurs, the application performing the table update will automatically have its priority elevated in order to release the resource as soon as possible.
2. An application must always release the returned table address using the `CFE_TBL_ReleaseAddress` or `CFE_TBL_ReleaseAddresses` function prior to either a `CFE_TBL_Update` call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.
3. `CFE_TBL_ERR_NEVER_LOADED` will be returned if the table has never been loaded (either from file or from a block of memory), but the function will still return a valid table pointer to a table with all zero content. This pointer must be released with the `CFE_TBL_ReleaseAddress` API before the table can be loaded with data.

#### Parameters

<code>out</code>	<code>TblPtr</code>	The address of a pointer (must not be null) that will be loaded with the address of the first byte of the table. This pointer can then be typecast by the calling application to the appropriate table data structure. <code>*TblPtr</code> is the address of the first byte of data associated with the specified table.
<code>in</code>	<code>TblHandle</code>	Handle, previously obtained from <code>CFE_TBL_Register</code> or <code>CFE_TBL_Share</code> , that identifies the Table whose address is to be returned.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_INFO_UPDATED</a>	Updated.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.
<a href="#">CFE_TBL_ERR_UNREGISTERED</a>	Unregistered.
<a href="#">CFE_TBL_ERR_NEVER_LOADED</a>	Never Loaded.
<a href="#">CFE_TBL_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_TBL\\_ReleaseAddress](#), [CFE\\_TBL\\_GetAddresses](#), [CFE\\_TBL\\_ReleaseAddresses](#)

Referenced by `CS_AttemptTableReshare()`, `CS_ComputeTables()`, `CS_HandleTableUpdate()`, and `CS_TableInit()`.

**10.42.2.2 CFE\_TBL\_GetAddresses()** [CFE\\_Status\\_t](#) `CFE_TBL_GetAddresses` (

```
    void ** TblPtrs[],
    uint16 NumTables,
    const CFE\_TBL\_Handle\_t TblHandles[] )
```

Obtain the current addresses of an array of specified tables.

**Description**

When a table has been created and initialized, it is available to any application that can identify it with its unique handle. In order to view the data contained in the table, an application must call this function or [CFE\\_TBL\\_GetAddress](#).

**Assumptions, External Events, and Notes:**

1. This call can be a blocking call when the table is not double buffered and is shared with another application of lower priority that just happens to be in the middle of a table update of the specific table. If this occurs, the application performing the table update will automatically have its priority elevated in order to release the resource as soon as possible.
2. An application must always release the returned table address using the [CFE\\_TBL\\_ReleaseAddress](#) or [CFE\\_TBL\\_ReleaseAddresses](#) function prior to either a [CFE\\_TBL\\_Update](#) call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.
3. [CFE\\_TBL\\_ERR\\_NEVER\\_LOADED](#) will be returned if the table has never been loaded (either from file or from a block of memory), but the function will still return a valid table pointer to a table with all zero content. This pointer must be released with the [CFE\\_TBL\\_ReleaseAddress](#) API before the table can be loaded with data.

**Parameters**

<i>out</i>	<i>TblPtrs</i>	Array of Pointers (must not be null) to variables that calling Application wishes to hold the start addresses of the Tables. *TblPtrs is an array of addresses of the first byte of data associated with the specified tables.
<i>in</i>	<i>NumTables</i>	Size of TblPtrs and TblHandles arrays.
<i>in</i>	<i>TblHandles</i>	Array of Table Handles, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , of those tables whose start addresses are to be obtained.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_INFO_UPDATED</a>	Updated.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.
<a href="#">CFE_TBL_ERR_UNREGISTERED</a>	Unregistered.
<a href="#">CFE_TBL_ERR_NEVER_LOADED</a>	Never Loaded.
<a href="#">CFE_TBL_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_TBL\\_GetAddress](#), [CFE\\_TBL\\_ReleaseAddress](#), [CFE\\_TBL\\_ReleaseAddresses](#)

#### 10.42.2.3 CFE\_TBL\_ReleaseAddress()

```
CFE_Status_t CFE_TBL_ReleaseAddress (
    CFE_TBL_Handle_t TblHandle )
```

Release previously obtained pointer to the contents of the specified table.

**Description**

Each application is **required** to release a table address obtained through the [CFE\\_TBL\\_GetAddress](#) function.

**Assumptions, External Events, and Notes:**

An application must always release the returned table address using the [CFE\\_TBL\\_ReleaseAddress](#) function prior to either a [CFE\\_TBL\\_Update](#) call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

**Parameters**

<i>in</i>	<i>TblHandle</i>	Handle, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , that identifies the Table whose address is to be released.
-----------	------------------	---

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATED</i>	Updated.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.
<i>CFE_TBL_ERR_NEVER_LOADED</i>	Never Loaded.

**See also**

[CFE\\_TBL\\_GetAddress](#), [CFE\\_TBL\\_GetAddresses](#), [CFE\\_TBL\\_ReleaseAddresses](#)

Referenced by `CS_AttemptTableReshare()`, `CS_ComputeTables()`, and `CS_HandleTableUpdate()`.

```
10.42.2.4 CFE_TBL_ReleaseAddresses() CFE_Status_t CFE_TBL_ReleaseAddresses (
    uint16 NumTables,
    const CFE_TBL_Handle_t TblHandles[] )
```

Release the addresses of an array of specified tables.

**Description**

Each application is **required** to release a table address obtained through the [CFE\\_TBL\\_GetAddress](#) function.

**Assumptions, External Events, and Notes:**

An application must always release the returned table address using the [CFE\\_TBL\\_ReleaseAddress](#) function prior to either a [CFE\\_TBL\\_Update](#) call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

**Parameters**

<i>in</i>	<i>NumTables</i>	Size of TblHandles array.
<i>in</i>	<i>TblHandles</i>	Array of Table Handles (must not be null), previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , of those tables whose start addresses are to be released.

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATED</i>	Updated.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.

**Return values**

<i>CFE_TBL_ERR_NEVER_LOADED</i>	Never Loaded.
<i>CFE_TBL_BAD_ARGUMENT</i>	Bad Argument.

**See also**

[CFE\\_TBL\\_GetAddress](#), [CFE\\_TBL\\_ReleaseAddress](#), [CFE\\_TBL\\_GetAddresses](#)

## 10.43 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.*

### 10.43.1 Detailed Description

### 10.43.2 Function Documentation

**10.43.2.1 CFE\_TBL\_GetInfo()** [CFE\\_Status\\_t CFE\\_TBL\\_GetInfo \(](#)  
[CFE\\_TBL\\_Info\\_t \\* TblInfoPtr,](#)  
[const char \\* TblName \)](#)

Obtain characteristics/information of/about a specified table.

#### Description

This API provides the registry information associated with the specified table. The function fills the given data structure with the data found in the Table Registry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

out	<i>TblInfoPtr</i>	A pointer to a CFE_TBL_Info_t data structure (must not be null) that is to be populated with table characteristics and information. *TblInfoPtr is the description of the tables characteristics and registry information stored in the <a href="#">CFE_TBL_Info_t</a> data structure format.
in	<i>TblName</i>	The application specific name (must not be null) of the table of the form "AppName.RawTableName", where RawTableName is the name specified in the <a href="#">CFE_TBL_Register</a> 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

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_ERR_INVALID_NAME</a>	Invalid Name.
<a href="#">CFE_TBL_BAD_ARGUMENT</a>	Bad Argument.

**See also**[CFE\\_TBL\\_GetStatus](#)

Referenced by CS\_AttemptTableReshare(), and CS\_ComputeTables().

**10.43.2.2 CFE\_TBL\_GetStatus()** `CFE_Status_t CFE_TBL_GetStatus ( CFE_TBL_Handle_t TblHandle )`

Obtain current status of pending actions for a table.

**Description**

An application is **required** to perform a periodic check for an update or a validation request for all the tables that it creates. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle. If a table update or validation request is pending, the Application should follow up with a call to [CFE\\_TBL\\_Update](#) or [CFE\\_TBL\\_Validate](#) respectively.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <a href="#">CFE_TBL_Register</a> or <a href="#">CFE_TBL_Share</a> , that identifies the Table to be managed.
----	------------------	---

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TBL_INFO_UPDATE_PENDING</a>	Update Pending.
<a href="#">CFE_TBL_INFO_VALIDATION_PENDING</a>	
<a href="#">CFE_TBL_INFO_DUMP_PENDING</a>	Dump Pending.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	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](#)

**10.43.2.3 CFE\_TBL\_NotifyByMessage()** `CFE_Status_t CFE_TBL_NotifyByMessage ( CFE_TBL_Handle_t TblHandle,`

```

CFE_SB_MsgId_t MsgId,
CFE_MSG_FcnCode_t CommandCode,
uint32 Parameter )

```

Instruct cFE Table Services to notify Application via message when table requires management.

#### Description

This API instructs Table Services to send a message to the calling Application whenever the specified table requires management by the application. This feature allows applications to avoid polling table services via the [CFE\\_TBL\\_Manage](#) call to determine whether a table requires updates, validation, etc. This API should be called following the [CFE\\_TBL\\_Register](#) API whenever the owning application requires this feature.

#### Assumptions, External Events, and Notes:

- Only the application that owns the table is allowed to register a notification message
- Recommend **NOT** using the ground command MID which typically impacts command counters. The typical approach is to use a unique MID for inter-task communications similar to how schedulers typically trigger application housekeeping messages.

#### Parameters

in	<i>TblHandle</i>	Handle of Table with which the message should be associated.
in	<i>MsgId</i>	Message ID to be used in notification message sent by Table Services.
in	<i>CommandCode</i>	Command Code value to be placed in secondary header of message sent by Table Services.
in	<i>Parameter</i>	Application defined value to be passed as a parameter in the message sent by Table Services. Suggested use includes an application's table index that allows the same <i>MsgId</i> and Command Code to be used for all table management notifications.

#### Returns

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

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_TBL_ERR_NO_ACCESS</a>	No Access.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.

#### See also

[CFE\\_TBL\\_Register](#)

## 10.44 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.*

### 10.44.1 Detailed Description

### 10.44.2 Macro Definition Documentation

#### 10.44.2.1 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.

#### 10.44.2.2 CFE\_TBL\_OPT\_CRITICAL `#define CFE_TBL_OPT_CRITICAL (0x0008)`

Critical table.

Definition at line 63 of file cfe\_tbl\_api\_typedefs.h.

#### 10.44.2.3 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.

**10.44.2.4 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.

**10.44.2.5 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.

**10.44.2.6 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.

**10.44.2.7 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.

**10.44.2.8 CFE\_TBL\_OPT\_LOAD\_DUMP** #define CFE\_TBL\_OPT\_LOAD\_DUMP (0x0000)

Load/Dump table.

Definition at line 53 of file cfe\_tbl\_api\_typedefs.h.

**10.44.2.9 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.

**10.44.2.10 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.

**10.44.2.11 CFE\_TBL\_OPT\_SNGL\_BUFFER** #define CFE\_TBL\_OPT\_SNGL\_BUFFER (0x0000)

Single buffer table.

Definition at line 49 of file cfe\_tbl\_api\_typedefs.h.

**10.44.2.12 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.

**10.44.2.13 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.

## 10.45 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.*

#### 10.45.1 Detailed Description

#### 10.45.2 Function Documentation

##### 10.45.2.1 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](#)

##### 10.45.2.2 CFE\_TIME\_GetMETseconds() [uint32 CFE\\_TIME\\_GetMETseconds \(void\)](#)

Get the current seconds count of the mission-elapsed time.

#### Description

This routine is the same as [CFE\\_TIME\\_GetMET](#), except that it returns only the integer seconds portion of the MET time.

**Assumptions, External Events, and Notes:**

None

**Returns**

The current MET seconds

**See also**

[CFE\\_TIME\\_GetTime](#), [CFE\\_TIME\\_GetTAI](#), [CFE\\_TIME\\_GetUTC](#), [CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETsubsecs](#),  
[CFE\\_TIME\\_MET2SCTime](#)

**10.45.2.3 CFE\_TIME\_GetMETsubsecs()** `uint32 CFE_TIME_GetMETsubsecs (`  
  `void )`

Get the current sub-seconds count of the mission-elapsed time.

**Description**

This routine is the same as [CFE\\_TIME\\_GetMET](#), except that it returns only the integer sub-seconds portion of the MET time. Each count is equal to  $2^{-32}$  seconds.

**Assumptions, External Events, and Notes:**

None

**Returns**

The current MET sub-seconds

**See also**

[CFE\\_TIME\\_GetTime](#), [CFE\\_TIME\\_GetTAI](#), [CFE\\_TIME\\_GetUTC](#), [CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETseconds](#),  
[CFE\\_TIME\\_MET2SCTime](#)

**10.45.2.4 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](#)

**10.45.2.5 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](#)

**10.45.2.6 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](#)

## 10.46 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.*

### 10.46.1 Detailed Description

### 10.46.2 Function Documentation

#### 10.46.2.1 CFE\_TIME\_GetClockInfo() `uint16 CFE_TIME_GetClockInfo (`     `void )`

Provides information about the spacecraft clock.

##### Description

This routine returns information on the spacecraft clock in a bit mask.

##### Assumptions, External Events, and Notes:

None

##### Returns

Spacecraft clock information, [cFE Clock State Flag Defines](#). To extract the information from the returned value, the flags can be used as in the following:

```
if ((ReturnValue & CFE_TIME_FLAG_xxxxxxx) == CFE_TIME_FLAG_xxxxxxx) then the following definition of the CFE_TIME_FLAG_xxxxxxx is true.
```

##### See also

[CFE\\_TIME\\_GetSTCF](#), [CFE\\_TIME\\_GetLeapSeconds](#), [CFE\\_TIME\\_GetClockState](#)

#### 10.46.2.2 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](#)

**10.46.2.3 CFE\_TIME\_GetLeapSeconds()** `int16 CFE_TIME_GetLeapSeconds ( void )`

Get the current value of the leap seconds counter.

**Description**

This routine returns the current value of the leap seconds counter. This is the delta seconds between international atomic time (TAI) and universal coordinated time (UTC). There is no API provided to set or adjust leap seconds or STCF, those actions should be done by command only. This API is provided for applications to be able to include leap seconds in their data products to aid in time correlation during downstream science data processing. Note that some mission operations teams do not maintain the leap seconds count, preferring to adjust the STCF instead. Users of this function should check with their mission ops team to see how they are planning to handle leap seconds.

**Assumptions, External Events, and Notes:**

None

**Returns**

The current spacecraft leap seconds.

**See also**

[CFE\\_TIME\\_GetSTCF](#), [CFE\\_TIME\\_GetClockState](#), [CFE\\_TIME\\_GetClockInfo](#)

**10.46.2.4 CFE\_TIME\_GetSTCF()** `CFE_TIME_SysTime_t CFE_TIME_GetSTCF ( void )`

Get the current value of the spacecraft time correction factor (STCF).

**Description**

This routine returns the current value of the spacecraft time correction factor. This is the delta time between the MET and the TAI time. There is no API provided to set or adjust leap seconds or STCF, those actions should be done by command only. This API is provided for applications to be able to include STCF in their data products to aid in time correlation during downstream science data processing.

**Assumptions, External Events, and Notes:**

Does not include leap seconds

**Returns**

The current SCTF

**See also**

[CFE\\_TIME\\_GetLeapSeconds](#), [CFE\\_TIME\\_GetClockState](#), [CFE\\_TIME\\_GetClockInfo](#)

## 10.47 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.*

#### 10.47.1 Detailed Description

#### 10.47.2 Function Documentation

**10.47.2.1 CFE\_TIME\_Add()** `CFE_TIME_SysTime_t CFE_TIME_Add (`  
    `CFE_TIME_SysTime_t Time1,`  
    `CFE_TIME_SysTime_t Time2 )`

Adds two time values.

#### Description

This routine adds the two specified times and returns the result. Normally, at least one of the input times should be a value representing a delta time. Adding two absolute times together will not cause an error, but the result will probably be meaningless.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>Time1</i>	The first time to be added.
in	<i>Time2</i>	The second time to be added.

#### Returns

The sum of the two times. If the sum is greater than the maximum value that can be stored in a `CFE_TIME_SysTime_t`, the result will roll over (this is not considered an error).

#### See also

[CFE\\_TIME\\_Subtract](#), [CFE\\_TIME\\_Compare](#)

**10.47.2.2 CFE\_TIME\_Compare()** `CFE_TIME_Compare_t CFE_TIME_Compare (`  
    `CFE_TIME_SysTime_t TimeA,`  
    `CFE_TIME_SysTime_t TimeB )`

Compares two time values.

### Description

This routine compares two time values to see which is "greater". It is important that applications use this function rather than trying to directly compare the component pieces of times. This function will handle roll-over cases seamlessly, which may not be intuitively obvious. The cFE's internal representation of time "rolls over" when the 32 bit seconds count reaches 0xFFFFFFFF. Also, subtracting a delta time from an absolute time close to the epoch could result in "roll under". The strange cases that result from these situations can be handled by defining the comparison function for times as follows: Plot the two times on the circumference of a circle where 0 is at the top and 0x80000000 is at the bottom. If the shortest arc from time A to time B runs clockwise around the circle, then time A is less than time B. If the shortest arc from A to B runs counter-clockwise, then time A is greater than time B.

### Assumptions, External Events, and Notes:

None

### Parameters

in	<i>TimeA</i>	The first time to compare.
in	<i>TimeB</i>	The second time to compare.

### Returns

The result of comparing the two times.

### Return values

<a href="#"><i>CFE_TIME_EQUAL</i></a>	The two specified times are considered to be equal.
<a href="#"><i>CFE_TIME_A_GT_B</i></a>	The first specified time is considered to be after the second specified time.
<a href="#"><i>CFE_TIME_A_LT_B</i></a>	The first specified time is considered to be before the second specified time.

### See also

[CFE\\_TIME\\_Add](#), [CFE\\_TIME\\_Subtract](#)

### 10.47.2.3 CFE\_TIME\_Subtract()

```
CFE_TIME_SysTime_t CFE_TIME_Subtract (
    CFE_TIME_SysTime_t Time1,
    CFE_TIME_SysTime_t Time2 )
```

Subtracts two time values.

### Description

This routine subtracts time2 from time1 and returns the result. The time values can represent either absolute or delta times, but not all combinations make sense.

- AbsTime - AbsTime = DeltaTime
- AbsTime - DeltaTime = AbsTime
- DeltaTime - DeltaTime = DeltaTime
- DeltaTime - AbsTime = garbage

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Time1</i>	The base time.
in	<i>Time2</i>	The time to be subtracted from the base time.

**Returns**

The result of subtracting the two times. If the subtraction results in an underflow, the result will roll over (this is not considered an error).

**See also**

[CFE\\_TIME\\_Add](#), [CFE\\_TIME\\_Compare](#)

## 10.48 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.*

#### 10.48.1 Detailed Description

#### 10.48.2 Function Documentation

##### 10.48.2.1 CFE\_TIME\_MET2SCTime() `CFE_TIME_SysTime_t CFE_TIME_MET2SCTime (` `CFE_TIME_SysTime_t METTime )`

Convert specified MET into Spacecraft Time.

#### Description

This function returns Spacecraft Time given MET. Note that Spacecraft Time is returned as either UTC or TAI depending on whether the mission configuration parameter `CFE_MISSION_TIME_CFG_DEFAULT_UTC` or `CFE_MISSION_TIME_CFG_DEFAULT_TAI` was set to true at compile time.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>METTime</code>	The MET to be converted.
----	----------------------	--------------------------

#### Returns

Spacecraft Time (UTC or TAI) corresponding to the specified MET

#### See also

`CFE_TIME_GetMET`, `CFE_TIME_GetMETseconds`, `CFE_TIME_GetMETsubsecs`, `CFE_TIME_Sub2MicroSecs`, `CFE_TIME_Micro2SubSecs`

##### 10.48.2.2 CFE\_TIME\_Micro2SubSecs() `uint32 CFE_TIME_Micro2SubSecs (` `uint32 MicroSeconds )`

Converts a number of microseconds to an equivalent sub-seconds count.

#### Description

This routine converts from microseconds (each tick is 1e-06 seconds) to a subseconds count (each tick is  $1 / 2^{32}$  seconds).

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>MicroSeconds</i>	The sub-seconds count to convert.
----	---------------------	-----------------------------------

**Returns**

The equivalent number of subseconds. If the number of microseconds passed in is greater than one second, (i.e. > 999,999), the return value is equal to 0xffffffff.

**See also**

[CFE\\_TIME\\_MET2SCTime](#), [CFE\\_TIME\\_Sub2MicroSecs](#),

**10.48.2.3 CFE\_TIME\_Sub2MicroSecs()** `uint32 CFE_TIME_Sub2MicroSecs (`  
`uint32 SubSeconds )`

Converts a sub-seconds count to an equivalent number of microseconds.

**Description**

This routine converts from a sub-seconds count (each tick is  $1 / 2^{32}$  seconds) to microseconds (each tick is  $1e-06$  seconds).

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>SubSeconds</i>	The sub-seconds count to convert.
----	-------------------	-----------------------------------

**Returns**

The equivalent number of microseconds.

**See also**

[CFE\\_TIME\\_MET2SCTime](#), [CFE\\_TIME\\_Micro2SubSecs](#),

## 10.49 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.*

#### 10.49.1 Detailed Description

#### 10.49.2 Function Documentation

##### 10.49.2.1 [CFE\\_TIME\\_ExternalGPS\(\)](#) void CFE\_TIME\_ExternalGPS (

```
    CFE\_TIME\_SysTime\_t NewTime,
    int16 NewLeaps )
```

Provide the time from an external source that has data common to GPS receivers.

#### Description

This routine provides a method to provide cFE TIME with current time data acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration parameter specified window for tone signal and data packet verification.

Internally, cFE TIME will calculate a new STCF as the difference between this new time value and the spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

#### Assumptions, External Events, and Notes:

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_GPS](#), which indicates that the external time data consists of a time value relative to a known epoch, plus a leap seconds value.

#### Parameters

in	<i>NewTime</i>	The MET value at the next (or previous) 1 Hz tone signal.
in	<i>NewLeaps</i>	The Leap Seconds value used to calculate time as UTC.

**See also**

[CFE\\_TIME\\_ExternalTone](#), [CFE\\_TIME\\_ExternalMET](#), [CFE\\_TIME\\_ExternalTime](#)

**10.49.2.2 CFE\_TIME\_ExternalMET()** `void CFE_TIME_ExternalMET (`  
`CFE_TIME_SysTime_t NewMET )`

Provides the Mission Elapsed Time from an external source.

**Description**

This routine provides a method to provide cFE TIME with MET acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration parameter specified window for tone signal and data packet verification.

The MET value at the tone "should" have zero subseconds. Although the interface accepts non-zero values for sub-seconds, it may be harmful to other applications that expect zero subseconds at the moment of the tone. Any decision to use non-zero subseconds should be carefully considered.

**Assumptions, External Events, and Notes:**

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_MET](#), which indicates that the external time data consists of MET.

**Parameters**

in	<i>NewMET</i>	The MET value at the next (or previous) 1 Hz tone signal.
----	---------------	---

**See also**

[CFE\\_TIME\\_ExternalTone](#), [CFE\\_TIME\\_ExternalGPS](#), [CFE\\_TIME\\_ExternalTime](#)

**10.49.2.3 CFE\_TIME\_ExternalTime()** `void CFE_TIME_ExternalTime (`  
`CFE_TIME_SysTime_t NewTime )`

Provide the time from an external source that measures time relative to a known epoch.

**Description**

This routine provides a method to provide cFE TIME with current time data acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration specified window for tone signal and data packet verification.

Internally, cFE TIME will calculate a new STCF as the difference between this new time value and the spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

**Assumptions, External Events, and Notes:**

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is `CFE_PLATFORM_TIME_CFG_SERVER` which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is `CFE_PLATFORM_TIME_CFG_SOURCE` which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is `CFE_PLATFORM_TIME_CFG_SRC_TIME`, which indicates that the external time data consists of a time value relative to a known epoch.

**Parameters**

in	<code>NewTime</code>	The MET value at the next (or previous) 1 Hz tone signal.
----	----------------------	---

**See also**

[CFE\\_TIME\\_ExternalTone](#), [CFE\\_TIME\\_ExternalMET](#), [CFE\\_TIME\\_ExternalGPS](#)

**10.49.2.4 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](#)

**10.49.2.5 CFE\_TIME\_RegisterSynchCallback()** `CFE_Status_t CFE_TIME_RegisterSynchCallback (`  
`CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr )`

Registers a callback function that is called whenever time synchronization occurs.

**Description**

This routine passes a callback function pointer for an Application that wishes to be notified whenever a legitimate time synchronization signal (typically a 1 Hz) is received.

**Assumptions, External Events, and Notes:**

Only a single callback per application is supported, and this function should only be called from a single thread within each application (typically the apps main thread). If an application requires triggering multiple child tasks at 1Hz, it should distribute the timing signal internally, rather than registering for multiple callbacks.

**Parameters**

in	<i>CallbackFuncPtr</i>	Function to call at synchronization interval (must not be null)
----	------------------------	---

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TIME_TOO_MANY_SYNCH_CALLBACKS</i>	Too Many Sync Callbacks.
<i>CFE_TIME_BAD_ARGUMENT</i>	Bad Argument.

**See also**

[CFE\\_TIME\\_UnregisterSynchCallback](#)

**10.49.2.6 CFE\_TIME\_UnregisterSynchCallback()** *CFE\_Status\_t CFE\_TIME\_UnregisterSynchCallback ( CFE\_TIME\_SynchCallbackPtr\_t CallbackFuncPtr )*

Unregisters a callback function that is called whenever time synchronization occurs.

**Description**

This routine removes the specified callback function pointer from the list of Callback functions that are called whenever a time synchronization (typically the 1Hz signal) is received.

**Assumptions, External Events, and Notes:**

Only a single callback per application is supported, and this function should only be called from a single thread within each application (typically the apps main thread).

**Parameters**

in	<i>CallbackFuncPtr</i>	Function to remove from synchronization call list (must not be null)
----	------------------------	--

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TIME_CALLBACK_NOT_REGISTERED</i>	Callback Not Registered.
<i>CFE_TIME_BAD_ARGUMENT</i>	Bad Argument.

See also

[CFE\\_TIME\\_RegisterSyncCallback](#)

## 10.50 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.*

#### 10.50.1 Detailed Description

#### 10.50.2 Function Documentation

##### 10.50.2.1 `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.

##### 10.50.2.2 `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)
- xxxx = subsecond formatted as a decimal fraction (1/4 second = 0.25000)
- \0 = trailing null

**Assumptions, External Events, and Notes:**

- The value of the time argument is simply added to the configuration definitions for the ground epoch and converted into a fixed length string in the buffer provided by the caller.
- A loss of data during the string conversion will occur if the computed year exceeds 9999. However, a year that large would require an unrealistic definition for the ground epoch since the maximum amount of time represented by a [CFE\\_TIME\\_SysTime](#) structure is approximately 136 years.

**Parameters**

out	<i>PrintBuffer</i>	Pointer to a character array (must not be null) of at least <a href="#">CFE_TIME_PRINTED_STRING_SIZE</a> characters in length. *PrintBuffer is the time as a character string as described above.
in	<i>TimeToPrint</i>	The time to print into the character array.

## 10.51 cFE Resource ID base values

### Enumerations

- enum {
   
`CFE_RESOURCEID_ES_TASKID_BASE_OFFSET = OS_OBJECT_TYPE_OS_TASK, CFE_RESOURCEID_ES_APPID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 1, CFE_RESOURCEID_ES_LIBID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 2, CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 3, CFE_RESOURCEID_ES_POOLID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 4, CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 5, CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET = OS_OBJECT_TYPE_USER + 6, CFE_RESOURCEID_CONFIGID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 7`
  
 }
- enum {
   
`CFE_ES_TASKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_TASKID_BASE_OFFSET), CFE_ES_APPID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_APPID_BASE_OFFSET), CFE_ES_LIBID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_LIBID_BASE_OFFSET), CFE_ES_COUNTID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET), CFE_ES_POOLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_POOLID_BASE_OFFSET), CFE_ES_CDSBLOCKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET), CFE_SB_PIPEID_RESOURCE_BASE_OFFSET = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET), CFE_CONFIGID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_CONFIGID_BASE_OFFSET)`
}

#### 10.51.1 Detailed Description

#### 10.51.2 Enumeration Type Documentation

##### 10.51.2.1 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.

##### 10.51.2.2 anonymous enum anonymous enum

###### Enumerator

CFE_ES_TASKID_BASE
CFE_ES_APPID_BASE
CFE_ES_LIBID_BASE
CFE_ES_COUNTID_BASE

**Enumerator**

CFE_ES_POOLID_BASE	
CFE_ES_CDSBLOCKID_BASE	
CFE_SB_PIPEID_BASE	
CFE_CONFIGID_BASE	

Definition at line 80 of file `cfe_core_resourceid_basevalues.h`.

## 10.52 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.*

### 10.52.1 Detailed Description

### 10.52.2 Macro Definition Documentation

#### 10.52.2.1 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 41 of file default\_cfe\_time\_msgdefs.h.

#### 10.52.2.2 CFE\_TIME\_FLAG\_ADDADJ #define CFE\_TIME\_FLAG\_ADDADJ 0x0200

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

Definition at line 40 of file default\_cfe\_time\_msgdefs.h.

#### 10.52.2.3 CFE\_TIME\_FLAG\_ADDTCL #define CFE\_TIME\_FLAG\_ADDTCL 0x0080

Time Client Latency is applied in a positive direction.

Definition at line 42 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.4 CFE\_TIME\_FLAG\_CLKSET** #define CFE\_TIME\_FLAG\_CLKSET 0x8000

The spacecraft time has been set.

Definition at line 34 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.5 CFE\_TIME\_FLAG\_CMDFLY** #define CFE\_TIME\_FLAG\_CMDFLY 0x0400

This instance of Time Services was commanded into flywheel mode.

Definition at line 39 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.6 CFE\_TIME\_FLAG\_FLYING** #define CFE\_TIME\_FLAG\_FLYING 0x4000

This instance of Time Services is flywheeling.

Definition at line 35 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.7 CFE\_TIME\_FLAG\_GDTONE** #define CFE\_TIME\_FLAG\_GDTONE 0x0020

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

Definition at line 44 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.8 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 45 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.9 CFE\_TIME\_FLAG\_SERVER** #define CFE\_TIME\_FLAG\_SERVER 0x0040

This instance of Time Services is a Time Server.

Definition at line 43 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.10 CFE\_TIME\_FLAG\_SIGPRI** #define CFE\_TIME\_FLAG\_SIGPRI 0x1000

The clock signal is set to "primary".

Definition at line 37 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.11 CFE\_TIME\_FLAG\_SRCINT** #define CFE\_TIME\_FLAG\_SRCINT 0x2000

The clock source is set to "internal".

Definition at line 36 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.12 CFE\_TIME\_FLAG\_SRVFLY** #define CFE\_TIME\_FLAG\_SRVFLY 0x0800

The Time Server is in flywheel mode.

Definition at line 38 of file default\_cfe\_time\_msgdefs.h.

**10.52.2.13 CFE\_TIME\_FLAG\_UNUSED** #define CFE\_TIME\_FLAG\_UNUSED 0x000F

Reserved flags - should be zero.

Definition at line 47 of file default\_cfe\_time\_msgdefs.h.

## 10.53 OSAL Semaphore State Defines

### Macros

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

#### 10.53.1 Detailed Description

#### 10.53.2 Macro Definition Documentation

##### 10.53.2.1 OS\_SEM\_EMPTY `#define OS_SEM_EMPTY 0`

Semaphore empty state.

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

##### 10.53.2.2 OS\_SEM\_FULL `#define OS_SEM_FULL 1`

Semaphore full state.

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

## 10.54 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.*

#### 10.54.1 Detailed Description

#### 10.54.2 Function Documentation

**10.54.2.1 OS\_BinSemCreate()** `int32 OS_BinSemCreate (`  
    `osal_id_t * sem_id,`  
    `const char * sem_name,`  
    `uint32 sem_initial_value,`  
    `uint32 options )`

Creates a binary semaphore.

Creates a binary semaphore with initial value specified by `sem_initial_value` and name specified by `sem_name`. `sem_id` will be returned to the caller

#### Parameters

<code>out</code>	<code>sem_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>sem_name</code>	the name of the new resource to create (must not be null)
<code>in</code>	<code>sem_initial_value</code>	the initial value of the binary semaphore
<code>in</code>	<code>options</code>	Reserved for future use, should be passed as 0.

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if sem name or sem_id are NULL

## Return values

<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NO_FREE_IDS</code>	if all of the semaphore ids are taken
<code>OS_ERR_NAME_TAKEN</code>	if this is already the name of a binary semaphore
<code>OS_SEM_FAILURE</code>	if the OS call failed (return value only verified in coverage test)

```
10.54.2.2 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	<code>sem_id</code>	The object ID to delete
----	---------------------	-------------------------

## Returns

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

## Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid binary semaphore
<code>OS_SEM_FAILURE</code>	if an unspecified failure occurs (return value only verified in coverage test)

```
10.54.2.3 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	<code>sem_id</code>	The object ID to operate on
----	---------------------	-----------------------------

## Returns

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

## Return values

<code>OS_SUCCESS</code>	Successful execution.
-------------------------	-----------------------

**Return values**

<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a binary semaphore
<code>OS_SEM_FAILURE</code>	if an unspecified failure occurs (return value only verified in coverage test)

```
10.54.2.4 OS_BinSemGetIdByName() int32 OS_BinSemGetIdByName (
    osal_id_t * sem_id,
    const char * sem_name )
```

Find an existing semaphore ID by name.

This function tries to find a binary sem Id given the name of a bin\_sem The id is returned through sem\_id

**Parameters**

out	<code>sem_id</code>	will be set to the ID of the existing resource
in	<code>sem_name</code>	the name of the existing resource to find (must not be null)

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	is semid or sem_name are NULL pointers
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name was not found in the table

```
10.54.2.5 OS_BinSemGetInfo() int32 OS_BinSemGetInfo (
```

```
    osal_id_t sem_id,
    OS_bin_sem_prop_t * bin_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info( name and creator) about the specified binary semaphore.

**Parameters**

in	<code>sem_id</code>	The object ID to operate on
out	<code>bin_prop</code>	The property object buffer to fill (must not be null)

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
-------------------------	-----------------------

## Return values

<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the bin_prop pointer is null

**10.54.2.6 OS\_BinSemGive()** `int32 OS_BinSemGive (`  
`osal_id_t sem_id )`

Increment the semaphore value.

The function unlocks the semaphore referenced by `sem_id` by performing a semaphore unlock operation on that semaphore. If the semaphore value resulting from this operation is positive, then no threads were blocked waiting for the semaphore to become unlocked; the semaphore value is simply incremented for this semaphore.

## Parameters

in	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

## Returns

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a binary semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified failure occurs (return value only verified in coverage test)

**10.54.2.7 OS\_BinSemTake()** `int32 OS_BinSemTake (`  
`osal_id_t sem_id )`

Decrement the semaphore value.

The locks the semaphore referenced by `sem_id` by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then the calling thread shall not return from the call until it either locks the semaphore or the call is interrupted.

## Parameters

in	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

## Returns

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	the Id passed in is not a valid binary semaphore

**Return values**

<a href="#">OS_SEM_FAILURE</a>	if an unspecified failure occurs (return value only verified in coverage test)
--------------------------------	--

**10.54.2.8 OS\_BinSemTimedWait()** `int32 OS_BinSemTimedWait (`  
`osal_id_t sem_id,`  
`uint32 msecs )`

Decrement the semaphore value with a timeout.

The function locks the semaphore referenced by `sem_id`. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore, this wait shall be terminated when the specified timeout, `msecs`, expires.

**Parameters**

in	<code>sem_id</code>	The object ID to operate on
in	<code>msecs</code>	The maximum amount of time to block, in milliseconds

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_SEM_TIMEOUT</a>	if semaphore was not relinquished in time
<a href="#">OS_ERR_INVALID_ID</a>	if the ID passed in is not a valid semaphore ID
<a href="#">OS_SEM_FAILURE</a>	if an unspecified failure occurs (return value only verified in coverage test)

## 10.55 OSAL BSP low level access APIs

These are for OSAL internal BSP information access to pass any BSP-specific boot/command line/startup arguments through to the application, and return a status code back to the OS after exit.

### Functions

- void `OS_BSP_SetResourceTypeConfig (uint32 ResourceType, uint32 ConfigOptionValue)`
- `uint32 OS_BSP_GetResourceTypeConfig (uint32 ResourceType)`
- `uint32 OS_BSP_GetArgC (void)`
- `char *const * OS_BSP_GetArgV (void)`
- void `OS_BSP_SetExitCode (int32 code)`

#### 10.55.1 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

#### 10.55.2 Function Documentation

**10.55.2.1 OS\_BSP\_GetArgC()** `uint32 OS_BSP_GetArgC (`  
`void )`

**10.55.2.2 OS\_BSP\_GetArgV()** `char* const * OS_BSP_GetArgV (`  
`void )`

**10.55.2.3 OS\_BSP\_GetResourceTypeConfig()** `uint32 OS_BSP_GetResourceTypeConfig (`  
`uint32 ResourceType )`

**10.55.2.4 OS\_BSP\_SetExitCode()** `void OS_BSP_SetExitCode (`  
`int32 code )`

**10.55.2.5 OS\_BSP\_SetResourceTypeConfig()** `void OS_BSP_SetResourceTypeConfig (`  
`uint32 ResourceType,`  
`uint32 ConfigOptionValue )`

## 10.56 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 OS_time_t OS_TimeFromTotalSeconds (int64 tm)`  
*Get an `OS_time_t` interval object from an integer number of seconds.*
- `static int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to millisecond units.*
- `static OS_time_t OS_TimeFromTotalMilliseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of milliseconds.*
- `static int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to microsecond units.*
- `static OS_time_t OS_TimeFromTotalMicroseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of microseconds.*
- `static int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to nanosecond units.*
- `static OS_time_t OS_TimeFromTotalNanoseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of nanoseconds.*
- `static int64 OS_TimeGetFractionalPart (OS_time_t tm)`  
*Get subseconds portion (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetSubsecondsPart (OS_time_t tm)`  
*Get 32-bit normalized subseconds (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetMillisecondsPart (OS_time_t tm)`  
*Get milliseconds portion (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetMicrosecondsPart (OS_time_t tm)`  
*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.*

### 10.56.1 Detailed Description

### 10.56.2 Function Documentation

#### 10.56.2.1 OS\_GetLocalTime() `int32 OS_GetLocalTime(`

```
    OS_time_t * time_struct )
```

Get the local time.

This function gets the local time from the underlying OS.

##### Note

Mission time management typically uses the cFE Time Service

##### Parameters

out	<code>time_struct</code>	An <a href="#">OS_time_t</a> that will be set to the current time (must not be null)
-----	--------------------------	--

##### Returns

Get local time status, see [OSAL Return Code Defines](#)

##### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if time_struct is null

#### 10.56.2.2 OS\_SetLocalTime() `int32 OS_SetLocalTime(`

```
    const OS_time_t * time_struct )
```

Set the local time.

This function sets the local time on the underlying OS.

##### Note

Mission time management typically uses the cFE Time Services

##### Parameters

in	<code>time_struct</code>	An <a href="#">OS_time_t</a> containing the current time (must not be null)
----	--------------------------	---

##### Returns

Set local time status, see [OSAL Return Code Defines](#)

##### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if time_struct is null

```
10.56.2.3 OS_TimeAdd() static OS_time_t OS_TimeAdd (
    OS_time_t time1,
    OS_time_t time2 ) [inline], [static]
```

Computes the sum of two time intervals.

#### Parameters

in	<i>time1</i>	The first interval
in	<i>time2</i>	The second interval

#### Returns

The sum of the two intervals (*time1* + *time2*)

Definition at line 467 of file osapi-clock.h.

References OS\_time\_t::ticks.

```
10.56.2.4 OS_TimeAssembleFromMicroseconds() static OS_time_t OS_TimeAssembleFromMicroseconds (
    int64 seconds,
    uint32 microseconds ) [inline], [static]
```

Assemble/Convert a number of seconds + microseconds into an [OS\\_time\\_t](#) interval.

This creates an [OS\\_time\\_t](#) value using a whole number of seconds and a fractional part in units of microseconds. This is the inverse of [OS\\_TimeGetTotalSeconds\(\)](#) and [OS\\_TimeGetMicrosecondsPart\(\)](#), and should recreate the original [OS\\_time\\_t](#) value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

#### See also

[OS\\_TimeGetTotalSeconds\(\)](#), [OS\\_TimeGetMicrosecondsPart\(\)](#)

#### Parameters

in	<i>seconds</i>	Whole number of seconds
in	<i>microseconds</i>	Number of microseconds (fractional part only)

#### Returns

The input arguments represented as an [OS\\_time\\_t](#) interval

Definition at line 402 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, OS\_TIME\_TICKS\_PER\_USEC, and OS\_time\_t::ticks.

```
10.56.2.5 OS_TimeAssembleFromMilliseconds() static OS_time_t OS_TimeAssembleFromMilliseconds (
    int64 seconds,
    uint32 milliseconds ) [inline], [static]
```

Assemble/Convert a number of seconds + milliseconds into an [OS\\_time\\_t](#) interval.

This creates an [OS\\_time\\_t](#) value using a whole number of seconds and a fractional part in units of milliseconds. This is the inverse of [OS\\_TimeGetTotalSeconds\(\)](#) and [OS\\_TimeGetMillisecondsPart\(\)](#), and should recreate the original [OS\\_time\\_t](#) value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

**See also**

[OS\\_TimeGetTotalSeconds\(\)](#), [OS\\_TimeGetMillisecondsPart\(\)](#)

**Parameters**

in	<i>seconds</i>	Whole number of seconds
in	<i>milliseconds</i>	Number of milliseconds (fractional part only)

**Returns**

The input arguments represented as an [OS\\_time\\_t](#) interval

Definition at line 426 of file osapi-clock.h.

References [OS\\_TIME\\_TICKS\\_PER\\_MSEC](#), [OS\\_TIME\\_TICKS\\_PER\\_SECOND](#), and [OS\\_time\\_t::ticks](#).

**10.56.2.6 OS\_TimeAssembleFromNanoseconds()** static [OS\\_time\\_t](#) OS\_TimeAssembleFromNanoseconds (

```
    int64 seconds,
    uint32 nanoseconds ) [inline], [static]
```

Assemble/Convert a number of seconds + nanoseconds into an [OS\\_time\\_t](#) interval.

This creates an [OS\\_time\\_t](#) value using a whole number of seconds and a fractional part in units of nanoseconds. This is the inverse of [OS\\_TimeGetTotalSeconds\(\)](#) and [OS\\_TimeGetNanosecondsPart\(\)](#), and should recreate the original [OS\\_time\\_t](#) value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

**See also**

[OS\\_TimeGetTotalSeconds\(\)](#), [OS\\_TimeGetNanosecondsPart\(\)](#)

**Parameters**

in	<i>seconds</i>	Whole number of seconds
in	<i>nanoseconds</i>	Number of nanoseconds (fractional part only)

**Returns**

The input arguments represented as an [OS\\_time\\_t](#) interval

Definition at line 378 of file osapi-clock.h.

References [OS\\_TIME\\_TICK\\_RESOLUTION\\_NS](#), [OS\\_TIME\\_TICKS\\_PER\\_SECOND](#), and [OS\\_time\\_t::ticks](#).

**10.56.2.7 OS\_TimeAssembleFromSubseconds()** static [OS\\_time\\_t](#) OS\_TimeAssembleFromSubseconds (

```
    int64 seconds,
    uint32 subseconds ) [inline], [static]
```

Assemble/Convert a number of seconds + subseconds into an [OS\\_time\\_t](#) interval.

This creates an [OS\\_time\\_t](#) value using a whole number of seconds and a fractional part in units of sub-seconds ( $1/2^{32}$ ). This is the inverse of [OS\\_TimeGetTotalSeconds\(\)](#) and [OS\\_TimeGetSubsecondsPart\(\)](#), and should recreate the original [OS\\_time\\_t](#) value from these separate values (aside from any potential conversion losses due to limited resolution of the data types/units).

**See also**

[OS\\_TimeGetTotalSeconds\(\)](#), [OS\\_TimeGetNanosecondsPart\(\)](#)

**Parameters**

in	<i>seconds</i>	Whole number of seconds
in	<i>subseconds</i>	Number of subseconds (32 bit fixed point fractional part)

**Returns**

The input arguments represented as an [OS\\_time\\_t](#) interval

Definition at line 449 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, and OS\_time\_t::ticks.

**10.56.2.8 OS\_TimeFromTotalMicroseconds()** static [OS\\_time\\_t](#) OS\_TimeFromTotalMicroseconds ( int64 tm ) [inline], [static]

Get an [OS\\_time\\_t](#) interval object from a integer number of microseconds.

This is the inverse operation of [OS\\_TimeGetTotalMicroseconds\(\)](#), converting the total number of microseconds into an [OS\\_time\\_t](#) value.

**See also**

[OS\\_TimeGetTotalMicroseconds\(\)](#)

**Parameters**

in	<i>tm</i>	Time interval value, in microseconds
----	-----------	--------------------------------------

**Returns**

[OS\\_time\\_t](#) value representing the interval

Definition at line 216 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_USEC.

**10.56.2.9 OS\_TimeFromTotalMilliseconds()** static [OS\\_time\\_t](#) OS\_TimeFromTotalMilliseconds ( int64 tm ) [inline], [static]

Get an [OS\\_time\\_t](#) interval object from a integer number of milliseconds.

This is the inverse operation of [OS\\_TimeGetTotalMilliseconds\(\)](#), converting the total number of milliseconds into an [OS\\_time\\_t](#) value.

**See also**

[OS\\_TimeGetTotalMilliseconds\(\)](#)

**Parameters**

in	<i>tm</i>	Time interval value, in milliseconds
----	-----------	--------------------------------------

**Returns**

[OS\\_time\\_t](#) value representing the interval

Definition at line 182 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_MSEC.

**10.56.2.10 OS\_TimeFromTotalNanoseconds()** static [OS\\_time\\_t](#) OS\_TimeFromTotalNanoseconds ( int64 tm ) [inline], [static]

Get an [OS\\_time\\_t](#) interval object from a integer number of nanoseconds.

This is the inverse operation of [OS\\_TimeGetTotalNanoseconds\(\)](#), converting the total number of nanoseconds into an [OS\\_time\\_t](#) value.

**See also**

[OS\\_TimeGetTotalNanoseconds\(\)](#)

**Parameters**

in	tm	Time interval value, in nanoseconds
----	----	-------------------------------------

**Returns**

[OS\\_time\\_t](#) value representing the interval

Definition at line 254 of file osapi-clock.h.

References OS\_TIME\_TICK\_RESOLUTION\_NS.

**10.56.2.11 OS\_TimeFromTotalSeconds()** static [OS\\_time\\_t](#) OS\_TimeFromTotalSeconds ( int64 tm ) [inline], [static]

Get an [OS\\_time\\_t](#) interval object from an integer number of seconds.

This is the inverse operation of [OS\\_TimeGetTotalSeconds\(\)](#), converting the total number of seconds into an [OS\\_time\\_t](#) value.

**See also**

[OS\\_TimeGetTotalSeconds\(\)](#)

**Parameters**

in	tm	Time interval value, in seconds
----	----	---------------------------------

**Returns**

[OS\\_time\\_t](#) value representing the interval

Definition at line 148 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND.

**10.56.2.12 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	<code>tm</code>	Time interval value
----	-----------------	---------------------

#### Returns

Fractional/subsecond portion of time interval in ticks

Definition at line 270 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()`.

**10.56.2.13 OS\_TimeGetMicrosecondsPart()** static `uint32` `OS_TimeGetMicrosecondsPart` (  
    `OS_time_t tm`) [inline], [static]

Get microseconds portion (fractional part only) from an `OS_time_t` object.

Extracts the fractional part from a given `OS_time_t` object normalized to units of microseconds.

This function may be used to adapt applications initially implemented using an older OSAL version where `OS_time_t` was a structure containing a "seconds" and "microsecs" field.

This function will obtain a value that is compatible with the "microsecs" field of `OS_time_t` as it was defined in previous versions of OSAL, as well as the "tv\_usecs" field of POSIX-style "struct timeval" values.

#### See also

[OS\\_TimeGetTotalSeconds\(\)](#)

#### Parameters

in	<code>tm</code>	Time interval value
----	-----------------	---------------------

#### Returns

Number of microseconds in time interval

Definition at line 338 of file osapi-clock.h.

References `OS_TIME_TICKS_PER_USEC`, and `OS_TimeGetFractionalPart()`.

Here is the call graph for this function:



**10.56.2.14 OS\_TimeGetMillisecondsPart()** static `uint32` `OS_TimeGetMillisecondsPart` (  
    `OS_time_t tm`) [inline], [static]

Get milliseconds portion (fractional part only) from an `OS_time_t` object.

Extracts the fractional part from a given `OS_time_t` object normalized to units of milliseconds.

See also

[OS\\_TimeGetTotalSeconds\(\)](#)

Parameters

in	<i>tm</i>	Time interval value
----	-----------	---------------------

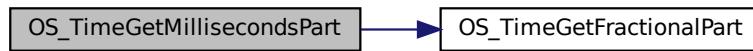
Returns

Number of milliseconds in time interval

Definition at line 313 of file osapi-clock.h.

References `OS_TIME_TICKS_PER_MSEC`, and `OS_TimeGetFractionalPart()`.

Here is the call graph for this function:



#### 10.56.2.15 `OS_TimeGetNanosecondsPart()` static uint32 OS\_TimeGetNanosecondsPart ( `OS_time_t tm` ) [inline], [static]

Get nanoseconds portion (fractional part only) from an `OS_time_t` object.

Extracts the only number of nanoseconds from a given `OS_time_t` object.

This function will obtain a value that is compatible with the "tv\_nsec" field of POSIX-style "struct timespec" values.

See also

[OS\\_TimeGetTotalSeconds\(\)](#)

Parameters

in	<i>tm</i>	Time interval value
----	-----------	---------------------

**Returns**

Number of nanoseconds in time interval

Definition at line 357 of file osapi-clock.h.

References OS\_TIME\_TICK\_RESOLUTION\_NS, and OS\_TimeGetFractionalPart().

Here is the call graph for this function:



**10.56.2.16 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	<code>tm</code>	Time interval value
----	-----------------	---------------------

**Returns**

Fractional/subsecond portion of time interval as 32-bit fixed point value

Definition at line 289 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, and OS\_TimeGetFractionalPart().

Here is the call graph for this function:



**10.56.2.17 OS\_TimeGetTotalMicroseconds()** static `int64` OS\_TimeGetTotalMicroseconds (  
    `OS_time_t tm` ) [inline], [static]

Get interval from an `OS_time_t` object normalized to microsecond units.

Note this refers to the complete interval, not just the fractional part.

**See also**

[OS\\_TimeFromTotalMicroseconds\(\)](#)

**Parameters**

in	tm	Time interval value
----	----	---------------------

**Returns**

Whole number of microseconds in time interval

Definition at line 199 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_USEC, and OS\_time\_t::ticks.

**10.56.2.18 OS\_TimeGetTotalMilliseconds()** static int64 OS\_TimeGetTotalMilliseconds (   
     OS\_time\_t tm ) [inline], [static]

Get interval from an [OS\\_time\\_t](#) object normalized to millisecond units.

Note this refers to the complete interval, not just the fractional part.

**See also**

[OS\\_TimeFromTotalMilliseconds\(\)](#)

**Parameters**

in	tm	Time interval value
----	----	---------------------

**Returns**

Whole number of milliseconds in time interval

Definition at line 165 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_MSEC, and OS\_time\_t::ticks.

**10.56.2.19 OS\_TimeGetTotalNanoseconds()** static int64 OS\_TimeGetTotalNanoseconds (   
     OS\_time\_t tm ) [inline], [static]

Get interval from an [OS\\_time\\_t](#) object normalized to nanosecond units.

Note this refers to the complete interval, not just the fractional part.

**Note**

There is no protection against overflow of the 64-bit return value. Applications must use caution to ensure that the interval does not exceed the representable range of a signed 64 bit integer - approximately 140 years.

**See also**

[OS\\_TimeFromTotalNanoseconds](#)

**Parameters**

in	tm	Time interval value
----	----	---------------------

**Returns**

Whole number of microseconds in time interval

Definition at line 237 of file osapi-clock.h.

References OS\_TIME\_TICK\_RESOLUTION\_NS, and OS\_time\_t::ticks.

**10.56.2.20 OS\_TimeGetTotalSeconds()** static int64 OS\_TimeGetTotalSeconds (

    OS\_time\_t tm ) [inline], [static]

Get interval from an [OS\\_time\\_t](#) object normalized to whole number of seconds.

Extracts the number of whole seconds from a given [OS\\_time\\_t](#) object, discarding any fractional component.

This may also replace a direct read of the "seconds" field from the [OS\\_time\\_t](#) object from previous versions of OSAL, where the structure was defined with separate seconds/microseconds fields.

**See also**

[OS\\_TimeGetMicrosecondsPart\(\)](#)

[OS\\_TimeFromTotalSeconds\(\)](#)

**Parameters**

in	<i>tm</i>	Time interval value
----	-----------	---------------------

**Returns**

Whole number of seconds in time interval

Definition at line 131 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, and OS\_time\_t::ticks.

**10.56.2.21 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	<i>time1</i>	The first interval
in	<i>time2</i>	The second interval

**Returns**

The difference of the two intervals (*time1* - *time2*)

Definition at line 482 of file osapi-clock.h.

References OS\_time\_t::ticks.

## 10.57 OSAL Core Operation APIs

These are for OSAL core operations for startup/initialization, running, and shutdown. Typically only used in bsp's, unit tests, psp's, etc.

### Functions

- void [OS\\_Application\\_Startup](#) (void)  
*Application startup.*
- void [OS\\_Application\\_Run](#) (void)  
*Application run.*
- int32 [OS\\_API\\_Init](#) (void)  
*Initialization of API.*
- void [OS\\_API\\_Teardown](#) (void)  
*Teardown/de-initialization of OSAL API.*
- void [OS\\_IdleLoop](#) (void)  
*Background thread implementation - waits forever for events to occur.*
- void [OS\\_DeleteAllObjects](#) (void)  
*delete all resources created in OSAL.*
- void [OS\\_ApplicationShutdown](#) (uint8 flag)  
*Initiate orderly shutdown.*
- void [OS\\_ApplicationExit](#) (int32 Status)  
*Exit/Abort the application.*
- int32 [OS\\_RegisterEventHandler](#) (OS\_EventHandler\_t handler)  
*Callback routine registration.*

### 10.57.1 Detailed Description

These are for OSAL core operations for startup/initialization, running, and shutdown. Typically only used in bsp's, unit tests, psp's, etc.

Not intended for user application use

### 10.57.2 Function Documentation

#### 10.57.2.1 [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

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERROR</a>	Failed execution. (return value only verified in coverage test)

```
10.57.2.2 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.

```
10.57.2.3 OS_Application_Run() void OS_Application_Run (
    void )
```

Application run.

Run abstraction such that the same BSP can be used for operations and testing.

```
10.57.2.4 OS_Application_Startup() void OS_Application_Startup (
    void )
```

Application startup.

Startup abstraction such that the same BSP can be used for operations and testing.

```
10.57.2.5 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.

```
10.57.2.6 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
----	------	---

```
10.57.2.7 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.

```
10.57.2.8 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.

```
10.57.2.9 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

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if handler is NULL

## 10.58 OSAL Condition Variable APIs

### Functions

- `int32 OS_CondVarCreate (osal_id_t *var_id, const char *var_name, uint32 options)`  
*Creates a condition variable resource.*
- `int32 OS_CondVarLock (osal_id_t var_id)`  
*Locks/Acquires the underlying mutex associated with a condition variable.*
- `int32 OS_CondVarUnlock (osal_id_t var_id)`  
*Unlocks/Releases the underlying mutex associated with a condition variable.*
- `int32 OS_CondVarSignal (osal_id_t var_id)`  
*Signals the condition variable resource referenced by var\_id.*
- `int32 OS_CondVarBroadcast (osal_id_t var_id)`  
*Broadcasts the condition variable resource referenced by var\_id.*
- `int32 OS_CondVarWait (osal_id_t var_id)`  
*Waits on the condition variable object referenced by var\_id.*
- `int32 OS_CondVarTimedWait (osal_id_t var_id, const OS_time_t *abs_wakeup_time)`  
*Time-limited wait on the condition variable object referenced by var\_id.*
- `int32 OS_CondVarDelete (osal_id_t var_id)`  
*Deletes the specified condition variable.*
- `int32 OS_CondVarGetIdByName (osal_id_t *var_id, const char *var_name)`  
*Find an existing condition variable ID by name.*
- `int32 OS_CondVarGetInfo (osal_id_t var_id, OS_condvar_prop_t *condvar_prop)`  
*Fill a property object buffer with details regarding the resource.*

#### 10.58.1 Detailed Description

#### 10.58.2 Function Documentation

##### 10.58.2.1 OS\_CondVarBroadcast() `int32 OS_CondVarBroadcast (` `osal_id_t var_id )`

Broadcasts the condition variable resource referenced by var\_id.

This function may be used to indicate when the state of a data object has been changed.

If there are threads blocked on the condition variable object referenced by var\_id when this function is called, all threads will be unblocked.

Note that although all threads are unblocked, because the mutex is re-acquired before the wait function returns, only a single task will be testing the condition at a given time. The order with which each blocked task runs is determined by the scheduling policy.

#### Parameters

in	<code>var←_id</code>	The object ID to operate on
----	----------------------	-----------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid condition variable

#### 10.58.2.2 `OS_CondVarCreate()`

```
int32 OS_CondVarCreate (
    osal_id_t * var_id,
    const char * var_name,
    uint32 options )
```

Creates a condition variable resource.

A condition variable adds a more sophisticated synchronization option for mutexes, such that it can operate on arbitrary user-defined conditions rather than simply a counter or boolean (as in the case of simple semaphores).

Creating a condition variable resource in OSAL will in turn create both a basic mutex as well as a synchronization overlay. The underlying mutex is similar to the mutex functionality provided by the OSAL mutex subsystem, and can be locked and unlocked normally.

This mutex is intended to protect access to any arbitrary user-defined data object that serves as the condition being tested.

A task that needs a particular state of the object should follow this general flow:

- Lock the underlying mutex
- Test for the condition being waited for (a user-defined check on user-defined data)
- If condition IS NOT met, then call `OS_CondVarWait()` to wait, then repeat test
- If condition IS met, then unlock the underlying mutex and continue

A task that changes the state of the object should follow this general flow:

- Lock the underlying mutex
- Change the state as necessary
- Call either `OS_CondVarSignal()` or `OS_CondVarBroadcast()`
- Unlock the underlying mutex

#### Parameters

<code>out</code>	<code>var_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>var_name</code>	the name of the new resource to create (must not be null)
<code>in</code>	<code>options</code>	reserved for future use. Should be passed as 0.

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if var_id or var_name are NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>

**Return values**

<a href="#">OS_ERR_NO_FREE_IDS</a>	if there are no more free condition variable IDs
<a href="#">OS_ERR_NAME_TAKEN</a>	if there is already a condition variable with the same name

**10.58.2.3 OS\_CondVarDelete()** `int32 OS_CondVarDelete ( osal_id_t var_id )`

Deletes the specified condition variable.

Delete the condition variable and releases any related system resources.

**Parameters**

in	<i>var_id</i>	The object ID to delete
----	---------------	-------------------------

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the id passed in is not a valid condvar

**10.58.2.4 OS\_CondVarGetIdByName()** `int32 OS_CondVarGetIdByName ( osal_id_t * var_id, const char * var_name )`

Find an existing condition variable ID by name.

This function tries to find an existing condition variable ID given the name. The id is returned through var\_id.

**Parameters**

out	<i>var_id</i>	will be set to the ID of the existing resource
in	<i>var_name</i>	the name of the existing resource to find (must not be null)

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	is var_id or var_name are NULL pointers
<a href="#">OS_ERR_NAME_TOO_LONG</a>	name length including null terminator greater than <a href="#">OS_MAX_API_NAME</a>
<a href="#">OS_ERR_NAME_NOT_FOUND</a>	if the name was not found in the table

```
10.58.2.5 OS_CondVarGetInfo() int32 OS_CondVarGetInfo (
    osal_id_t var_id,
    OS_condvar_prop_t * condvar_prop )
```

Fill a property object buffer with details regarding the resource.

This function will fill a structure to contain the information (name and creator) about the specified condition variable.

#### Parameters

in	<i>var_id</i>	The object ID to operate on
out	<i>condvar_prop</i>	The property object buffer to fill (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the mut_prop pointer is null

```
10.58.2.6 OS_CondVarLock() int32 OS_CondVarLock (
    osal_id_t var_id )
```

Locks/Acquires the underlying mutex associated with a condition variable.

The mutex should always be locked by a task before reading or modifying the data object associated with a condition variable.

#### Note

This lock must be acquired by a task before invoking [OS\\_CondVarWait\(\)](#) or [OS\\_CondVarTimedWait\(\)](#) on the same condition variable.

#### Parameters

in	<i>var_id</i>	The object ID to operate on
----	---------------	-----------------------------

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid condition variable

**10.58.2.7 OS\_CondVarSignal()** `int32 OS_CondVarSignal (`  
    `osal_id_t var_id )`

Signals the condition variable resource referenced by var\_id.

This function may be used to indicate when the state of a data object has been changed.

If there are threads blocked on the condition variable object referenced by var\_id when this function is called, one of those threads will be unblocked, as determined by the scheduling policy.

#### Parameters

in	<i>var_id</i>	The object ID to operate on
----	---------------	-----------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid condition variable

**10.58.2.8 OS\_CondVarTimedWait()** `int32 OS_CondVarTimedWait (`  
    `osal_id_t var_id,`  
    `const OS_time_t * abs_wakeup_time )`

Time-limited wait on the condition variable object referenced by var\_id.

Identical in operation to [OS\\_CondVarWait\(\)](#), except that the maximum amount of time that the task will be blocked is limited.

The abs\_wakeup\_time refers to the absolute time of the system clock at which the task should be unblocked to run, regardless of the state of the condition variable. This refers to the same system clock that is the subject of the [OS\\_GetLocalTime\(\)](#) API.

#### Parameters

in	<i>var_id</i>	The object ID to operate on
in	<i>abs_wakeup_time</i>	The system time at which the task should be unblocked (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	the id passed in is not a valid condvar

**10.58.2.9 OS\_CondVarUnlock()** `int32 OS_CondVarUnlock (`  
    `osal_id_t var_id )`

Unlocks/Releases the underlying mutex associated with a condition variable.  
The mutex should be unlocked by a task once reading or modifying the data object associated with a condition variable is complete.

#### Parameters

in	<i>var←_id</i>	The object ID to operate on
----	----------------	-----------------------------

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid condition variable

**10.58.2.10 OS\_CondVarWait()** `int32 OS_CondVarWait ( osal_id_t var_id )`

Waits on the condition variable object referenced by var\_id.

The calling task will be blocked until another task calls the function [OS\\_CondVarSignal\(\)](#) or [OS\\_CondVarBroadcast\(\)](#) on the same condition variable.

The underlying mutex associated with the condition variable must be locked and owned by the calling task at the time this function is invoked. As part of this call, the mutex will be unlocked as the task blocks. This is done in such a way that there is no possibility that another task could acquire the mutex before the calling task has actually blocked.

This atomicity with respect to blocking the task and unlocking the mutex is a critical difference between condition variables and other synchronization primitives. It avoids a window of opportunity where inherent in the simpler synchronization resource types where the state of the data could change between the time that the calling task tested the state and the time that the task actually blocks on the sync resource.

#### Parameters

in	<i>var←_id</i>	The object ID to operate on
----	----------------	-----------------------------

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	the id passed in is not a valid condvar

## 10.59 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.*

#### 10.59.1 Detailed Description

#### 10.59.2 Function Documentation

**10.59.2.1 OS\_CountSemCreate()** `int32 OS_CountSemCreate (`  
    `osal_id_t * sem_id,`  
    `const char * sem_name,`  
    `uint32 sem_initial_value,`  
    `uint32 options )`

Creates a counting semaphore.

Creates a counting semaphore with initial value specified by `sem_initial_value` and name specified by `sem_name`. `sem_id` will be returned to the caller.

#### Note

Underlying RTOS implementations may or may not impose a specific upper limit to the value of a counting semaphore. If the OS has a specific limit and the `sem_initial_value` exceeds this limit, then `OS_INVALID_SEM_VALUE` is returned. On other implementations, any 32-bit integer value may be acceptable. For maximum portability, it is recommended to keep counting semaphore values within the range of a "short int" (i.e. between 0 and 32767). Many platforms do accept larger values, but may not be guaranteed.

#### Parameters

out	<code>sem_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<code>sem_name</code>	the name of the new resource to create (must not be null)
in	<code>sem_initial_value</code>	the initial value of the counting semaphore
in	<code>options</code>	Reserved for future use, should be passed as 0.

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if sem name or sem_id are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NO_FREE_IDS</i>	if all of the semaphore ids are taken
<i>OS_ERR_NAME_TAKEN</i>	if this is already the name of a counting semaphore
<i>OS_INVALID_SEM_VALUE</i>	if the semaphore value is too high (return value only verified in coverage test)
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

**10.59.2.2 OS\_CountSemDelete()** `int32 OS_CountSemDelete (`

`osal_id_t sem_id )`

Deletes the specified counting Semaphore.

**Parameters**

<i>in</i>	<i>sem_id</i>	The object ID to delete
-----------	---------------	-------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

**10.59.2.3 OS\_CountSemGetIdByName()** `int32 OS_CountSemGetIdByName (`

`osal_id_t * sem_id,`  
`const char * sem_name )`

Find an existing semaphore ID by name.

This function tries to find a counting sem id given the name of a count\_sem. The id is returned through sem\_id

**Parameters**

<i>out</i>	<i>sem_id</i>	will be set to the ID of the existing resource
<i>in</i>	<i>sem_name</i>	the name of the existing resource to find (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	is semid or sem_name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table

**10.59.2.4 OS\_CountSemGetInfo()** `int32 OS_CountSemGetInfo (`

```
    osal_id_t sem_id,
    OS_count_sem_prop_t * count_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info( name and creator) about the specified counting semaphore.

**Parameters**

<i>in</i>	<i>sem_id</i>	The object ID to operate on
<i>out</i>	<i>count_prop</i>	The property object buffer to fill (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the count_prop pointer is null

**10.59.2.5 OS\_CountSemGive()** `int32 OS_CountSemGive (`

```
    osal_id_t sem_id )
```

Increment the semaphore value.

The function unlocks the semaphore referenced by *sem\_id* by performing a semaphore unlock operation on that semaphore. If the semaphore value resulting from this operation is positive, then no threads were blocked waiting for the semaphore to become unlocked; the semaphore value is simply incremented for this semaphore.

**Parameters**

<i>in</i>	<i>sem_id</i>	The object ID to operate on
-----------	---------------	-----------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

**10.59.2.6 OS\_CountSemTake()** `int32 OS_CountSemTake ( osal_id_t sem_id )`

Decrement the semaphore value.

The locks the semaphore referenced by `sem_id` by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then the calling thread shall not return from the call until it either locks the semaphore or the call is interrupted.

**Parameters**

in	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	the Id passed in is not a valid counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

**10.59.2.7 OS\_CountSemTimedWait()** `int32 OS_CountSemTimedWait ( osal_id_t sem_id, uint32 msecs )`

Decrement the semaphore value with timeout.

The function locks the semaphore referenced by `sem_id`. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore, this wait shall be terminated when the specified timeout, `msecs`, expires.

**Parameters**

in	<i>sem_id</i>	The object ID to operate on
in	<i>msecs</i>	The maximum amount of time to block, in milliseconds

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_SEM_TIMEOUT</i>	if semaphore was not relinquished in time
<i>OS_ERR_INVALID_ID</i>	if the ID passed in is not a valid semaphore ID
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

## 10.60 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.*

#### 10.60.1 Detailed Description

#### 10.60.2 Function Documentation

##### 10.60.2.1 OS\_DirectoryClose() `int32 OS_DirectoryClose (` `osal_id_t dir_id )`

Closes an open directory.

The directory referred to by `dir_id` will be closed

#### Parameters

in	<code>dir_id</code>	The handle ID of the directory
----	---------------------	--------------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the directory handle is invalid

##### 10.60.2.2 OS\_DirectoryOpen() `int32 OS_DirectoryOpen (` `osal_id_t * dir_id,` `const char * path )`

Opens a directory.

Prepares for reading the files within a directory

**Parameters**

out	<i>dir_id</i>	Location to store handle ID of the directory (must not be null)
in	<i>path</i>	The directory to open (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if dir_id or path is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the path argument exceeds the maximum length
<i>OS_FS_ERR_PATH_INVALID</i>	if the path argument is not valid
<i>OS_ERROR</i>	if the directory could not be opened

**10.60.2.3 OS\_DirectoryRead()** `int32 OS_DirectoryRead (`

```
    osal_id_t dir_id,
    os_dirent_t * dirent )
```

Reads the next name in the directory.

Obtains directory entry data for the next file from an open directory

**Parameters**

in	<i>dir_id</i>	The handle ID of the directory
out	<i>dirent</i>	Buffer to store directory entry information (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if dirent argument is NULL
<i>OS_ERR_INVALID_ID</i>	if the directory handle is invalid
<i>OS_ERROR</i>	at the end of the directory or if the OS call otherwise fails

**10.60.2.4 OS\_DirectoryRewind()** `int32 OS_DirectoryRewind (`

```
    osal_id_t dir_id )
```

Rewinds an open directory.

Resets a directory read handle back to the first file.

**Parameters**

in	<i>dir_id</i>	The handle ID of the directory
----	---------------	--------------------------------

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the directory handle is invalid

```
10.60.2.5 OS_mkdir() int32 OS_mkdir (
    const char * path,
    uint32 access )
```

Makes a new directory.

Makes a directory specified by path.

**Parameters**

in	<i>path</i>	The new directory name (must not be null)
in	<i>access</i>	The permissions for the directory (reserved for future use)

**Note**

Current implementations do not utilize the "access" parameter. Applications should still pass the intended value ([OS\\_READ\\_WRITE](#) or [OS\\_READ\\_ONLY](#)) to be compatible with future implementations.

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if path is NULL
<a href="#">OS_FS_ERR_PATH_TOO_LONG</a>	if the path is too long to be stored locally
<a href="#">OS_FS_ERR_PATH_INVALID</a>	if path cannot be parsed
<a href="#">OS_ERROR</a>	if the OS call fails (return value only verified in coverage test)

```
10.60.2.6 OS_rmdir() int32 OS_rmdir (
    const char * path )
```

Removes a directory from the file system.

Removes a directory from the structure. The directory must be empty prior to this operation.

**Parameters**

in	<i>path</i>	The directory to remove
----	-------------	-------------------------

**Returns**

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

**Return values**

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_INVALID_POINTER</i></a>	if path is NULL
<a href="#"><i>OS_FS_ERR_PATH_INVALID</i></a>	if path cannot be parsed
<a href="#"><i>OS_FS_ERR_PATH_TOO_LONG</i></a>	
<a href="#"><i>OS_ERROR</i></a>	if the directory remove operation failed (return value only verified in coverage test)

## 10.61 OSAL Return Code Defines

The specific status/return code definitions listed in this section may be extended or refined in future versions of OSAL.

### Macros

- `#define OS_SUCCESS (0)`  
*Successful execution.*
- `#define OS_ERROR (-1)`  
*Failed execution.*
- `#define OS_INVALID_POINTER (-2)`  
*Invalid pointer.*
- `#define OS_ERROR_ADDRESS_MISALIGNED (-3)`  
*Address misalignment.*
- `#define OS_ERROR_TIMEOUT (-4)`  
*Error timeout.*
- `#define OS_INVALID_INT_NUM (-5)`  
*Invalid interrupt number.*
- `#define OS_SEM_FAILURE (-6)`  
*Semaphore failure.*
- `#define OS_SEM_TIMEOUT (-7)`  
*Semaphore timeout.*
- `#define OS_QUEUE_EMPTY (-8)`  
*Queue empty.*
- `#define OS_QUEUE_FULL (-9)`  
*Queue full.*
- `#define OS_QUEUE_TIMEOUT (-10)`  
*Queue timeout.*
- `#define OS_QUEUE_INVALID_SIZE (-11)`  
*Queue invalid size.*
- `#define OS_QUEUE_ID_ERROR (-12)`  
*Queue ID error.*
- `#define OS_ERR_NAME_TOO_LONG (-13)`  
*name length including null terminator greater than `OS_MAX_API_NAME`*
- `#define OS_ERR_NO_FREE_IDS (-14)`  
*No free IDs.*
- `#define OS_ERR_NAME_TAKEN (-15)`  
*Name taken.*
- `#define OS_ERR_INVALID_ID (-16)`  
*Invalid ID.*
- `#define OS_ERR_NAME_NOT_FOUND (-17)`  
*Name not found.*
- `#define OS_ERR_SEM_NOT_FULL (-18)`  
*Semaphore not full.*
- `#define OS_ERR_INVALID_PRIORITY (-19)`  
*Invalid priority.*
- `#define OS_INVALID_SEM_VALUE (-20)`  
*Invalid semaphore value.*

- #define OS\_ERR\_FILE (-27)  
*File error.*
- #define OS\_ERR\_NOT\_IMPLEMENTED (-28)  
*Not implemented.*
- #define OS\_TIMER\_ERR\_INVALID\_ARGS (-29)  
*Timer invalid arguments.*
- #define OS\_TIMER\_ERR\_TIMER\_ID (-30)  
*Timer ID error.*
- #define OS\_TIMER\_ERR\_UNAVAILABLE (-31)  
*Timer unavailable.*
- #define OS\_TIMER\_ERR\_INTERNAL (-32)  
*Timer internal error.*
- #define OS\_ERR\_OBJECT\_IN\_USE (-33)  
*Object in use.*
- #define OS\_ERR\_BAD\_ADDRESS (-34)  
*Bad address.*
- #define OS\_ERR\_INCORRECT\_OBJ\_STATE (-35)  
*Incorrect object state.*
- #define OS\_ERR\_INCORRECT\_OBJ\_TYPE (-36)  
*Incorrect object type.*
- #define OS\_ERR\_STREAM\_DISCONNECTED (-37)  
*Stream disconnected.*
- #define OS\_ERR\_OPERATION\_NOT\_SUPPORTED (-38)  
*Requested operation not support on supplied object(s)*
- #define OS\_ERR\_INVALID\_SIZE (-40)  
*Invalid Size.*
- #define OS\_ERR\_OUTPUT\_TOO\_LARGE (-41)  
*Size of output exceeds limit*
- #define OS\_ERR\_INVALID\_ARGUMENT (-42)  
*Invalid argument value (other than ID or size)*
- #define OS\_FS\_ERR\_PATH\_TOO\_LONG (-103)  
*FS path too long.*
- #define OS\_FS\_ERR\_NAME\_TOO\_LONG (-104)  
*FS name too long.*
- #define OS\_FS\_ERR\_DRIVE\_NOT\_CREATED (-106)  
*FS drive not created.*
- #define OS\_FS\_ERR\_DEVICE\_NOT\_FREE (-107)  
*FS device not free.*
- #define OS\_FS\_ERR\_PATH\_INVALID (-108)  
*FS path invalid.*

### 10.61.1 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...
}
```

### 10.61.2 Macro Definition Documentation

#### 10.61.2.1 OS\_ERR\_BAD\_ADDRESS #define OS\_ERR\_BAD\_ADDRESS (-34)

Bad address.

Definition at line 124 of file osapi-error.h.

#### 10.61.2.2 OS\_ERR\_FILE #define OS\_ERR\_FILE (-27)

File error.

Definition at line 117 of file osapi-error.h.

#### 10.61.2.3 OS\_ERR\_INCORRECT\_OBJ\_STATE #define OS\_ERR\_INCORRECT\_OBJ\_STATE (-35)

Incorrect object state.

Definition at line 125 of file osapi-error.h.

#### 10.61.2.4 OS\_ERR\_INCORRECT\_OBJ\_TYPE #define OS\_ERR\_INCORRECT\_OBJ\_TYPE (-36)

Incorrect object type.

Definition at line 126 of file osapi-error.h.

#### 10.61.2.5 OS\_ERR\_INVALID\_ARGUMENT #define OS\_ERR\_INVALID\_ARGUMENT (-42)

Invalid argument value (other than ID or size)

Definition at line 131 of file osapi-error.h.

**10.61.2.6 OS\_ERR\_INVALID\_ID** #define OS\_ERR\_INVALID\_ID (-16)

Invalid ID.

Definition at line 112 of file osapi-error.h.

**10.61.2.7 OS\_ERR\_INVALID\_PRIORITY** #define OS\_ERR\_INVALID\_PRIORITY (-19)

Invalid priority.

Definition at line 115 of file osapi-error.h.

**10.61.2.8 OS\_ERR\_INVALID\_SIZE** #define OS\_ERR\_INVALID\_SIZE (-40)

Invalid Size.

Definition at line 129 of file osapi-error.h.

**10.61.2.9 OS\_ERR\_NAME\_NOT\_FOUND** #define OS\_ERR\_NAME\_NOT\_FOUND (-17)

Name not found.

Definition at line 113 of file osapi-error.h.

**10.61.2.10 OS\_ERR\_NAME\_TAKEN** #define OS\_ERR\_NAME\_TAKEN (-15)

Name taken.

Definition at line 111 of file osapi-error.h.

**10.61.2.11 OS\_ERR\_NAME\_TOO\_LONG** #define OS\_ERR\_NAME\_TOO\_LONG (-13)

name length including null terminator greater than [OS\\_MAX\\_API\\_NAME](#)

Definition at line 109 of file osapi-error.h.

**10.61.2.12 OS\_ERR\_NO\_FREE\_IDS** #define OS\_ERR\_NO\_FREE\_IDS (-14)

No free IDs.

Definition at line 110 of file osapi-error.h.

**10.61.2.13 OS\_ERR\_NOT\_IMPLEMENTED** #define OS\_ERR\_NOT\_IMPLEMENTED (-28)

Not implemented.

Definition at line 118 of file osapi-error.h.

**10.61.2.14 OS\_ERR\_OBJECT\_IN\_USE** #define OS\_ERR\_OBJECT\_IN\_USE (-33)

Object in use.

Definition at line 123 of file osapi-error.h.

**10.61.2.15 OS\_ERR\_OPERATION\_NOT\_SUPPORTED** #define OS\_ERR\_OPERATION\_NOT\_SUPPORTED (-38)

Requested operation not support on supplied object(s)

Definition at line 128 of file osapi-error.h.

**10.61.2.16 OS\_ERR\_OUTPUT\_TOO\_LARGE** #define OS\_ERR\_OUTPUT\_TOO\_LARGE (-41)  
Size of output exceeds limit

Definition at line 130 of file osapi-error.h.

**10.61.2.17 OS\_ERR\_SEM\_NOT\_FULL** #define OS\_ERR\_SEM\_NOT\_FULL (-18)  
Semaphore not full.  
Definition at line 114 of file osapi-error.h.

**10.61.2.18 OS\_ERR\_STREAM\_DISCONNECTED** #define OS\_ERR\_STREAM\_DISCONNECTED (-37)  
Stream disconnected.  
Definition at line 127 of file osapi-error.h.

**10.61.2.19 OS\_ERROR** #define OS\_ERROR (-1)  
Failed execution.  
Definition at line 97 of file osapi-error.h.

**10.61.2.20 OS\_ERROR\_ADDRESS\_MISALIGNED** #define OS\_ERROR\_ADDRESS\_MISALIGNED (-3)  
Address misalignment.  
Definition at line 99 of file osapi-error.h.

**10.61.2.21 OS\_ERROR\_TIMEOUT** #define OS\_ERROR\_TIMEOUT (-4)  
Error timeout.  
Definition at line 100 of file osapi-error.h.

**10.61.2.22 OS\_FS\_ERR\_DEVICE\_NOT\_FREE** #define OS\_FS\_ERR\_DEVICE\_NOT\_FREE (-107)  
FS device not free.  
Definition at line 144 of file osapi-error.h.

**10.61.2.23 OS\_FS\_ERR\_DRIVE\_NOT\_CREATED** #define OS\_FS\_ERR\_DRIVE\_NOT\_CREATED (-106)  
FS drive not created.  
Definition at line 143 of file osapi-error.h.

**10.61.2.24 OS\_FS\_ERR\_NAME\_TOO\_LONG** #define OS\_FS\_ERR\_NAME\_TOO\_LONG (-104)  
FS name too long.  
Definition at line 142 of file osapi-error.h.

**10.61.2.25 OS\_FS\_ERR\_PATH\_INVALID** #define OS\_FS\_ERR\_PATH\_INVALID (-108)  
FS path invalid.  
Definition at line 145 of file osapi-error.h.

**10.61.2.26 OS\_FS\_ERR\_PATH\_TOO\_LONG** #define OS\_FS\_ERR\_PATH\_TOO\_LONG (-103)  
FS path too long.  
Definition at line 141 of file osapi-error.h.

**10.61.2.27 OS\_INVALID\_INT\_NUM** #define OS\_INVALID\_INT\_NUM (-5)  
Invalid Interrupt number.  
Definition at line 101 of file osapi-error.h.

**10.61.2.28 OS\_INVALID\_POINTER** #define OS\_INVALID\_POINTER (-2)  
Invalid pointer.  
Definition at line 98 of file osapi-error.h.

**10.61.2.29 OS\_INVALID\_SEM\_VALUE** #define OS\_INVALID\_SEM\_VALUE (-20)  
Invalid semaphore value.  
Definition at line 116 of file osapi-error.h.

**10.61.2.30 OS\_QUEUE\_EMPTY** #define OS\_QUEUE\_EMPTY (-8)  
Queue empty.  
Definition at line 104 of file osapi-error.h.

**10.61.2.31 OS\_QUEUE\_FULL** #define OS\_QUEUE\_FULL (-9)  
Queue full.  
Definition at line 105 of file osapi-error.h.

**10.61.2.32 OS\_QUEUE\_ID\_ERROR** #define OS\_QUEUE\_ID\_ERROR (-12)  
Queue ID error.  
Definition at line 108 of file osapi-error.h.

**10.61.2.33 OS\_QUEUE\_INVALID\_SIZE** #define OS\_QUEUE\_INVALID\_SIZE (-11)  
Queue invalid size.  
Definition at line 107 of file osapi-error.h.

**10.61.2.34 OS\_QUEUE\_TIMEOUT** #define OS\_QUEUE\_TIMEOUT (-10)  
Queue timeout.  
Definition at line 106 of file osapi-error.h.

**10.61.2.35 OS\_SEM\_FAILURE** #define OS\_SEM\_FAILURE (-6)  
Semaphore failure.  
Definition at line 102 of file osapi-error.h.

**10.61.2.36 OS\_SEM\_TIMEOUT** #define OS\_SEM\_TIMEOUT (-7)  
Semaphore timeout.  
Definition at line 103 of file osapi-error.h.

**10.61.2.37 OS\_SUCCESS** #define OS\_SUCCESS (0)  
Successful execution.  
Definition at line 96 of file osapi-error.h.

**10.61.2.38 OS\_TIMER\_ERR\_INTERNAL** #define OS\_TIMER\_ERR\_INTERNAL (-32)  
Timer internal error.  
Definition at line 122 of file osapi-error.h.

**10.61.2.39 OS\_TIMER\_ERR\_INVALID\_ARGS** #define OS\_TIMER\_ERR\_INVALID\_ARGS (-29)  
Timer invalid arguments.  
Definition at line 119 of file osapi-error.h.

**10.61.2.40 OS\_TIMER\_ERR\_TIMER\_ID** #define OS\_TIMER\_ERR\_TIMER\_ID (-30)  
Timer ID error.  
Definition at line 120 of file osapi-error.h.

**10.61.2.41 OS\_TIMER\_ERR\_UNAVAILABLE** #define OS\_TIMER\_ERR\_UNAVAILABLE (-31)  
Timer unavailable.  
Definition at line 121 of file osapi-error.h.

## 10.62 OSAL Error Info APIs

### Functions

- static long `OS_StatusToInteger (osal_status_t Status)`  
*Convert a status code to a native "long" type.*
- `int32 OS_GetErrorName (int32 error_num, os_err_name_t *err_name)`  
*Convert an error number to a string.*
- `char * OS_StatusToString (osal_status_t status, os_status_string_t *status_string)`  
*Convert status to a string.*

#### 10.62.1 Detailed Description

#### 10.62.2 Function Documentation

**10.62.2.1 OS\_GetErrorName()** `int32 OS_GetErrorName (`  
    `int32 error_num,`  
    `os_err_name_t * err_name )`

Convert an error number to a string.

##### Parameters

in	<code>error_num</code>	Error number to convert
out	<code>err_name</code>	Buffer to store error string

##### Returns

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

##### Return values

<code>OS_SUCCESS</code>	if successfully converted to a string
<code>OS_INVALID_POINTER</code>	if err_name is NULL
<code>OS_ERROR</code>	if error could not be converted

**10.62.2.2 OS\_StatusToInteger()** `static long OS_StatusToInteger (`  
    `osal_status_t Status ) [inline], [static]`

Convert a status code to a native "long" type.

For printing or logging purposes, this converts the given status code to a "long" (signed integer) value. It should be used in conjunction with the "%ld" conversion specifier in printf-style statements.

##### Parameters

in	<code>Status</code>	Execution status, see <a href="#">OSAL Return Code Defines</a>
----	---------------------	--

**Returns**

Same status value converted to the "long" data type

Definition at line 164 of file osapi-error.h.

**10.62.2.3 OS\_StatusToString()** `char* OS_StatusToString (`  
`osal_status_t status,`  
`os_status_string_t * status_string )`

Convert status to a string.

**Parameters**

in	<i>status</i>	Status value to convert
out	<i>status_string</i>	Buffer to store status converted to string

**Returns**

Passed in string pointer

## 10.63 OSAL File Access Option Defines

### Macros

- #define OS\_READ\_ONLY 0
- #define OS\_WRITE\_ONLY 1
- #define OS\_READ\_WRITE 2

#### 10.63.1 Detailed Description

#### 10.63.2 Macro Definition Documentation

##### 10.63.2.1 OS\_READ\_ONLY #define OS\_READ\_ONLY 0

Read only file access

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

##### 10.63.2.2 OS\_READ\_WRITE #define OS\_READ\_WRITE 2

Read write file access

Definition at line 37 of file osapi-file.h.

##### 10.63.2.3 OS\_WRITE\_ONLY #define OS\_WRITE\_ONLY 1

Write only file access

Definition at line 36 of file osapi-file.h.

## 10.64 OSAL Reference Point For Seek Offset Defines

### Macros

- #define OS\_SEEK\_SET 0
- #define OS\_SEEK\_CUR 1
- #define OS\_SEEK\_END 2

#### 10.64.1 Detailed Description

#### 10.64.2 Macro Definition Documentation

##### 10.64.2.1 OS\_SEEK\_CUR #define OS\_SEEK\_CUR 1

Seek offset current

Definition at line 44 of file osapi-file.h.

##### 10.64.2.2 OS\_SEEK\_END #define OS\_SEEK\_END 2

Seek offset end

Definition at line 45 of file osapi-file.h.

##### 10.64.2.3 OS\_SEEK\_SET #define OS\_SEEK\_SET 0

Seek offset set

Definition at line 43 of file osapi-file.h.

## 10.65 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.*

### 10.65.1 Detailed Description

### 10.65.2 Function Documentation

**10.65.2.1 OS\_chmod()** `int32 OS_chmod (`  
    `const char * path,`  
    `uint32 access_mode )`

Changes the permissions of a file.

**Parameters**

in	<i>path</i>	File to change (must not be null)
in	<i>access_mode</i>	Desired access mode - see <a href="#">OSAL File Access Option Defines</a>

**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**

<a href="#">OS_SUCCESS</a>	Successful execution. (return value only verified in coverage test)
<a href="#">OS_ERR_NOT_IMPLEMENTED</a>	if the filesystem does not support this call
<a href="#">OS_INVALID_POINTER</a>	if the path argument is NULL

**10.65.2.2 OS\_close()** `int32 OS_close ( osal_id_t filedes )`

Closes an open file handle.

This closes regular file handles and any other file-like resource, such as network streams or pipes.

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
----	----------------	-----------------------------

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the file descriptor passed in is invalid
<a href="#">OS_ERROR</a>	if an unexpected/unhandled error occurs (return value only verified in coverage test)

Referenced by CS\_TableInit().

**10.65.2.3 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**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERROR</a>	if one or more file close returned an error (return value only verified in coverage test)

**10.65.2.4 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**

<code>in</code>	<code>Filename</code>	The file to close (must not be null)
-----------------	-----------------------	--------------------------------------

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_FS_ERR_PATH_INVALID</a>	if the file is not found
<a href="#">OS_ERROR</a>	if the file close returned an error (return value only verified in coverage test)
<a href="#">OS_INVALID_POINTER</a>	if the filename argument is NULL

**10.65.2.5 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**

<code>in</code>	<code>src</code>	The source file to operate on (must not be null)
<code>in</code>	<code>dest</code>	The destination file (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the file could not be accessed
<i>OS_INVALID_POINTER</i>	if src or dest are NULL
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the paths given are too long to be stored locally
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the dest name is too long to be stored locally

**10.65.2.6 OS\_FDGetInfo()** `int32 OS_FDGetInfo ( osal_id_t filedes, OS_file_prop_t * fd_prop )`

Obtain information about an open file.

Copies the information of the given file descriptor into a structure passed in

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
out	<i>fd_prop</i>	Storage buffer for file information (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_INVALID_POINTER</i>	if the fd_prop argument is NULL

**10.65.2.7 OS\_FileOpenCheck()** `int32 OS_FileOpenCheck ( const char * Filename )`

Checks to see if a file is open.

This function takes a filename and determines if the file is open. The function will return success if the file is open.

**Parameters**

in	<i>Filename</i>	The file to operate on (must not be null)
----	-----------------	---

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	if the file is open
<i>OS_ERROR</i>	if the file is not open
<i>OS_INVALID_POINTER</i>	if the filename argument is NULL

**10.65.2.8 OS\_lseek() `int32 OS_lseek (`**

```
    osal_id_t filedes,  
    int32 offset,  
    uint32 whence )
```

Seeks to the specified position of an open file.

Sets the read/write pointer to a specific offset in a specific file.

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
in	<i>offset</i>	The file offset to seek to
in	<i>whence</i>	The reference point for offset, see <a href="#">OSAL Reference Point For Seek Offset Defines</a>

**Returns**

Byte offset from the beginning of the file or appropriate error code, see [OSAL Return Code Defines](#)

**Return values**

<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_ERROR</i>	if OS call failed (return value only verified in coverage test)

**10.65.2.9 OS\_mv() `int32 OS_mv (`**

```
    const char * src,  
    const char * dest )
```

Move a single file from *src* to *dest*.

This first attempts to rename the file, which is faster if the source and destination reside on the same file system.

If this fails, it falls back to copying the file and removing the original.

**Note**

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

**Parameters**

in	<i>src</i>	The source file to operate on (must not be null)
in	<i>dest</i>	The destination file (must not be null)

**Returns**Execution status, see [OSAL Return Code Defines](#)**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERROR</a>	if the file could not be renamed.
<a href="#">OS_INVALID_POINTER</a>	if src or dest are NULL
<a href="#">OS_FS_ERR_PATH_INVALID</a>	if path cannot be parsed
<a href="#">OS_FS_ERR_PATH_TOO_LONG</a>	if the paths given are too long to be stored locally
<a href="#">OS_FS_ERR_NAME_TOO_LONG</a>	if the dest name is too long to be stored locally

**10.65.2.10 OS\_OpenCreate()** `int32 OS_OpenCreate(`

```
    osal_id_t * filedes,
    const char * path,
    int32 flags,
    int32 access_mode )
```

Open or create a file.

Implements the same as OS\_open/OS\_creat but follows the OSAL paradigm of outputting the ID/descriptor separately from the return value, rather than relying on the user to convert it back.

**Parameters**

out	<i>filedes</i>	The handle ID (OS_OBJECT_ID_UNDEFINED on failure) (must not be null)
in	<i>path</i>	File name to create or open (must not be null)
in	<i>flags</i>	The file permissions - see <a href="#">OS_file_flag_t</a>
in	<i>access_mode</i>	Intended access mode - see <a href="#">OSAL File Access Option Defines</a>

**Returns**Execution status, see [OSAL Return Code Defines](#)**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERROR</a>	if the command was not executed properly
<a href="#">OS_INVALID_POINTER</a>	if pointer argument was NULL
<a href="#">OS_ERR_NO_FREE_IDS</a>	if all available file handles are in use
<a href="#">OS_FS_ERR_NAME_TOO_LONG</a>	if the filename portion of the path exceeds OS_MAX_FILE_NAME
<a href="#">OS_FS_ERR_PATH_INVALID</a>	if the path argument is not valid
<a href="#">OS_FS_ERR_PATH_TOO_LONG</a>	if the path argument exceeds OS_MAX_PATH_LEN

Referenced by CS\_TableInit().

```
10.65.2.11 OS_read() int32 OS_read (
    osal_id_t filedes,
    void * buffer,
    size_t nbytes )
```

Read from a file handle.

Reads up to nbytes from a file, and puts them into buffer.

If the file position is at the end of file (or beyond, if the OS allows) then this function will return 0.

#### Parameters

in	<i>filedes</i>	The handle ID to operate on
out	<i>buffer</i>	Storage location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)

#### Note

All OSAL error codes are negative int32 values. Failure of this call can be checked by testing if the result is less than 0.

#### Returns

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

#### Return values

<i>OS_INVALID_POINTER</i>	if buffer is a null pointer
<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_ERROR</i>	if OS call failed (return value only verified in coverage test)
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
0	if at end of file/stream data

```
10.65.2.12 OS_remove() int32 OS_remove (
    const char * path )
```

Removes a file from the file system.

Removes a given filename from the drive

#### Note

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

#### Parameters

in	<i>path</i>	The file to operate on (must not be null)
----	-------------	---

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if there is no device or the driver returns error
<i>OS_INVALID_POINTER</i>	if path is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if path is too long to be stored locally
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the name of the file to remove is too long

```
10.65.2.13 OS_rename() int32 OS_rename (
    const char * old_filename,
    const char * new_filename )
```

Renames a file.

Changes the name of a file, where the source and destination reside on the same file system.

**Note**

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

**Parameters**

in	<i>old_filename</i>	The original filename (must not be null)
in	<i>new_filename</i>	The desired filename (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the file could not be opened or renamed.
<i>OS_INVALID_POINTER</i>	if old or new are NULL
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the paths given are too long to be stored locally
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the new name is too long to be stored locally

```
10.65.2.14 OS_stat() int32 OS_stat (
    const char * path,
    os_fstat_t * filestats )
```

Obtain information about a file or directory.

Returns information about a file or directory in an [os\\_fstat\\_t](#) structure

#### Parameters

in	<i>path</i>	The file to operate on (must not be null)
out	<i>filestats</i>	Buffer to store file information (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if path or filestats is NULL
<a href="#">OS_FS_ERR_PATH_TOO_LONG</a>	if the path is too long to be stored locally
<a href="#">OS_FS_ERR_NAME_TOO_LONG</a>	if the name of the file is too long to be stored
<a href="#">OS_FS_ERR_PATH_INVALID</a>	if path cannot be parsed
<a href="#">OS_ERROR</a>	if the OS call failed

### 10.65.2.15 OS\_TimedRead() [int32](#) OS\_TimedRead (

```
    osal\_id\_t filedes,  
    void * buffer,  
    size_t nbytes,  
    int32 timeout )
```

File/Stream input read with a timeout.

This implements a time-limited read and is primarily intended for use with sockets but may also work with any other stream-like resource that the underlying OS supports, such as pipes or special devices.

If data is immediately available on the file/socket, this will return that data along with the actual number of bytes that were immediately available. It will not block.

If the file position is at the end of file or end of stream data (e.g. if the remote end has closed the connection), then this function will immediately return 0 without blocking for the timeout period.

If no data is immediately available, but the underlying resource/stream is still connected to a peer, this will wait up to the given timeout for additional data to appear. If no data appears within the timeout period, then this returns the [OS\\_ERROR\\_TIMEOUT](#) status code. This allows the caller to differentiate an open (but idle) socket connection from a connection which has been closed by the remote peer.

In all cases this will return successfully as soon as at least 1 byte of actual data is available. It will not attempt to read the entire input buffer.

If an EOF condition occurs prior to timeout, this function returns zero.

#### Parameters

in	<i>filedes</i>	The handle ID to operate on
out	<i>buffer</i>	Storage location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>timeout</i>	Maximum time to wait, in milliseconds (OS_PEND = forever)

**Returns**

Byte count on success or appropriate error code, see [OSAL Return Code Defines](#)

**Return values**

<i>OS_ERROR_TIMEOUT</i>	if no data became available during timeout period
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_INVALID_POINTER</i>	if the passed-in buffer is not valid
<i>0</i>	if at end of file/stream data

**10.65.2.16 OS\_TimedWrite()** `int32 OS_TimedWrite(`

```
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes,
    int32 timeout )
```

File/Stream output write with a timeout.

This implements a time-limited write and is primarily intended for use with sockets but may also work with any other stream-like resource that the underlying OS supports.

If output buffer space is immediately available on the file/socket, this will place data into the buffer and return the actual number of bytes that were queued for output. It will not block.

If no output buffer space is immediately available, this will wait up to the given timeout for space to become available. If no space becomes available within the timeout period, then this returns an error code (not zero).

In all cases this will return successfully as soon as at least 1 byte of actual data is output. It will *not* attempt to write the entire output buffer.

If an EOF condition occurs prior to timeout, this function returns zero.

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>timeout</i>	Maximum time to wait, in milliseconds (OS_PEND = forever)

**Returns**

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

**Return values**

<i>OS_ERROR_TIMEOUT</i>	if no data became available during timeout period
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_INVALID_POINTER</i>	if the passed-in buffer is not valid
<i>0</i>	if file/stream cannot accept any more data

```
10.65.2.17 OS_write() int32 OS_write (
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes )
```

Write to a file handle.

Writes to a file. copies up to a maximum of nbytes of buffer to the file described in filedes

#### Parameters

in	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)

#### Note

All OSAL error codes are negative int32 values. Failure of this call can be checked by testing if the result is less than 0.

#### Returns

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

#### Return values

<a href="#">OS_INVALID_POINTER</a>	if buffer is NULL
<a href="#">OS_ERR_INVALID_SIZE</a>	if the passed-in size is not valid
<a href="#">OS_ERROR</a>	if OS call failed (return value only verified in coverage test)
<a href="#">OS_ERR_INVALID_ID</a>	if the file descriptor passed in is invalid
0	if file/stream cannot accept any more data

## 10.66 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.*

### 10.66.1 Detailed Description

### 10.66.2 Function Documentation

#### 10.66.2.1 OS\_chkfs() `int32 OS_chkfs (`

```
    const char * name,
    bool repair )
```

Checks the health of a file system and repairs it if necessary.

Checks the drives for inconsistencies and optionally also repairs it

#### Note

not all operating systems implement this function. If the underlying OS does not provide a facility to check the volume, then OS\_ERR\_NOT\_IMPLEMENTED will be returned.

#### Parameters

in	<code>name</code>	The device/path to operate on (must not be null)
in	<code>repair</code>	Whether to also repair inconsistencies

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution. (return value only verified in coverage test)
<i>OS_INVALID_POINTER</i>	Name is NULL
<i>OS_ERR_NOT_IMPLEMENTED</i>	Not implemented.
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the name is too long
<i>OS_ERROR</i>	Failed execution. (return value only verified in coverage test)

```
10.66.2.2 OS_FileSysAddFixedMap() int32 OS_FileSysAddFixedMap (
    osal_id_t * filesystem_id,
    const char * phys_path,
    const char * virt_path )
```

Create a fixed mapping between an existing directory and a virtual OSAL mount point.

This mimics the behavior of a "FS\_BASED" entry in the VolumeTable but is registered at runtime. It is intended to be called by the PSP/BSP prior to starting the application.

**Note**

OSAL virtual mount points are required to be a single, non-empty top-level directory name. Virtual path names always follow the form /<virt\_mount\_point>/<relative\_path>/<file>. Only the relative path may be omitted/empty (i.e. /<virt\_mount\_point>/<file>) but the virtual mount point must be present and not an empty string. In particular this means it is not possible to directly refer to files in the "root" of the native file system from OSAL. However it is possible to create a virtual map to the root, such as by calling:

```
OS_FileSysAddFixedMap (&fs_id, "/", "/root");
```

**Parameters**

out	<i>filesystem_id</i>	A buffer to store the ID of the file system mapping (must not be null)
in	<i>phys_path</i>	The native system directory (an existing mount point) (must not be null)
in	<i>virt_path</i>	The virtual mount point of this filesystem (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the overall <i>phys_path</i> is too long
<i>OS_ERR_NAME_TOO_LONG</i>	if the <i>phys_path</i> basename (filesystem name) is too long
<i>OS_INVALID_POINTER</i>	if any argument is NULL

```
10.66.2.3 OS_FileSysStatVolume() int32 OS_FileSysStatVolume (
    const char * name,
    OS_statvfs_t * statbuf )
```

Obtains information about size and free space in a volume.

Populates the supplied `OS_statvfs_t` structure, which includes the block size and total/free blocks in a file system volume.

This replaces two older OSAL calls:

`OS_fsBlocksFree()` is determined by reading the `blocks_free` output struct member `OS_fsBytesFree()` is determined by multiplying `blocks_free` by the `block_size` member

#### Parameters

in	<i>name</i>	The device/path to operate on (must not be null)
out	<i>statbuf</i>	Output structure to populate (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if name or statbuf is NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the name is too long
<code>OS_ERROR</code>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.66.2.4 OS_FS_GetPhysDriveName() int32 OS_FS_GetPhysDriveName (
```

```
    char * PhysDriveName,
    const char * MountPoint )
```

Obtains the physical drive name associated with a mount point.

Returns the name of the physical volume associated with the drive, when given the OSAL mount point of the drive

#### Parameters

out	<i>PhysDriveName</i>	Buffer to store physical drive name (must not be null)
in	<i>MountPoint</i>	OSAL mount point (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if either parameter is NULL
<code>OS_ERR_NAME_NOT_FOUND</code>	if the MountPoint is not mounted in OSAL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the MountPoint is too long

**10.66.2.5 OS\_GetFsInfo()** `int32 OS_GetFsInfo (`  
    `os_fsinfo_t * filesys_info )`

Returns information about the file system.

Returns information about the file system in an `os_fsinfo_t`. This includes the number of open files and file systems

#### Parameters

<code>out</code>	<code>filesys_info</code>	Buffer to store filesystem information (must not be null)
------------------	---------------------------	---

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if <code>filesys_info</code> is NULL

**10.66.2.6 OS\_initfs()** `int32 OS_initfs (`  
    `char * address,`  
    `const char * devname,`  
    `const char * volname,`  
    `size_t blocksize,`  
    `osal_blockcount_t numblocks )`

Initializes an existing file system.

Initializes a file system on the target.

#### Note

The "volname" parameter of RAM disks should always begin with the string "RAM", e.g. "RAMDISK" or "RA←M0","RAM1", etc if multiple devices are created. The underlying implementation uses this to select the correct filesystem type/format, and this may also be used to differentiate between RAM disks and real physical disks.

#### Parameters

<code>in</code>	<code>address</code>	The address at which to start the new disk. If <code>address == 0</code> , then space will be allocated by the OS
<code>in</code>	<code>devname</code>	The underlying kernel device to use, if applicable. (must not be null)
<code>in</code>	<code>volname</code>	The name of the volume (see note) (must not be null)
<code>in</code>	<code>blocksize</code>	The size of a single block on the drive
<code>in</code>	<code>numblocks</code>	The number of blocks to allocate for the drive

#### Returns

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if devname or volname are NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the name is too long
<i>OS_FS_ERR_DEVICE_NOT_FREE</i>	if the volume table is full
<i>OS_FS_ERR_DRIVE_NOT_CREATED</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.66.2.7 OS_mkfs() int32 OS_mkfs (
    char * address,
    const char * devname,
    const char * volname,
    size_t blocksize,
    osal_blockcount_t numblocks )
```

Makes a file system on the target.

Makes a file system on the target. Highly dependent on underlying OS and dependent on OS volume table definition.

## Note

The "volname" parameter of RAM disks should always begin with the string "RAM", e.g. "RAMDISK" or "RA←M0","RAM1", etc if multiple devices are created. The underlying implementation uses this to select the correct filesystem type/format, and this may also be used to differentiate between RAM disks and real physical disks.

## Parameters

in	<i>address</i>	The address at which to start the new disk. If address == 0 space will be allocated by the OS.
in	<i>devname</i>	The underlying kernel device to use, if applicable. (must not be null)
in	<i>volname</i>	The name of the volume (see note) (must not be null)
in	<i>blocksize</i>	The size of a single block on the drive
in	<i>numblocks</i>	The number of blocks to allocate for the drive

## Returns

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if devname or volname is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the overall devname or volname is too long
<i>OS_FS_ERR_DEVICE_NOT_FREE</i>	if the volume table is full
<i>OS_FS_ERR_DRIVE_NOT_CREATED</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.66.2.8 OS_mount() int32 OS_mount (
```

```
    const char * devname,
    const char * mountpoint )
```

Mounts a file system.

Mounts a file system / block device at the given mount point.

#### Parameters

in	<i>devname</i>	The name of the drive to mount. devname is the same from <a href="#">OS_mkfs</a> (must not be null)
in	<i>mountpoint</i>	The name to call this disk from now on (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_NAME_NOT_FOUND</a>	if the device name does not exist in OSAL
<a href="#">OS_FS_ERR_PATH_TOO_LONG</a>	if the mount point string is too long
<a href="#">OS_INVALID_POINTER</a>	if any argument is NULL
<a href="#">OS_ERROR</a>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

### 10.66.2.9 OS\_rmfs()

```
int32 OS_rmfs(
```

```
    const char * devname )
```

Removes a file system.

This function will remove or un-map the target file system. Note that this is not the same as un-mounting the file system.

#### Parameters

in	<i>devname</i>	The name of the "generic" drive (must not be null)
----	----------------	--

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if devname is NULL
<a href="#">OS_FS_ERR_PATH_TOO_LONG</a>	if the devname is too long
<a href="#">OS_ERR_NAME_NOT_FOUND</a>	if the devname does not exist in OSAL
<a href="#">OS_ERROR</a>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

```
10.66.2.10 OS_TranslatePath() int32 OS_TranslatePath (
    const char * VirtualPath,
    char * LocalPath )
```

Translates an OSAL Virtual file system path to a host Local path.  
 Translates a virtual path to an actual system path name

#### Note

The buffer provided in the LocalPath argument is required to be at least OS\_MAX\_PATH\_LEN characters in length.

#### Parameters

in	<i>VirtualPath</i>	OSAL virtual path name (must not be null)
out	<i>LocalPath</i>	Buffer to store native/translated path name (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if either parameter is NULL
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the filename component is too long
<i>OS_FS_ERR_PATH_INVALID</i>	if either parameter cannot be interpreted as a path
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if either input or output pathnames are too long

```
10.66.2.11 OS_unmount() int32 OS_unmount (
```

```
    const char * mountpoint )
```

Unmounts a mounted file system.

This function will unmount a drive from the file system and make all open file descriptors useless.

#### Note

Any open file descriptors referencing this file system should be closed prior to unmounting a drive

#### Parameters

in	<i>mountpoint</i>	The mount point to remove from <a href="#">OS_mount</a> (must not be null)
----	-------------------	--

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if name is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the absolute path given is too long

**Return values**

<i>OS_ERR_NAME_NOT_FOUND</i>	if the mountpoint is not mounted in OSAL
<i>OS_ERROR</i>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

## 10.67 OSAL Heap APIs

### Functions

- `int32 OS_HeapGetInfo (OS_heap_prop_t *heap_prop)`

*Return current info on the heap.*

#### 10.67.1 Detailed Description

#### 10.67.2 Function Documentation

##### 10.67.2.1 `OS_HeapGetInfo()` `int32 OS_HeapGetInfo (` `OS_heap_prop_t * heap_prop )`

Return current info on the heap.

#### Parameters

<code>out</code>	<code>heap_prop</code>	Storage buffer for heap info
------------------	------------------------	------------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if the heap_prop argument is NULL

## 10.68 OSAL Object Type Defines

### Macros

- `#define OS_OBJECT_TYPE_UNDEFINED 0x00`  
*Object type undefined.*
- `#define OS_OBJECT_TYPE_OS_TASK 0x01`  
*Object task type.*
- `#define OS_OBJECT_TYPE_OS_QUEUE 0x02`  
*Object queue type.*
- `#define OS_OBJECT_TYPE_OS_COUNTSEM 0x03`  
*Object counting semaphore type.*
- `#define OS_OBJECT_TYPE_OS_BINSEM 0x04`  
*Object binary semaphore type.*
- `#define OS_OBJECT_TYPE_OS_MUTEX 0x05`  
*Object mutex type.*
- `#define OS_OBJECT_TYPE_OS_STREAM 0x06`  
*Object stream type.*
- `#define OS_OBJECT_TYPE_OS_DIR 0x07`  
*Object directory type.*
- `#define OS_OBJECT_TYPE_OS_TIMEBASE 0x08`  
*Object timebase type.*
- `#define OS_OBJECT_TYPE_OS_TIMECB 0x09`  
*Object timer callback type.*
- `#define OS_OBJECT_TYPE_OS_MODULE 0x0A`  
*Object module type.*
- `#define OS_OBJECT_TYPE_OS_FILESYS 0x0B`  
*Object file system type.*
- `#define OS_OBJECT_TYPE_OS_CONSOLE 0x0C`  
*Object console type.*
- `#define OS_OBJECT_TYPE_OS_CONDVAR 0x0D`  
*Object condition variable type.*
- `#define OS_OBJECT_TYPE_USER 0x10`  
*Object user type.*

### 10.68.1 Detailed Description

### 10.68.2 Macro Definition Documentation

**10.68.2.1 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.

**10.68.2.2 OS\_OBJECT\_TYPE\_OS\_CONDVAR** `#define OS_OBJECT_TYPE_OS_CONDVAR 0x0D`  
Object condition variable type.  
Definition at line 51 of file osapi-idmap.h.

**10.68.2.3 OS\_OBJECT\_TYPE\_OS\_CONSOLE** #define OS\_OBJECT\_TYPE\_OS\_CONSOLE 0x0C  
Object console type.  
Definition at line 50 of file osapi-idmap.h.

**10.68.2.4 OS\_OBJECT\_TYPE\_OS\_COUNTSEM** #define OS\_OBJECT\_TYPE\_OS\_COUNTSEM 0x03  
Object counting semaphore type.  
Definition at line 41 of file osapi-idmap.h.

**10.68.2.5 OS\_OBJECT\_TYPE\_OS\_DIR** #define OS\_OBJECT\_TYPE\_OS\_DIR 0x07  
Object directory type.  
Definition at line 45 of file osapi-idmap.h.

**10.68.2.6 OS\_OBJECT\_TYPE\_OS\_FILESYS** #define OS\_OBJECT\_TYPE\_OS\_FILESYS 0x0B  
Object file system type.  
Definition at line 49 of file osapi-idmap.h.

**10.68.2.7 OS\_OBJECT\_TYPE\_OS\_MODULE** #define OS\_OBJECT\_TYPE\_OS\_MODULE 0x0A  
Object module type.  
Definition at line 48 of file osapi-idmap.h.

**10.68.2.8 OS\_OBJECT\_TYPE\_OS\_MUTEX** #define OS\_OBJECT\_TYPE\_OS\_MUTEX 0x05  
Object mutex type.  
Definition at line 43 of file osapi-idmap.h.

**10.68.2.9 OS\_OBJECT\_TYPE\_OS\_QUEUE** #define OS\_OBJECT\_TYPE\_OS\_QUEUE 0x02  
Object queue type.  
Definition at line 40 of file osapi-idmap.h.

**10.68.2.10 OS\_OBJECT\_TYPE\_OS\_STREAM** #define OS\_OBJECT\_TYPE\_OS\_STREAM 0x06  
Object stream type.  
Definition at line 44 of file osapi-idmap.h.

**10.68.2.11 OS\_OBJECT\_TYPE\_OS\_TASK** #define OS\_OBJECT\_TYPE\_OS\_TASK 0x01  
Object task type.  
Definition at line 39 of file osapi-idmap.h.

**10.68.2.12 OS\_OBJECT\_TYPE\_OS\_TIMEBASE** #define OS\_OBJECT\_TYPE\_OS\_TIMEBASE 0x08  
Object timebase type.  
Definition at line 46 of file osapi-idmap.h.

**10.68.2.13 OS\_OBJECT\_TYPE\_OS\_TIMECB** #define OS\_OBJECT\_TYPE\_OS\_TIMECB 0x09  
Object timer callback type.  
Definition at line 47 of file osapi-idmap.h.

**10.68.2.14 OS\_OBJECT\_TYPE\_UNDEFINED** #define OS\_OBJECT\_TYPE\_UNDEFINED 0x00  
Object type undefined.  
Definition at line 38 of file osapi-idmap.h.

**10.68.2.15 OS\_OBJECT\_TYPE\_USER** #define OS\_OBJECT\_TYPE\_USER 0x10  
Object user type.  
Definition at line 52 of file osapi-idmap.h.

## 10.69 OSAL Object ID Utility APIs

### Functions

- static unsigned long [OS\\_ObjectIdToInteger](#) ([osal\\_id\\_t](#) object\_id)  
*Obtain an integer value corresponding to an object ID.*
- static [osal\\_id\\_t OS\\_ObjectIdFromInteger](#) (unsigned long value)  
*Obtain an osal ID corresponding to an integer value.*
- static bool [OS\\_ObjectIdEqual](#) ([osal\\_id\\_t](#) object\_id1, [osal\\_id\\_t](#) object\_id2)  
*Check two OSAL object ID values for equality.*
- static bool [OS\\_ObjectIdDefined](#) ([osal\\_id\\_t](#) object\_id)  
*Check if an object ID is defined.*
- int32 [OS\\_GetResourceName](#) ([osal\\_id\\_t](#) object\_id, char \*buffer, size\_t buffer\_size)  
*Obtain the name of an object given an arbitrary object ID.*
- [osal\\_objtype\\_t OS\\_IdentifyObject](#) ([osal\\_id\\_t](#) object\_id)  
*Obtain the type of an object given an arbitrary object ID.*
- int32 [OS\\_ConvertToArrayIndex](#) ([osal\\_id\\_t](#) object\_id, [osal\\_index\\_t](#) \*ArrayIndex)  
*Converts an abstract ID into a number suitable for use as an array index.*
- int32 [OS\\_ObjectIdToArrayIndex](#) ([osal\\_objtype\\_t](#) idtype, [osal\\_id\\_t](#) object\_id, [osal\\_index\\_t](#) \*ArrayIndex)  
*Converts an abstract ID into a number suitable for use as an array index.*
- void [OS\\_ForEachObject](#) ([osal\\_id\\_t](#) creator\_id, [OS\\_ArgCallback\\_t](#) callback\_ptr, void \*callback\_arg)  
*call the supplied callback function for all valid object IDs*
- void [OS\\_ForEachObjectType](#) ([osal\\_objtype\\_t](#) objtype, [osal\\_id\\_t](#) creator\_id, [OS\\_ArgCallback\\_t](#) callback\_ptr, void \*callback\_arg)  
*call the supplied callback function for valid object IDs of a specific type*

### 10.69.1 Detailed Description

### 10.69.2 Function Documentation

**10.69.2.1 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	<a href="#">object_id</a>	The object ID to operate on
out	<a href="#">*ArrayIndex</a>	The Index to return (must not be null)

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the object_id argument is not valid
<code>OS_INVALID_POINTER</code>	if the ArrayIndex is NULL

**10.69.2.2 OS\_ForEachObject()** `void OS_ForEachObject (`  
    `osal_id_t creator_id,`  
    `OS_ArgCallback_t callback_ptr,`  
    `void * callback_arg )`

call the supplied callback function for all valid object IDs

Loops through all defined OSAL objects of all types and calls callback\_ptr on each one If creator\_id is nonzero then only objects with matching creator id are processed.

**Parameters**

in	<i>creator_id</i>	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	<i>callback_ptr</i>	Function to invoke for each matching object ID
in	<i>callback_arg</i>	Opaque Argument to pass to callback function (may be NULL)

**10.69.2.3 OS\_ForEachObjectType()** `void OS_ForEachObjectType (`  
    `osal_objtype_t objtype,`  
    `osal_id_t creator_id,`  
    `OS_ArgCallback_t callback_ptr,`  
    `void * callback_arg )`

call the supplied callback function for valid object IDs of a specific type

Loops through all defined OSAL objects of a specific type and calls callback\_ptr on each one If creator\_id is nonzero then only objects with matching creator id are processed.

**Parameters**

in	<i>objtype</i>	The type of objects to iterate
in	<i>creator_id</i>	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	<i>callback_ptr</i>	Function to invoke for each matching object ID
in	<i>callback_arg</i>	Opaque Argument to pass to callback function (may be NULL)

**10.69.2.4 OS\_GetResourceName()** `int32 OS_GetResourceName (`  
    `osal_id_t object_id,`  
    `char * buffer,`

```
size_t buffer_size )
```

Obtain the name of an object given an arbitrary object ID.

All OSAL resources generally have a name associated with them. This allows application code to retrieve the name of any valid OSAL object ID.

#### Parameters

in	<i>object_id</i>	The object ID to operate on
out	<i>buffer</i>	Buffer in which to store the name (must not be null)
in	<i>buffer_size</i>	Size of the output storage buffer (must not be zero)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the passed-in ID is not a valid OSAL ID
<i>OS_INVALID_POINTER</i>	if the passed-in buffer is invalid
<i>OS_ERR_NAME_TOO_LONG</i>	if the name will not fit in the buffer provided

#### 10.69.2.5 OS\_IdentifyObject()

```
osal_objtype_t OS_IdentifyObject (
    osal_id_t object_id )
```

Obtain the type of an object given an arbitrary object ID.

Given an arbitrary object ID, get the type of the object

#### Parameters

in	<i>object_id</i>	The object ID to operate on
----	------------------	-----------------------------

#### Returns

The object type portion of the object\_id, see [OSAL Object Type Defines](#) for expected values

#### 10.69.2.6 OS\_ObjectIdDefined()

```
static bool OS_ObjectIdDefined (
    osal_id_t object_id ) [inline], [static]
```

Check if an object ID is defined.

The OSAL ID values should be treated as abstract values by applications, and not directly manipulated using standard C operators.

This returns false if the ID is NOT a defined resource (i.e. free/empty/invalid).

#### Note

OS\_ObjectIdDefined(OS\_OBJECT\_ID\_UNDEFINED) is always guaranteed to be false.

**Parameters**

in	<i>object_id</i>	The first object ID
----	------------------	---------------------

Definition at line 150 of file osapi-idmap.h.

References OS\_ObjectIdToInteger().

**10.69.2.7 OS\_ObjectIdEqual()** static bool OS\_ObjectIdEqual (   
     osal\_id\_t object\_id1,  
     osal\_id\_t object\_id2 ) [inline], [static]

Check two OSAL object ID values for equality.

The OSAL ID values should be treated as abstract values by applications, and not directly manipulated using standard C operators.

This checks two values for equality, replacing the "==" operator.

**Parameters**

in	<i>object_id1</i>	The first object ID
in	<i>object_id2</i>	The second object ID

**Returns**

true if the object IDs are equal

Definition at line 129 of file osapi-idmap.h.

References OS\_ObjectIdToInteger().

**10.69.2.8 OS\_ObjectIdFromInteger()** static osal\_id\_t OS\_ObjectIdFromInteger (   
     unsigned long value ) [inline], [static]

Obtain an osal ID corresponding to an integer value.

Provides the inverse of [OS\\_ObjectIdToInteger\(\)](#). Reconstitutes the original osal\_id\_t type from an integer representation.

**Parameters**

in	<i>value</i>	The integer representation of an OSAL ID
----	--------------	--

**Returns**

The ID value converted to an osal\_id\_t

Definition at line 102 of file osapi-idmap.h.

**10.69.2.9 OS\_ObjectIdToArrayIndex()** int32 OS\_ObjectIdToArrayIndex (   
     osal\_objtype\_t idtype,  
     osal\_id\_t object\_id,  
     osal\_index\_t \* ArrayIndex )

Converts an abstract ID into a number suitable for use as an array index.

This will return a unique zero-based integer number in the range of [0,MAX) for any valid object ID. This may be used by application code as an array index for indexing into local tables.

This routine operates on a specific object type, and returns a value based on the maximum number of objects for that type.

If the idtype is passed as `OS_OBJECT_TYPE_UNDEFINED`, then object type verification is skipped and any object ID will be accepted and converted to an index. In this mode, the range of the output depends on the actual passed-in object type.

If the idtype is passed as any other value, the passed-in ID value is first confirmed to be the correct type. This check will guarantee that the output is within an expected range; for instance, if the type is passed as `OS_OBJECT_TYPE_OS_TASK`, then the output index is guaranteed to be between 0 and `OS_MAX_TASKS`-1 after successful conversion.

#### Parameters

in	<i>idtype</i>	The object type to convert
in	<i>object_id</i>	The object ID to operate on
out	<code>*ArrayIndex</code>	The Index to return (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the object_id argument is not valid
<code>OS_INVALID_POINTER</code>	if the ArrayIndex is NULL

**10.69.2.10 OS\_ObjectIdToInteger()** `static unsigned long OS_ObjectIdToInteger ( osal_id_t object_id ) [inline], [static]`

Obtain an integer value corresponding to an object ID.

Obtains an integer representation of an object id, generally for the purpose of printing to the console or system logs. The returned value is of the type "unsigned long" for direct use with printf-style functions. It is recommended to use the "%lx" conversion specifier as the hexadecimal encoding clearly delineates the internal fields.

#### Note

This provides the raw integer value and is *not* suitable for use as an array index, as the result is not zero-based. See the [OS\\_ConvertToArrayIndex\(\)](#) to obtain a zero-based index value.

#### Parameters

in	<i>object_id</i>	The object ID
----	------------------	---------------

#### Returns

integer value representation of object ID

Definition at line 80 of file osapi-idmap.h.

Referenced by `OS_ObjectIdDefined()`, and `OS_ObjectIdEqual()`.

## 10.70 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.*

#### 10.70.1 Detailed Description

#### 10.70.2 Function Documentation

**10.70.2.1 OS\_ModuleInfo()** `int32 OS_ModuleInfo (`  
    `osal_id_t module_id,`  
    `OS_module_prop_t * module_info )`

Obtain information about a module.

Returns information about the loadable module

#### Parameters

in	<code>module_id</code>	OSAL ID of the previously the loaded module
out	<code>module_info</code>	Buffer to store module information (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the module id invalid
<code>OS_INVALID_POINTER</code>	if the pointer to the ModuleInfo structure is invalid
<code>OS_ERROR</code>	if an other/unspecified error occurs (return value only verified in coverage test)

**10.70.2.2 OS\_ModuleLoad()** `int32 OS_ModuleLoad (`  
    `osal_id_t * module_id,`  
    `const char * module_name,`  
    `const char * filename,`

```
    uint32 flags )
```

Loads an object file.

Loads an object file into the running operating system

The "flags" parameter may influence how the loaded module symbols are made available for use in the application. See [OS\\_MODULE\\_FLAG\\_LOCAL\\_SYMBOLS](#) and [OS\\_MODULE\\_FLAG\\_GLOBAL\\_SYMBOLS](#) for descriptions.

#### Parameters

out	<i>module_id</i>	Non-zero OSAL ID corresponding to the loaded module
in	<i>module_name</i>	Name of module (must not be null)
in	<i>filename</i>	File containing the object code to load (must not be null)
in	<i>flags</i>	Options for the loaded module

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if one of the parameters is NULL
<a href="#">OS_ERR_NO_FREE_IDS</a>	if the module table is full
<a href="#">OS_ERR_NAME_TAKEN</a>	if the name is in use
<a href="#">OS_ERR_NAME_TOO_LONG</a>	if the module_name is too long
<a href="#">OS_FS_ERR_PATH_INVALID</a>	if the filename argument is not valid
<a href="#">OS_ERROR</a>	if an other/unspecified error occurs (return value only verified in coverage test)

### 10.70.2.3 OS\_ModuleSymbolLookup() [int32 OS\\_ModuleSymbolLookup \(](#)

```
    osal_id_t module_id,
    cpuaddr * symbol_address,
    const char * symbol_name )
```

Find the Address of a Symbol within a module.

This is similar to [OS\\_SymbolLookup\(\)](#) but for a specific module ID. This should be used to look up a symbol in a module that has been loaded with the [OS\\_MODULE\\_FLAG\\_LOCAL\\_SYMBOLS](#) flag.

#### Parameters

in	<i>module_id</i>	Module ID that should contain the symbol
out	<i>symbol_address</i>	Set to the address of the symbol (must not be null)
in	<i>symbol_name</i>	Name of the symbol to look up (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
----------------------------	-----------------------

**Return values**

<i>OS_ERROR</i>	if the symbol could not be found
<i>OS_INVALID_POINTER</i>	if one of the pointers passed in are NULL

**10.70.2.4 OS\_ModuleUnload()** `int32 OS_ModuleUnload ( osal_id_t module_id )`

Unloads the module file.

Unloads the module file from the running operating system

**Parameters**

in	<i>module_id</i>	OSAL ID of the previously the loaded module
----	------------------	---

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the module id invalid
<i>OS_ERROR</i>	if an other/unspecified error occurs (return value only verified in coverage test)

**10.70.2.5 OS\_SymbolLookup()** `int32 OS_SymbolLookup (`

`cpuaddr * symbol_address,`  
`const char * symbol_name )`

Find the Address of a Symbol.

This calls to the OS dynamic symbol lookup implementation, and/or checks a static symbol table for a matching symbol name.

The static table is intended to support embedded targets that do not have module loading capability or have it disabled.

**Parameters**

out	<i>symbol_address</i>	Set to the address of the symbol (must not be null)
in	<i>symbol_name</i>	Name of the symbol to look up (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the symbol could not be found
<i>OS_INVALID_POINTER</i>	if one of the pointers passed in are NULL

```
10.70.2.6 OS_SymbolTableDump() int32 OS_SymbolTableDump (
    const char * filename,
    size_t size_limit )
```

Dumps the system symbol table to a file.

Dumps the system symbol table to the specified filename

#### Note

Not all RTOS implementations support this API. If the underlying module subsystem does not provide a facility to iterate through the symbol table, then the [OS\\_ERR\\_NOT\\_IMPLEMENTED](#) status code is returned.

#### Parameters

in	<i>filename</i>	File to write to (must not be null)
in	<i>size_limit</i>	Maximum number of bytes to write

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_NOT_IMPLEMENTED</a>	Not implemented.
<a href="#">OS_INVALID_POINTER</a>	if the filename argument is NULL
<a href="#">OS_FS_ERR_PATH_INVALID</a>	if the filename argument is not valid
<a href="#">OS_ERR_NAME_TOO_LONG</a>	if any of the symbol names are too long (return value only verified in coverage test)
<a href="#">OS_ERR_OUTPUT_TOO_LARGE</a>	if the size_limit was reached before completing all symbols (return value only verified in coverage test)
<a href="#">OS_ERROR</a>	if an other/unspecified error occurs (return value only verified in coverage test)

## 10.71 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.*

### 10.71.1 Detailed Description

### 10.71.2 Function Documentation

#### 10.71.2.1 OS\_MutSemCreate() `int32 OS_MutSemCreate (`

```
    osal_id_t * sem_id,
    const char * sem_name,
    uint32 options )
```

Creates a mutex semaphore.

Mutex semaphores are always created in the unlocked (full) state.

#### Parameters

out	<code>sem_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<code>sem_name</code>	the name of the new resource to create (must not be null)
in	<code>options</code>	reserved for future use. Should be passed as 0.

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if sem_id or sem_name are NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NO_FREE_IDS</code>	if there are no more free mutex IDs
<code>OS_ERR_NAME_TAKEN</code>	if there is already a mutex with the same name
<code>OS_SEM_FAILURE</code>	if the OS call failed (return value only verified in coverage test)

**10.71.2.2 OS\_MutSemDelete()** `int32 OS_MutSemDelete ( osal_id_t sem_id )`

Deletes the specified Mutex Semaphore.

Delete the semaphore. This also frees the respective sem\_id such that it can be used again when another is created.

#### Parameters

in	<code>sem_id</code>	The object ID to delete
----	---------------------	-------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid mutex
<code>OS_SEM_FAILURE</code>	if an unspecified error occurs (return value only verified in coverage test)

**10.71.2.3 OS\_MutSemGetIdByName()** `int32 OS_MutSemGetIdByName (`

```
osal_id_t * sem_id,
const char * sem_name )
```

Find an existing mutex ID by name.

This function tries to find a mutex sem Id given the name of a mut\_sem. The id is returned through sem\_id

#### Parameters

out	<code>sem_id</code>	will be set to the ID of the existing resource
in	<code>sem_name</code>	the name of the existing resource to find (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	is semid or sem_name are NULL pointers
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name was not found in the table

**10.71.2.4 OS\_MutSemGetInfo()** `int32 OS_MutSemGetInfo (`

```
osal_id_t sem_id,
OS_mut_sem_prop_t * mut_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info( name and creator) about the specified mutex semaphore.

#### Parameters

in	<i>sem_id</i>	The object ID to operate on
out	<i>mut_prop</i>	The property object buffer to fill (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the mut_prop pointer is null

### 10.71.2.5 OS\_MutSemGive() `int32 OS_MutSemGive ( osal_id_t sem_id )`

Releases the mutex object referenced by *sem\_id*.

If there are threads blocked on the mutex object referenced by mutex when this function is called, resulting in the mutex becoming available, the scheduling policy shall determine which thread shall acquire the mutex.

#### Parameters

in	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid mutex
<i>OS_SEM_FAILURE</i>	if an unspecified error occurs (return value only verified in coverage test)

### 10.71.2.6 OS\_MutSemTake() `int32 OS_MutSemTake ( osal_id_t sem_id )`

Acquire the mutex object referenced by *sem\_id*.

If the mutex is already locked, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by mutex in the locked state with the calling thread as its owner.

**Parameters**

in	<i>sem-&gt;_id</i>	The object ID to operate on
----	--------------------	-----------------------------

**Returns**

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

**Return values**

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_ERR_INVALID_ID</i></a>	the id passed in is not a valid mutex
<a href="#"><i>OS_SEM_FAILURE</i></a>	if an unspecified error occurs (return value only verified in coverage test)

## 10.72 OSAL Network ID APIs

Provides some basic methods to query a network host name and ID.

### Functions

- `int32 OS_NetworkGetID (void)`  
*Gets the network ID of the local machine.*
- `int32 OS_NetworkGetHostName (char *host_name, size_t name_len)`  
*Gets the local machine network host name.*

#### 10.72.1 Detailed Description

Provides some basic methods to query a network host name and ID.

#### 10.72.2 Function Documentation

**10.72.2.1 OS\_NetworkGetHostName()** `int32 OS_NetworkGetHostName (`  
    `char * host_name,`  
    `size_t name_len )`

Gets the local machine network host name.

If configured in the underlying network stack, this function retrieves the local hostname of the system.

##### Parameters

<code>out</code>	<code>host_name</code>	Buffer to hold name information (must not be null)
<code>in</code>	<code>name_len</code>	Maximum length of host name buffer (must not be zero)

##### Returns

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

##### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_SIZE</code>	if the name_len is zero
<code>OS_INVALID_POINTER</code>	if the host_name is NULL

**10.72.2.2 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.

## 10.73 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.*

#### 10.73.1 Detailed Description

#### 10.73.2 Function Documentation

**10.73.2.1 OS\_printf()** void OS\_printf (  
    const char \* string,  
    ... )

Abstraction for the system printf() call.

This function abstracts out the printf type statements. This is useful for using OS- specific thots that will allow non-polled print statements for the real time systems.

Operates in a manner similar to the printf() call defined by the standard C library and takes all the parameters and formatting options of printf. This abstraction may implement additional buffering, if necessary, to improve the real-time performance of the call.

Strings (including terminator) longer than [OS\\_BUFFER\\_SIZE](#) will be truncated.

The output of this routine also may be dynamically enabled or disabled by the [OS\\_printf\\_enable\(\)](#) and [OS\\_printf\\_disable\(\)](#) calls, respectively.

#### Parameters

in	<i>string</i>	Format string, followed by additional arguments
----	---------------	---

**10.73.2.2 OS\_printf\_disable()** void void OS\_printf\_disable (  
    void )

This function disables the output from OS\_printf.

**10.73.2.3 OS\_printf\_enable()** void OS\_printf\_enable (  
    void )

This function enables the output from OS\_printf.

## 10.74 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.*

#### 10.74.1 Detailed Description

#### 10.74.2 Function Documentation

**10.74.2.1 OS\_QueueCreate()** `int32 OS_QueueCreate (`  
`osal_id_t * queue_id,`  
`const char * queue_name,`  
`osal_blockcount_t queue_depth,`  
`size_t data_size,`  
`uint32 flags )`

Create a message queue.

This is the function used to create a queue in the operating system. Depending on the underlying operating system, the memory for the queue will be allocated automatically or allocated by the code that sets up the queue. Queue names must be unique; if the name already exists this function fails. Names cannot be NULL.

#### Parameters

<code>out</code>	<code>queue_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>queue_name</code>	the name of the new resource to create (must not be null)
<code>in</code>	<code>queue_depth</code>	the maximum depth of the queue
<code>in</code>	<code>data_size</code>	the size of each entry in the queue (must not be zero)
<code>in</code>	<code>flags</code>	options for the queue (reserved for future use, pass as 0)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if a pointer passed in is NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>

**Return values**

<i>OS_ERR_NO_FREE_IDS</i>	if there are already the max queues created
<i>OS_ERR_NAME_TAKEN</i>	if the name is already being used on another queue
<i>OS_ERR_INVALID_SIZE</i>	if data_size is 0
<i>OS_QUEUE_INVALID_SIZE</i>	if the queue depth exceeds the limit
<i>OS_ERROR</i>	if the OS create call fails

**10.74.2.2 OS\_QueueDelete()** `int32 OS_QueueDelete (``osal_id_t queue_id )`

Deletes the specified message queue.

This is the function used to delete a queue in the operating system. This also frees the respective queue\_id to be used again when another queue is created.

**Note**

If There are messages on the queue, they will be lost and any subsequent calls to QueueGet or QueuePut to this queue will result in errors

**Parameters**

in	<i>queue_id</i>	The object ID to delete
----	-----------------	-------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in does not exist
<i>OS_ERROR</i>	if the OS call returns an unexpected error (return value only verified in coverage test)

**10.74.2.3 OS\_QueueGet()** `int32 OS_QueueGet (``osal_id_t queue_id,
    void * data,
    size_t size,
    size_t * size_copied,
    int32 timeout )`

Receive a message on a message queue.

If a message is pending, it is returned immediately. Otherwise the calling task will block until a message arrives or the timeout expires.

**Parameters**

in	<i>queue_id</i>	The object ID to operate on
----	-----------------	-----------------------------

**Parameters**

<i>out</i>	<i>data</i>	The buffer to store the received message (must not be null)
<i>in</i>	<i>size</i>	The size of the data buffer (must not be zero)
<i>out</i>	<i>size_copied</i>	Set to the actual size of the message (must not be null)
<i>in</i>	<i>timeout</i>	The maximum amount of time to block, or OS_PEND to wait forever

**Returns**

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

**Return values**

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_ERR_INVALID_ID</i></a>	if the given ID does not exist
<a href="#"><i>OS_INVALID_POINTER</i></a>	if a pointer passed in is NULL
<a href="#"><i>OS_QUEUE_EMPTY</i></a>	if the Queue has no messages on it to be received
<a href="#"><i>OS_QUEUE_TIMEOUT</i></a>	if the timeout was OS_PEND and the time expired
<a href="#"><i>OS_QUEUE_INVALID_SIZE</i></a>	if the size copied from the queue was not correct
<a href="#"><i>OS_ERROR</i></a>	if the OS call returns an unexpected error (return value only verified in coverage test)

**10.74.2.4 OS\_QueueGetIdByName()** `int32 OS_QueueGetIdByName (`

```
    osal_id_t * queue_id,
    const char * queue_name )
```

Find an existing queue ID by name.

This function tries to find a queue Id given the name of the queue. The id of the queue is passed back in `queue_id`.

**Parameters**

<i>out</i>	<i>queue_id</i>	will be set to the ID of the existing resource
<i>in</i>	<i>queue_name</i>	the name of the existing resource to find (must not be null)

**Returns**

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

**Return values**

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_INVALID_POINTER</i></a>	if the name or id pointers are NULL
<a href="#"><i>OS_ERR_NAME_TOO_LONG</i></a>	name length including null terminator greater than <a href="#"><i>OS_MAX_API_NAME</i></a>
<a href="#"><i>OS_ERR_NAME_NOT_FOUND</i></a>	the name was not found in the table

**10.74.2.5 OS\_QueueGetInfo()** `int32 OS_QueueGetInfo (`

```
    osal_id_t queue_id,
    OS_QUEUE_PROP_T * queue_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info (name and creator) about the specified queue.

#### Parameters

in	<i>queue_id</i>	The object ID to operate on
out	<i>queue_prop</i>	The property object buffer to fill (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if queue_prop is NULL
<i>OS_ERR_INVALID_ID</i>	if the ID given is not a valid queue

### 10.74.2.6 OS\_QueuePut() `int32 OS_QueuePut(`

```
    osal_id_t queue_id,
    const void * data,
    size_t size,
    uint32 flags )
```

Put a message on a message queue.

#### Parameters

in	<i>queue_id</i>	The object ID to operate on
in	<i>data</i>	The buffer containing the message to put (must not be null)
in	<i>size</i>	The size of the data buffer (must not be zero)
in	<i>flags</i>	Currently reserved/unused, should be passed as 0

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the queue id passed in is not a valid queue
<i>OS_INVALID_POINTER</i>	if the data pointer is NULL
<i>OS_QUEUE_INVALID_SIZE</i>	if the data message is too large for the queue
<i>OS_QUEUE_FULL</i>	if the queue cannot accept another message
<i>OS_ERROR</i>	if the OS call returns an unexpected error (return value only verified in coverage test)

## 10.75 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.*

#### 10.75.1 Detailed Description

#### 10.75.2 Function Documentation

##### 10.75.2.1 OS\_SelectFdAdd() `int32 OS_SelectFdAdd (`

```
    OS_FdSet * Set,
    osal_id_t objid )
```

Add an ID to an FdSet structure.

After this call the set will contain the given OSAL ID

#### Parameters

in, out	<code>Set</code>	Pointer to <code>OS_FdSet</code> object to operate on (must not be null)
in	<code>objid</code>	The handle ID to add to the set

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if argument is NULL
<code>OS_ERR_INVALID_ID</code>	if the objid is not a valid handle

##### 10.75.2.2 OS\_SelectFdClear() `int32 OS_SelectFdClear (`

```
    OS_FdSet * Set,
    osal_id_t objid )
```

Clear an ID from an FdSet structure.

After this call the set will no longer contain the given OSAL ID

**Parameters**

in, out	<i>Set</i>	Pointer to <a href="#">OS_FdSet</a> object to operate on (must not be null)
in	<i>objid</i>	The handle ID to remove from the set

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if argument is NULL
<a href="#">OS_ERR_INVALID_ID</a>	if the objid is not a valid handle

**10.75.2.3 OS\_SelectFdIsSet()** `bool OS_SelectFdIsSet (`  
    `const OS_FdSet * Set,`  
    `osal_id_t objid )`

Check if an FdSet structure contains a given ID.

**Parameters**

in	<i>Set</i>	Pointer to <a href="#">OS_FdSet</a> object to operate on (must not be null)
in	<i>objid</i>	The handle ID to check for in the set

**Returns**

Boolean set status

**Return values**

<i>true</i>	FdSet structure contains ID
<i>false</i>	FdSet structure does not contain ID

**10.75.2.4 OS\_SelectFdZero()** `int32 OS_SelectFdZero (`  
    `OS_FdSet * Set )`

Clear a FdSet structure.

After this call the set will contain no OSAL IDs

**Parameters**

out	<i>Set</i>	Pointer to <a href="#">OS_FdSet</a> object to clear (must not be null)
-----	------------	--

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL

**10.75.2.5 OS\_SelectMultiple()** *int32* *OS\_SelectMultiple* (

```
    OS_FdSet * ReadSet,
    OS_FdSet * WriteSet,
    int32 msecs )
```

Wait for events across multiple file handles.

Wait for any of the given sets of IDs to become readable or writable

This function will block until any of the following occurs:

- At least one OSAL ID in the ReadSet is readable
- At least one OSAL ID in the WriteSet is writable
- The timeout has elapsed

The sets are input/output parameters. On entry, these indicate the file handle(s) to wait for. On exit, these are set to the actual file handle(s) that have activity.

If the timeout occurs this returns an error code and all output sets should be empty.

**Note**

This does not lock or otherwise protect the file handles in the given sets. If a filehandle supplied via one of the FdSet arguments is closed or modified by another while this function is in progress, the results are undefined. Because of this limitation, it is recommended to use [OS\\_SelectSingle\(\)](#) whenever possible.

**Parameters**

<i>in, out</i>	<i>ReadSet</i>	Set of handles to check/wait to become readable
<i>in, out</i>	<i>WriteSet</i>	Set of handles to check/wait to become writable
<i>in</i>	<i>msecs</i>	Indicates the timeout. Positive values will wait up to that many milliseconds. Zero will not wait (poll). Negative values will wait forever (pend)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	If any handle in the ReadSet or WriteSet is readable or writable, respectively
<i>OS_ERROR_TIMEOUT</i>	If no handles in the ReadSet or WriteSet became readable or writable within the timeout
<i>OS_ERR_OPERATION_NOT_SUPPORTED</i>	if a specified handle does not support select
<i>OS_ERR_INVALID_ID</i>	if no valid handles were contained in the ReadSet/WriteSet

```
10.75.2.6 OS_SelectSingle() int32 OS_SelectSingle (
    osal_id_t objid,
    uint32 * StateFlags,
    int32 msecs )
```

Wait for events on a single file handle.

Wait for a single OSAL filehandle to change state

This function can be used to wait for a single OSAL stream ID to become readable or writable. On entry, the "StateFlags" parameter should be set to the desired state (OS\_STREAM\_STATE\_READABLE and/or OS\_STREAM\_STATE\_WRITABLE) and upon return the flags will be set to the state actually detected.

As this operates on a single ID, the filehandle is protected during this call, such that another thread accessing the same handle will return an error. However, it is important to note that once the call returns then other threads may then also read/write and affect the state before the current thread can service it.

To mitigate this risk the application may prefer to use the OS\_TimedRead/OS\_TimedWrite calls.

#### Parameters

in	<i>objid</i>	The handle ID to select on
in, out	<i>StateFlags</i>	State flag(s) (readable or writable) (must not be null)
in	<i>msecs</i>	Indicates the timeout. Positive values will wait up to that many milliseconds. Zero will not wait (poll). Negative values will wait forever (pend)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	If the handle is readable and/or writable, as requested
<a href="#">OS_ERROR_TIMEOUT</a>	If the handle did not become readable or writable within the timeout
<a href="#">OS_INVALID_POINTER</a>	if argument is NULL
<a href="#">OS_ERR_INVALID_ID</a>	if the objid is not a valid handle

## 10.76 OSAL Shell APIs

### Functions

- int32 `OS_ShellOutputToFile` (const char \*Cmd, `osal_id_t` filedes)

*Executes the command and sends output to a file.*

#### 10.76.1 Detailed Description

#### 10.76.2 Function Documentation

##### 10.76.2.1 `OS_ShellOutputToFile()`

```
int32 OS_ShellOutputToFile (
    const char * Cmd,
    osal_id_t filedes )
```

Executes the command and sends output to a file.

Takes a shell command in and writes the output of that command to the specified file. The output file must be opened previously with write access (OS\_WRITE\_ONLY or OS\_READ\_WRITE).

#### Parameters

in	<i>Cmd</i>	Command to pass to shell (must not be null)
in	<i>filedes</i>	File to send output to.

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERROR</code>	if the command was not executed properly
<code>OS_INVALID_POINTER</code>	if Cmd argument is NULL
<code>OS_ERR_INVALID_ID</code>	if the file descriptor passed in is invalid

## 10.77 OSAL Socket Address APIs

These functions provide a means to manipulate network addresses in a manner that is (mostly) agnostic to the actual network address type.

### Functions

- `int32 OS_SocketAddrInit (OS_SockAddr_t *Addr, OS_SocketDomain_t Domain)`  
*Initialize a socket address structure to hold an address of the given family.*
- `int32 OS_SocketAddrToString (char *buffer, size_t buflen, const OS_SockAddr_t *Addr)`  
*Get a string representation of a network host address.*
- `int32 OS_SocketAddrFromString (OS_SockAddr_t *Addr, const char *string)`  
*Set a network host address from a string representation.*
- `int32 OS_SocketAddrGetPort (uint16 *PortNum, const OS_SockAddr_t *Addr)`  
*Get the port number of a network address.*
- `int32 OS_SocketAddrSetPort (OS_SockAddr_t *Addr, uint16 PortNum)`  
*Set the port number of a network address.*

### 10.77.1 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.

### 10.77.2 Function Documentation

**10.77.2.1 OS\_SocketAddrFromString()** `int32 OS_SocketAddrFromString (`  
    `OS_SockAddr_t * Addr,`  
    `const char * string )`

Set a network host address from a string representation.

The specific format of the output string depends on the address family.

The address structure should have been previously initialized using [OS\\_SocketAddrInit\(\)](#) to set the address family type.

#### Note

For IPv4, this would typically be the dotted-decimal format (X.X.X.X). It is up to the discretion of the underlying implementation whether to accept hostnames, as this depends on the availability of DNS services. Since many embedded deployments do not have name services, this should not be relied upon.

#### Parameters

<code>out</code>	<code>Addr</code>	The address buffer to initialize (must not be null)
<code>in</code>	<code>string</code>	The string to initialize the address from (must not be null)

#### Returns

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERROR</i>	if the string cannot be converted to an address

**10.77.2.2 OS\_SocketAddrGetPort()** `int32 OS_SocketAddrGetPort (`

```
    uint16 * PortNum,
    const OS_SockAddr_t * Addr )
```

Get the port number of a network address.

For network protocols that have the concept of a port number (such as TCP/IP and UDP/IP) this function gets the port number from the address structure.

**Parameters**

<i>out</i>	<i>PortNum</i>	Buffer to store the port number (must not be null)
<i>in</i>	<i>Addr</i>	The network address buffer (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_BAD_ADDRESS</i>	if the address domain is not compatible

**10.77.2.3 OS\_SocketAddrInit()** `int32 OS_SocketAddrInit (`

```
    OS_SockAddr_t * Addr,
    OS_SocketDomain_t Domain )
```

Initialize a socket address structure to hold an address of the given family.

The address is set to a suitable default value for the family.

**Parameters**

<i>out</i>	<i>Addr</i>	The address buffer to initialize (must not be null)
<i>in</i>	<i>Domain</i>	The address family

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
-------------------	-----------------------

**Return values**

<i>OS_INVALID_POINTER</i>	if Addr argument is NULL
<i>OS_ERR_NOT_IMPLEMENTED</i>	if the system does not implement the requested domain

**10.77.2.4 OS\_SocketAddrSetPort()** `int32 OS_SocketAddrSetPort (`

```
    OS_SockAddr_t * Addr,  
    uint16 PortNum )
```

Set the port number of a network address.

For network protocols that have the concept of a port number (such as TCP/IP and UDP/IP) this function sets the port number from the address structure.

**Parameters**

<i>out</i>	<i>Addr</i>	The network address buffer (must not be null)
<i>in</i>	<i>PortNum</i>	The port number to set

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_BAD_ADDRESS</i>	if the address domain is not compatible

**10.77.2.5 OS\_SocketAddrToString()** `int32 OS_SocketAddrToString (`

```
    char * buffer,  
    size_t bufLen,  
    const OS_SockAddr_t * Addr )
```

Get a string representation of a network host address.

The specific format of the output string depends on the address family.

This string should be suitable to pass back into [OS\\_SocketAddrFromString\(\)](#) which should recreate the same network address, and it should also be meaningful to a user of printed or logged as a C string.

**Note**

For IPv4, this would typically be the dotted-decimal format (X.X.X.X).

**Parameters**

<i>out</i>	<i>buffer</i>	Buffer to hold the output string (must not be null)
<i>in</i>	<i>bufLen</i>	Maximum length of the output string (must not be zero)
<i>in</i>	<i>Addr</i>	The network address buffer to convert (must not be null)

**Returns**

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

**Return values**

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_INVALID_POINTER</i></a>	if argument is NULL
<a href="#"><i>OS_ERR_INVALID_SIZE</i></a>	if passed-in buflen is not valid
<a href="#"><i>OS_ERROR</i></a>	if the address cannot be converted to string, or string buffer too small

## 10.78 OSAL Socket Management APIs

These functions are loosely related to the BSD Sockets API but made to be more consistent with other OSAL API functions. That is, they operate on OSAL IDs (32-bit opaque number values) and return an OSAL error code.

### Functions

- `int32 OS_SocketOpen (osal_id_t *sock_id, OS_SocketDomain_t Domain, OS_SocketType_t Type)`  
*Opens a socket.*
- `int32 OS_SocketBind (osal_id_t sock_id, const OS_SockAddr_t *Addr)`  
*Binds a socket to a given local address and enter listening (server) mode.*
- `int32 OS_SocketListen (osal_id_t sock_id)`  
*Places the specified socket into a listening state.*
- `int32 OS_SocketBindAddress (osal_id_t sock_id, const OS_SockAddr_t *Addr)`  
*Binds a socket to a given local address.*
- `int32 OS_SocketConnect (osal_id_t sock_id, const OS_SockAddr_t *Addr, int32 timeout)`  
*Connects a socket to a given remote address.*
- `int32 OS_SocketShutdown (osal_id_t sock_id, OS_SocketShutdownMode_t Mode)`  
*Implement graceful shutdown of a stream socket.*
- `int32 OS_SocketAccept (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, int32 timeout)`  
*Waits for and accept the next incoming connection on the given socket.*
- `int32 OS_SocketRecvFrom (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, int32 timeout)`  
*Reads data from a message-oriented (datagram) socket.*
- `int32 OS_SocketSendTo (osal_id_t sock_id, const void *buffer, size_t buflen, const OS_SockAddr_t *RemoteAddr)`  
*Sends data to a message-oriented (datagram) socket.*
- `int32 OS_SocketGetIdByName (osal_id_t *sock_id, const char *sock_name)`  
*Gets an OSAL ID from a given name.*
- `int32 OS_SocketGetInfo (osal_id_t sock_id, OS_socket_prop_t *sock_prop)`  
*Gets information about an OSAL Socket ID.*

### 10.78.1 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.

### 10.78.2 Function Documentation

```
10.78.2.1 OS_SocketAccept() int32 OS_SocketAccept (
    osal_id_t sock_id,
    osal_id_t * connsock_id,
    OS_SockAddr_t * Addr,
    int32 timeout )
```

Waits for and accept the next incoming connection on the given socket.

This is used for sockets operating in a "server" role. The socket must be a stream type (connection-oriented) and previously bound to a local address using [OS\\_SocketBind\(\)](#). This will block the caller up to the given timeout or until an incoming connection request occurs, whichever happens first.

The new stream connection is then returned to the caller and the original server socket ID can be reused for the next connection.

#### Parameters

in	<i>sock_id</i>	The server socket ID, previously bound using <a href="#">OS_SocketBind()</a>
out	<i>connsock_id</i>	The connection socket, a new ID that can be read/written (must not be null)
in	<i>Addr</i>	The remote address of the incoming connection (must not be null)
in	<i>timeout</i>	The maximum amount of time to wait, or OS_PEND to wait forever

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if argument is NULL
<a href="#">OS_ERR_INVALID_ID</a>	if the <i>sock_id</i> parameter is not valid
<a href="#">OS_ERR_INCORRECT_OBJ_TYPE</a>	if the handle is not a socket
<a href="#">OS_ERR_INCORRECT_OBJ_STATE</a>	if the socket is not bound or already connected

#### 10.78.2.2 OS\_SocketBind() `int32 OS_SocketBind(`

```
    osal_id_t sock_id,
    const OS_SockAddr_t * Addr )
```

Binds a socket to a given local address and enter listening (server) mode.

This is a convenience/compatibility routine to perform both [OS\\_SocketBindAddress\(\)](#) and [OS\\_SocketListen\(\)](#) operations in a single call, intended to simplify the setup for a server role.

If the socket is connectionless, then it only binds to the local address.

#### Parameters

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The local address to bind to (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the <i>sock_id</i> parameter is not valid

**Return values**

<a href="#">OS_INVALID_POINTER</a>	if argument is NULL
<a href="#">OS_ERR_INCORRECT_OBJ_STATE</a>	if the socket is already bound
<a href="#">OS_ERR_INCORRECT_OBJ_TYPE</a>	if the handle is not a socket

**10.78.2.3 OS\_SocketBindAddress()** `int32 OS_SocketBindAddress (`

```
    osal_id_t sock_id,  
    const OS_SockAddr_t * Addr )
```

Binds a socket to a given local address.

The specified socket will be bound to the local address and port, if available. This controls the source address reflected in network traffic transmitted via this socket.

After binding to the address, a stream socket may be followed by a call to either [OS\\_SocketListen\(\)](#) for a server role or to [OS\\_SocketConnect\(\)](#) for a client role.

**Parameters**

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The local address to bind to (must not be null)

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the <i>sock_id</i> parameter is not valid
<a href="#">OS_INVALID_POINTER</a>	if argument is NULL
<a href="#">OS_ERR_INCORRECT_OBJ_STATE</a>	if the socket is already bound
<a href="#">OS_ERR_INCORRECT_OBJ_TYPE</a>	if the handle is not a socket

**10.78.2.4 OS\_SocketConnect()** `int32 OS_SocketConnect (`

```
    osal_id_t sock_id,  
    const OS_SockAddr_t * Addr,  
    int32 timeout )
```

Connects a socket to a given remote address.

The socket will be connected to the remote address and port, if available. This only applies to stream-oriented sockets. Calling this on a datagram socket will return an error (these sockets should use SendTo/RecvFrom).

**Parameters**

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The remote address to connect to (must not be null)
in	<i>timeout</i>	The maximum amount of time to wait, or OS_PEND to wait forever

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already connected
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket
<i>OS_INVALID_POINTER</i>	if <i>Addr</i> argument is NULL

**10.78.2.5 OS\_SocketGetIdByName()** `int32 OS_SocketGetIdByName (`

```
    osal_id_t * sock_id,
    const char * sock_name )
```

Gets an OSAL ID from a given name.

**Note**

OSAL Sockets use generated names according to the address and type.

**See also**

[OS\\_SocketGetInfo\(\)](#)

**Parameters**

<i>out</i>	<i>sock_id</i>	Buffer to hold result (must not be null)
<i>in</i>	<i>sock_name</i>	Name of socket to find (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	is id or name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table

**10.78.2.6 OS\_SocketGetInfo()** `int32 OS_SocketGetInfo (`

```
    osal_id_t sock_id,
    OS_socket_prop_t * sock_prop )
```

Gets information about an OSAL Socket ID.

OSAL Sockets use generated names according to the address and type. This allows applications to find the name of a given socket.

**Parameters**

in	<i>sock_id</i>	The socket ID
out	<i>sock_prop</i>	Buffer to hold socket information (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the count_prop pointer is null

**10.78.2.7 OS\_SocketListen()** `int32 OS_SocketListen ( osal_id_t sock_id )`

Places the specified socket into a listening state.

This function only applies to connection-oriented (stream) sockets that are intended to be used in a server-side role. This places the socket into a state where it can accept incoming connections from clients.

**Parameters**

in	<i>sock_id</i>	The socket ID
----	----------------	---------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the sock_id parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already listening
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a stream socket

**10.78.2.8 OS\_SocketOpen()** `int32 OS_SocketOpen ( osal_id_t * sock_id, OS_SocketDomain_t Domain, OS_SocketType_t Type )`

Opens a socket.

A new, unconnected and unbound socket is allocated of the given domain and type.

**Parameters**

out	<i>sock_id</i>	Buffer to hold the non-zero OSAL ID (must not be null)
in	<i>Domain</i>	The domain / address family of the socket (INET or INET6, etc)
in	<i>Type</i>	The type of the socket (STREAM or DATAGRAM)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_NOT_IMPLEMENTED</i>	if the system does not implement the requested socket/address domain

**10.78.2.9 OS\_SocketRecvFrom()**

```
int32 OS_SocketRecvFrom (
    osal_id_t sock_id,
    void * buffer,
    size_t buflen,
    OS_SockAddr_t * RemoteAddr,
    int32 timeout )
```

Reads data from a message-oriented (datagram) socket.

If a message is already available on the socket, this should immediately return that data without blocking. Otherwise, it may block up to the given timeout.

**Parameters**

in	<i>sock_id</i>	The socket ID, previously bound using <a href="#">OS_SocketBind()</a>
out	<i>buffer</i>	Pointer to message data receive buffer (must not be null)
in	<i>buflen</i>	The maximum length of the message data to receive (must not be zero)
out	<i>RemoteAddr</i>	Buffer to store the remote network address (may be NULL)
in	<i>timeout</i>	The maximum amount of time to wait, or OS_PEND to wait forever

**Returns**

Count of actual bytes received or error status, see [OSAL Return Code Defines](#)

**Return values**

<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_SIZE</i>	if passed-in buflen is not valid
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

```
10.78.2.10 OS_SocketSendTo() int32 OS_SocketSendTo (
    osal_id_t sock_id,
    const void * buffer,
    size_t buflen,
    const OS_SockAddr_t * RemoteAddr )
```

Sends data to a message-oriented (datagram) socket.

This sends data in a non-blocking mode. If the socket is not currently able to queue the message, such as if its outbound buffer is full, then this returns an error code.

#### Parameters

in	<i>sock_id</i>	The socket ID, which must be of the datagram type
in	<i>buffer</i>	Pointer to message data to send (must not be null)
in	<i>buflen</i>	The length of the message data to send (must not be zero)
in	<i>RemoteAddr</i>	Buffer containing the remote network address to send to

#### Returns

Count of actual bytes sent or error status, see [OSAL Return Code Defines](#)

#### Return values

<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_SIZE</i>	if passed-in buflen is not valid
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

```
10.78.2.11 OS_SocketShutdown() int32 OS_SocketShutdown (
    osal_id_t sock_id,
    OS_SocketShutdownMode_t Mode )
```

Implement graceful shutdown of a stream socket.

This can be utilized to indicate the end of data stream without immediately closing the socket, giving the remote side an indication that the data transfer is complete.

#### Parameters

in	<i>sock_id</i>	The socket ID
in	<i>Mode</i>	Whether to shutdown reading, writing, or both.

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid

**Return values**

<i>OS_ERR_INVALID_ARGUMENT</i>	if the Mode argument is not one of the valid options
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is not connected

## 10.79 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.*

### 10.79.1 Detailed Description

### 10.79.2 Function Documentation

```
10.79.2.1 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	<code>task_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
-----	----------------------	---

**Parameters**

in	<i>task_name</i>	the name of the new resource to create (must not be null)
in	<i>function_pointer</i>	the entry point of the new task (must not be null)
in	<i>stack_pointer</i>	pointer to the stack for the task, or NULL to allocate a stack from the system memory heap
in	<i>stack_size</i>	the size of the stack (must not be zero)
in	<i>priority</i>	initial priority of the new task
in	<i>flags</i>	initial options for the new task

**Returns**Execution status, see [OSAL Return Code Defines](#)**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if any of the necessary pointers are NULL
<i>OS_ERR_INVALID_SIZE</i>	if the stack_size argument is zero
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_INVALID_PRIORITY</i>	if the priority is bad (return value only verified in coverage test)
<i>OS_ERR_NO_FREE_IDS</i>	if there can be no more tasks created
<i>OS_ERR_NAME_TAKEN</i>	if the name specified is already used by a task
<i>OS_ERROR</i>	if an unspecified/other error occurs (return value only verified in coverage test)

**10.79.2.2 OS\_TaskDelay()** `int32 OS_TaskDelay ( uint32 millisecond )`

Delay a task for specified amount of milliseconds.

Causes the current thread to be suspended from execution for the period of millisecond. This is a scheduled wait (clock\_nanosleep/rtems\_task\_wake\_after/taskDelay), not a "busy" wait.

**Parameters**

in	<i>millisecond</i>	Amount of time to delay
----	--------------------	-------------------------

**Returns**Execution status, see [OSAL Return Code Defines](#)**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if an unspecified/other error occurs (return value only verified in coverage test)

Referenced by CS\_OneShotChildTask(), CS\_RecomputeAppChildTask(), CS\_RecomputeEepromMemoryChildTask(), and CS\_RecomputeTablesChildTask().

**10.79.2.3 OS\_TaskDelete()** `int32 OS_TaskDelete (`  
    `osal_id_t task_id )`

Deletes the specified Task.

The task will be removed from the local tables. and the OS will be configured to stop executing the task at the next opportunity.

#### Parameters

in	<i>task_id</i>	The object ID to operate on
----	----------------	-----------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the ID given to it is invalid
<code>OS_ERROR</code>	if the OS delete call fails (return value only verified in coverage test)

**10.79.2.4 OS\_TaskExit()** `void OS_TaskExit (`  
    `void )`

Exits the calling task.

The calling thread is terminated. This function does not return.

**10.79.2.5 OS\_TaskFindIdBySystemData()** `int32 OS_TaskFindIdBySystemData (`  
    `osal_id_t * task_id,`  
    `const void * sysdata,`  
    `size_t sysdata_size )`

Reverse-lookup the OSAL task ID from an operating system ID.

This provides a method by which an external entity may find the OSAL task ID corresponding to a system-defined identifier (e.g. TASK\_ID, pthread\_t, rtems\_id, etc).

Normally OSAL does not expose the underlying OS-specific values to the application, but in some circumstances, such as exception handling, the OS may provide this information directly to a BSP handler outside of the normal OSAL API.

#### Parameters

out	<i>task_id</i>	The buffer where the task id output is stored (must not be null)
in	<i>sysdata</i>	Pointer to the system-provided identification data
in	<i>sysdata_size</i>	Size of the system-provided identification data

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution. (return value only verified in coverage test)
<code>OS_INVALID_POINTER</code>	if a pointer argument is NULL

```
10.79.2.6 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)

```
10.79.2.7 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

<code>out</code>	<code>task_id</code>	will be set to the ID of the existing resource
<code>in</code>	<code>task_name</code>	the name of the existing resource to find (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if the pointers passed in are NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name wasn't found in the table

```
10.79.2.8 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

<code>in</code>	<code>task_id</code>	The object ID to operate on
<code>out</code>	<code>task_prop</code>	The property object buffer to fill (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the ID passed to it is invalid
<i>OS_INVALID_POINTER</i>	if the task_prop pointer is NULL

**10.79.2.9 OS\_TaskInstallDeleteHandler()** `int32 OS_TaskInstallDeleteHandler ( osal_task_entry function_pointer )`

Installs a handler for when the task is deleted.

This function is used to install a callback that is called when the task is deleted. The callback is called when `OS_TaskDelete` is called with the task ID. A task delete handler is useful for cleaning up resources that a task creates, before the task is removed from the system.

**Parameters**

<i>in</i>	<i>function_pointer</i>	function to be called when task exits
-----------	-------------------------	---------------------------------------

**Returns**

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

**Return values**

<i>OS_ERR_INVALID_ID</i>	if the calling context is not an OSAL task
--------------------------	--

**10.79.2.10 OS\_TaskSetPriority()** `int32 OS_TaskSetPriority ( osal_id_t task_id, osal_priority_t new_priority )`

Sets the given task to a new priority.

**Parameters**

<i>in</i>	<i>task_id</i>	The object ID to operate on
<i>in</i>	<i>new_priority</i>	Set the new priority

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the ID passed to it is invalid

**Return values**

<i>OS_ERR_INVALID_PRIORITY</i>	if the priority is greater than the max allowed (return value only verified in coverage test)
<i>OS_ERROR</i>	if an unspecified/other error occurs (return value only verified in coverage test)

## 10.80 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.*

### 10.80.1 Detailed Description

### 10.80.2 Function Documentation

**10.80.2.1 OS\_TimeBaseCreate()** `int32 OS_TimeBaseCreate (`  
    `osal_id_t * timebase_id,`  
    `const char * timebase_name,`  
    `OS_TimerSync_t external_sync )`

Create an abstract Time Base resource.

An OSAL time base is an abstraction of a "timer tick" that can, in turn, be used for measurement of elapsed time between events.

Time bases can be simulated by the operating system using the OS kernel-provided timing facilities, or based on a hardware timing source if provided by the BSP.

A time base object has a servicing task associated with it, that runs at elevated priority and will thereby interrupt user-level tasks when timing ticks occur.

If the `external_sync` function is passed as NULL, the operating system kernel timing resources will be utilized for a simulated timer tick.

If the `external_sync` function is not NULL, this should point to a BSP-provided function that will block the calling task until the next tick occurs. This can be used for synchronizing with hardware events.

#### Note

When provisioning a tunable RTOS kernel, such as RTEMS, the kernel should be configured to support at least `(OS_MAX_TASKS + OS_MAX_TIMEBASES)` threads, to account for the helper threads associated with time base objects.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

#### Parameters

<code>out</code>	<code>timebase_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>timebase_name</code>	The name of the time base (must not be null)
<code>in</code>	<code>external_sync</code>	A synchronization function for BSP hardware-based timer ticks

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_NAME_TAKEN</i>	if the name specified is already used
<i>OS_ERR_NO_FREE_IDS</i>	if there can be no more timebase resources created
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context
<i>OS_ERR_NAME_TOO_LONG</i>	if the timebase_name is too long
<i>OS_INVALID_POINTER</i>	if a pointer argument is NULL

**10.80.2.2 OS\_TimeBaseDelete()** `int32 OS_TimeBaseDelete ( osal_id_t timebase_id )`

Deletes a time base object.

The helper task and any other resources associated with the time base abstraction will be freed.

**Note**

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**Parameters**

in	<i>timebase_id</i>	The timebase resource to delete
----	--------------------	---------------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

**10.80.2.3 OS\_TimeBaseGetFreeRun()** `int32 OS_TimeBaseGetFreeRun ( osal_id_t timebase_id, uint32 * freerun_val )`

Read the value of the timebase free run counter.

Poll the timer free-running time counter in a lightweight fashion.

The free run count is a monotonically increasing value reflecting the total time elapsed since the timebase inception. Units are the same as the timebase itself, usually microseconds.

Applications may quickly and efficiently calculate relative time differences by polling this value and subtracting the previous counter value.

The absolute value of this counter is not relevant, because it will "roll over" after  $2^{32}$  units of time. For a timebase with microsecond units, this occurs approximately every 4294 seconds, or about 1.2 hours.

#### Note

To ensure consistency of results, the application should sample the value at a minimum of two times the roll over frequency, and calculate the difference between the consecutive samples.

#### Parameters

in	<i>timebase_id</i>	The timebase to operate on
out	<i>freerun_val</i>	Buffer to store the free run counter (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_INVALID_POINTER</i>	if pointer argument is NULL

```
10.80.2.4 OS_TimeBaseGetIdByName() int32 OS_TimeBaseGetIdByName (
    osal_id_t * timebase_id,
    const char * timebase_name )
```

Find the ID of an existing time base resource.

Given a time base name, find and output the ID associated with it.

#### Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

#### Parameters

out	<i>timebase_id</i>	will be set to the non-zero ID of the matching resource (must not be null)
in	<i>timebase_name</i>	The name of the timebase resource to find (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if timebase_id or timebase_name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>

## Return values

<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.80.2.5 OS_TimeBaseGetInfo() int32 OS_TimeBaseGetInfo (
    osal_id_t timebase_id,
    OS_timebase_prop_t * timebase_prop )
```

Obtain information about a timebase resource.

Fills the buffer referred to by the timebase\_prop parameter with relevant information about the time base resource. This function will pass back a pointer to structure that contains all of the relevant info( name and creator) about the specified timebase.

## Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

## Parameters

in	<i>timebase_id</i>	The timebase resource ID
out	<i>timebase_prop</i>	Buffer to store timebase properties (must not be null)

## Returns

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_INVALID_POINTER</i>	if the timebase_prop pointer is null
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.80.2.6 OS_TimeBaseSet() int32 OS_TimeBaseSet (
    osal_id_t timebase_id,
    uint32 start_time,
    uint32 interval_time )
```

Sets the tick period for simulated time base objects.

This sets the actual tick period for timing ticks that are simulated by the RTOS kernel (i.e. the "external\_sync" parameter on the call to [OS\\_TimeBaseCreate\(\)](#) is NULL).

The RTOS will be configured to wake up the helper thread at the requested interval.

This function has no effect for time bases that are using a BSP-provided external\_sync function.

## Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**Parameters**

in	<i>timebase_id</i>	The timebase resource to configure
in	<i>start_time</i>	The amount of delay for the first tick, in microseconds.
in	<i>interval_time</i>	The amount of delay between ticks, in microseconds.

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timebase
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context
<i>OS_TIMER_ERR_INVALID_ARGS</i>	if start_time or interval_time are out of range

## 10.81 OSAL Timer APIs

### Functions

- `int32 OS_TimerCreate (osal_id_t *timer_id, const char *timer_name, uint32 *clock_accuracy, OS_TimerCallback_t callback_ptr)`  
*Create a timer object.*
- `int32 OS_TimerAdd (osal_id_t *timer_id, const char *timer_name, osal_id_t timebase_id, OS_ArgCallback_t callback_ptr, void *callback_arg)`  
*Add a timer object based on an existing TimeBase resource.*
- `int32 OS_TimerSet (osal_id_t timer_id, uint32 start_time, uint32 interval_time)`  
*Configures a periodic or one shot timer.*
- `int32 OS_TimerDelete (osal_id_t timer_id)`  
*Deletes a timer resource.*
- `int32 OS_TimerGetIdByName (osal_id_t *timer_id, const char *timer_name)`  
*Locate an existing timer resource by name.*
- `int32 OS_TimerGetInfo (osal_id_t timer_id, OS_timer_prop_t *timer_prop)`  
*Gets information about an existing timer.*

### 10.81.1 Detailed Description

### 10.81.2 Function Documentation

**10.81.2.1 OS\_TimerAdd()** `int32 OS_TimerAdd (`  
    `osal_id_t * timer_id,`  
    `const char * timer_name,`  
    `osal_id_t timebase_id,`  
    `OS_ArgCallback_t callback_ptr,`  
    `void * callback_arg )`

Add a timer object based on an existing TimeBase resource.

A timer object is a resource that invokes the specified application-provided function upon timer expiration. Timers may be one-shot or periodic in nature.

This function uses an existing time base object to service this timer, which must exist prior to adding the timer. The precision of the timer is the same as that of the underlying time base object. Multiple timer objects can be created referring to a single time base object.

This routine also uses a different callback function prototype from `OS_TimerCreate()`, allowing a single opaque argument to be passed to the callback routine. The OSAL implementation does not use this parameter, and may be set NULL.

The callback function for this method should be declared according to the `OS_ArgCallback_t` function pointer type. The `timer_id` is passed in to the function by the OSAL, and the `arg` parameter is passed through from the `callback_arg` argument on this call.

#### Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

#### See also

[OS\\_ArgCallback\\_t](#)

**Parameters**

<i>out</i>	<i>timer_id</i>	Will be set to the non-zero resource ID of the timer object (must not be null)
<i>in</i>	<i>timer_name</i>	Name of the timer object (must not be null)
<i>in</i>	<i>timebase_id</i>	The time base resource to use as a reference
<i>in</i>	<i>callback_ptr</i>	Application-provided function to invoke (must not be null)
<i>in</i>	<i>callback_arg</i>	Opaque argument to pass to callback function, may be NULL

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if any parameters are NULL
<a href="#">OS_ERR_INVALID_ID</a>	if the timebase_id parameter is not valid
<a href="#">OS_ERR_NAME_TOO_LONG</a>	name length including null terminator greater than <a href="#">OS_MAX_API_NAME</a>
<a href="#">OS_ERR_NAME_TAKEN</a>	if the name is already in use by another timer.
<a href="#">OS_ERR_NO_FREE_IDS</a>	if all of the timers are already allocated.
<a href="#">OS_ERR_INCORRECT_OBJ_STATE</a>	if invoked from a timer context
<a href="#">OS_TIMER_ERR_INTERNAL</a>	if there was an error programming the OS timer (return value only verified in coverage test)

```
10.81.2.2 OS_TimerCreate() int32 OS_TimerCreate (
    osal_id_t * timer_id,
    const char * timer_name,
    uint32 * clock_accuracy,
    OS_TimerCallback_t callback_ptr )
```

Create a timer object.

A timer object is a resource that invokes the specified application-provided function upon timer expiration. Timers may be one-shot or periodic in nature.

This function creates a dedicated (hidden) time base object to service this timer, which is created and deleted with the timer object itself. The internal time base is configured for an OS simulated timer tick at the same interval as the timer. The callback function should be declared according to the `OS_TimerCallback_t` function pointer type. The `timer_id` value is passed to the callback function.

**Note**

`clock_accuracy` comes from the underlying OS tick value. The nearest integer microsecond value is returned, so may not be exact.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**See also**

[OS\\_TimerCallback\\_t](#)

**Parameters**

out	<i>timer_id</i>	Will be set to the non-zero resource ID of the timer object (must not be null)
in	<i>timer_name</i>	Name of the timer object (must not be null)
out	<i>clock_accuracy</i>	Expected precision of the timer, in microseconds. This is the underlying tick value rounded to the nearest microsecond integer. (must not be null)
in	<i>callback_ptr</i>	The function pointer of the timer callback (must not be null).

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if any parameters are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NAME_TAKEN</i>	if the name is already in use by another timer.
<i>OS_ERR_NO_FREE_IDS</i>	if all of the timers are already allocated.
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if invoked from a timer context
<i>OS_TIMER_ERR_INTERNAL</i>	if there was an error programming the OS timer (return value only verified in coverage test)

**10.81.2.3 OS\_TimerDelete()** `int32 OS_TimerDelete ( osal_id_t timer_id )`

Deletes a timer resource.

The application callback associated with the timer will be stopped, and the resources freed for future use.

**Note**

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**Parameters**

in	<i>timer_id</i>	The timer ID to operate on
----	-----------------	----------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the timer_id is invalid.
<i>OS_TIMER_ERR_INTERNAL</i>	if there was a problem deleting the timer in the host OS (return value only verified in coverage test)
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.81.2.4 OS_TimerGetIdByName() int32 OS_TimerGetIdByName (
    osal_id_t * timer_id,
    const char * timer_name )
```

Locate an existing timer resource by name.

Outputs the ID associated with the given timer, if it exists.

#### Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

#### Parameters

out	<i>timer_id</i>	Will be set to the timer ID corresponding to the name (must not be null)
in	<i>timer_name</i>	The timer name to find (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if timer_id or timer_name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <a href="#">OS_MAX_API_NAME</a>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.81.2.5 OS_TimerGetInfo() int32 OS_TimerGetInfo (
```

```
    osal_id_t timer_id,
    OS_timer_prop_t * timer_prop )
```

Gets information about an existing timer.

This function takes timer\_id, and looks it up in the OS table. It puts all of the information known about that timer into a structure pointer to by timer\_prop.

## Parameters

---

### Note

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

## Parameters

in	<i>timer_id</i>	The timer ID to operate on
out	<i>timer_prop</i>	<p>Buffer containing timer properties (must not be null)</p> <ul style="list-style-type: none"> <li>• creator: the OS task ID of the task that created this timer</li> <li>• name: the string name of the timer</li> <li>• start_time: the start time in microseconds, if any</li> <li>• interval_time: the interval time in microseconds, if any</li> <li>• accuracy: the accuracy of the timer in microseconds</li> </ul>

## Returns

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

## Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the id passed in is not a valid timer
<a href="#">OS_INVALID_POINTER</a>	if the timer_prop pointer is null
<a href="#">OS_ERR_INCORRECT_OBJ_STATE</a>	if called from timer/timebase context

```
10.81.2.6 OS_TimerSet() int32 OS_TimerSet (
    osal_id_t timer_id,
    uint32 start_time,
    uint32 interval_time )
```

Configures a periodic or one shot timer.

This function programs the timer with a start time and an optional interval time. The start time is the time in microseconds when the user callback function will be called. If the interval time is non-zero, the timer will be reprogrammed with that interval in microseconds to call the user callback function periodically. If the start time and interval time are zero, the function will return an error.

For a "one-shot" timer, the start\_time configures the expiration time, and the interval\_time should be passed as zero to indicate the timer is not to be automatically reset.

### Note

The resolution of the times specified is limited to the clock accuracy returned in the OS\_TimerCreate call. If the times specified in the start\_msec or interval\_msec parameters are less than the accuracy, they will be rounded up to the accuracy of the timer.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**Parameters**

in	<i>timer_id</i>	The timer ID to operate on
in	<i>start_time</i>	Time in microseconds to the first expiration
in	<i>interval_time</i>	Time in microseconds between subsequent intervals, value of zero will only call the user callback function once after the start_msec time.

**Returns**

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

**Return values**

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_ERR_INVALID_ID</i></a>	if the timer_id is not valid.
<a href="#"><i>OS_TIMER_ERR_INTERNAL</i></a>	if there was an error programming the OS timer (return value only verified in coverage test)
<a href="#"><i>OS_ERR_INCORRECT_OBJ_STATE</i></a>	if called from timer/timebase context
<a href="#"><i>OS_TIMER_ERR_INVALID_ARGS</i></a>	if the start_time or interval_time is out of range, or both 0

## 11 Data Structure Documentation

### 11.1 CCSDS\_ExtendedHeader Struct Reference

CCSDS packet extended header.

```
#include <ccsds_hdr.h>
```

#### Data Fields

- **uint8 Subsystem [2]**  
*subsystem qualifier*
- **uint8 SystemId [2]**  
*system qualifier*

#### 11.1.1 Detailed Description

CCSDS packet extended header.

Definition at line 73 of file ccsds\_hdr.h.

#### 11.1.2 Field Documentation

##### 11.1.2.1 Subsystem `uint8 CCSDS_ExtendedHeader::Subsystem[2]`

subsystem qualifier

Definition at line 75 of file ccsds\_hdr.h.

##### 11.1.2.2 SystemId `uint8 CCSDS_ExtendedHeader::SystemId[2]`

system qualifier

Definition at line 82 of file ccsds\_hdr.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/msg/fsw/inc/ccsds_hdr.h`

### 11.2 CCSDS\_PrimaryHeader Struct Reference

CCSDS packet primary header.

```
#include <ccsds_hdr.h>
```

#### Data Fields

- **uint8 StreamId [2]**  
*packet identifier word (stream ID)*
- **uint8 Sequence [2]**  
*packet sequence word*
- **uint8 Length [2]**  
*packet length word*

#### 11.2.1 Detailed Description

CCSDS packet primary header.

Definition at line 51 of file ccsds\_hdr.h.

### 11.2.2 Field Documentation

#### 11.2.2.1 Length `uint8 CCSDS_PrimaryHeader::Length[2]`

packet length word

Definition at line 71 of file `ccsds_hdr.h`.

#### 11.2.2.2 Sequence `uint8 CCSDS_PrimaryHeader::Sequence[2]`

packet sequence word

Definition at line 66 of file `ccsds_hdr.h`.

#### 11.2.2.3 StreamId `uint8 CCSDS_PrimaryHeader::StreamId[2]`

packet identifier word (stream ID)

Definition at line 59 of file `ccsds_hdr.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/msg/fsw/inc/ccsds_hdr.h`

## 11.3 CFE\_ES\_AppInfo Struct Reference

Application Information.

```
#include <default_cfe_es_extern_typedefs.h>
```

### Data Fields

- `CFE_ResourceId_t ResourceId`  
*Application or Library ID for this resource.*
- `uint32 Type`  
*The type of App: CORE or EXTERNAL.*
- `char Name [CFE_MISSION_MAX_API_LEN]`  
*The Registered Name of the Application.*
- `char EntryPoint [CFE_MISSION_MAX_API_LEN]`  
*The Entry Point label for the Application.*
- `char FileName [CFE_MISSION_MAX_PATH_LEN]`  
*The Filename of the file containing the Application.*
- `CFE_ES_MemOffset_t StackSize`  
*The Stack Size of the Application.*
- `uint32 AddressesAreValid`  
*Indicates that the Code, Data, and BSS addresses/sizes are valid.*
- `CFE_ES_MemAddress_t CodeAddress`  
*The Address of the Application Code Segment.*
- `CFE_ES_MemOffset_t CodeSize`  
*The Code Size of the Application.*
- `CFE_ES_MemAddress_t DataAddress`  
*The Address of the Application Data Segment.*
- `CFE_ES_MemOffset_t DataSize`  
*The Data Size of the Application.*
- `CFE_ES_MemAddress_t BSSAddress`

- **CFE\_ES\_MemOffset\_t BSSSize**  
*The Address of the Application BSS Segment.*
- **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 441 of file default\_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 457 of file default\_cfe\_es\_extern\_typedefs.h.

Referenced by CS\_ComputeApp().

#### 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 467 of file default\_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 469 of file default\_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 459 of file default\_cfe\_es\_extern\_typedefs.h.

Referenced by `CS_ComputeApp()`.

**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 461 of file default\_cfe\_es\_extern\_typedefs.h.

Referenced by `CS_ComputeApp()`.

**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 463 of file default\_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 465 of file default\_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 450 of file default\_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_ExcepActn`

Definition at line 473 of file default\_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 480 of file default\_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 452 of file default\_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 478 of file default\_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 482 of file default\_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 448 of file default\_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 484 of file default\_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 476 of file default\_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 443 of file default\_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 455 of file default\_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 471 of file default\_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 445 of file default\_cfe\_es\_extern\_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_extern\_typedefs.h

## 11.4 CFE\_ES\_AppNameCmd Struct Reference

Generic application name command.

```
#include <default_cfe_es_msgstruct.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 192 of file default\_cfe\_es\_msgstruct.h.

### 11.4.2 Field Documentation

#### 11.4.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_ES_AppNameCmd::CommandHeader`

Command header.

Definition at line 194 of file default\_cfe\_es\_msgstruct.h.

**11.4.2.2 Payload** `CFE_ES_AppNameCmd_Payload_t` `CFE_ES_AppNameCmd::Payload`  
Command payload.

Definition at line 195 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.5 CFE\_ES\_AppNameCmd\_Payload Struct Reference

Generic application name command payload.

```
#include <default_cfe_es_msgstruct.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 184 of file default\_cfe\_es\_msgstruct.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 186 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.6 CFE\_ES\_AppReloadCmd\_Payload Struct Reference

Reload Application Command Payload.

```
#include <default_cfe_es_msgstruct.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 213 of file default\_cfe\_es\_msgstruct.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 216 of file default\_cfe\_es\_msgstruct.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 215 of file default\_cfe\_es\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.7 CFE\_ES\_BlockStats Struct Reference

Block statistics.

```
#include <default_cfe_es_extern_typedefs.h>
```

### Data Fields

- **CFE\_ES\_MemOffset\_t BlockSize**  
*Number of bytes in each of these blocks.*
- **uint32 NumCreated**  
*Number of Memory Blocks of this size created.*
- **uint32 NumFree**  
*Number of Memory Blocks of this size that are free.*

### 11.7.1 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 538 of file default\_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 540 of file default\_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 541 of file default\_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 542 of file default\_cfe\_es\_extern\_typedefs.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_extern\_typedefs.h

## 11.8 CFE\_ES\_CDSRegDumpRec Struct Reference

CDS Register Dump Record.

```
#include <default_cfe_es_extern_typedefs.h>
```

### Data Fields

- **CFE\_ES\_CDSHandle\_t Handle**  
*Handle of CDS.*
- **CFE\_ES\_MemOffset\_t Size**  
*Size, in bytes, of the CDS memory block.*
- **bool Table**  
*Flag that indicates whether CDS contains a Critical Table.*
- **char Name [CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN]**  
*Processor Unique Name of CDS.*
- **uint8 ByteAlignSpare [3]**  
*Spare bytes to ensure structure size is multiple of 4 bytes.*

### 11.8.1 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 523 of file default\_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 529 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.8.2.2 Handle `CFE_ES_CDSHandle_t CFE_ES_CDSRegDumpRec::Handle`

Handle of CDS.

Definition at line 525 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.8.2.3 Name `char CFE_ES_CDSRegDumpRec::Name [CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`

Processor Unique Name of CDS.

Definition at line 528 of file default\_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 526 of file default\_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 527 of file default\_cfe\_es\_extern\_typedefs.h.  
The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_extern_typedefs.h`

## 11.9 CFE\_ES\_DeleteCDSCmd Struct Reference

Delete Critical Data Store Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t** `CommandHeader`  
*Command header.*
- **CFE\_ES\_DeleteCDSCmd\_Payload\_t** `Payload`  
*Command payload.*

### 11.9.1 Detailed Description

Delete Critical Data Store Command.

Definition at line 265 of file default\_cfe\_es\_msgstruct.h.

### 11.9.2 Field Documentation

**11.9.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_DeleteCDSCmd::CommandHeader`  
Command header.  
Definition at line 267 of file default\_cfe\_es\_msgstruct.h.

**11.9.2.2 Payload** `CFE_ES_DeleteCDSCmd_Payload_t` `CFE_ES_DeleteCDSCmd::Payload`  
Command payload.  
Definition at line 268 of file default\_cfe\_es\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.10 CFE\_ES\_DeleteCDSCmd\_Payload Struct Reference

Delete Critical Data Store Command Payload.

```
#include <default_cfe_es_msgstruct.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 256 of file default\_cfe\_es\_msgstruct.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 259 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.11 CFE\_ES\_DumpCDSRegistryCmd Struct Reference

Dump CDS Registry Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t** `CommandHeader`  
*Command header.*
- **CFE\_ES\_DumpCDSRegistryCmd\_Payload\_t** `Payload`  
*Command payload.*

### 11.11.1 Detailed Description

Dump CDS Registry Command.

Definition at line 390 of file default\_cfe\_es\_msgstruct.h.

### 11.11.2 Field Documentation

**11.11.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_DumpCDSRegistryCmd::CommandHeader`  
Command header.

Definition at line 392 of file default\_cfe\_es\_msgstruct.h.

**11.11.2.2 Payload** `CFE_ES_DumpCDSRegistryCmd_Payload_t` `CFE_ES_DumpCDSRegistryCmd::Payload`  
Command payload.

Definition at line 393 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.12 CFE\_ES\_DumpCDSRegistryCmd\_Payload Struct Reference

Dump CDS Registry Command Payload.

```
#include <default_cfe_es_msgstruct.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 381 of file default\_cfe\_es\_msgstruct.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 383 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default\\_cfe\\_es\\_msgstruct.h](#)

## 11.13 CFE\_ES\_FileNameCmd Struct Reference

Generic file name command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CFE\\_ES\\_FileNameCmd\\_Payload\\_t Payload](#)  
*Command payload.*

### 11.13.1 Detailed Description

Generic file name command.

Definition at line 111 of file default\_cfe\_es\_msgstruct.h.

### 11.13.2 Field Documentation

#### 11.13.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_ES_FileNameCmd::CommandHeader`

Command header.

Definition at line 113 of file default\_cfe\_es\_msgstruct.h.

#### 11.13.2.2 Payload `CFE_ES_FileNameCmd_Payload_t CFE_ES_FileNameCmd::Payload`

Command payload.

Definition at line 114 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default\\_cfe\\_es\\_msgstruct.h](#)

## 11.14 CFE\_ES\_FileNameCmd\_Payload Struct Reference

Generic file name command payload.

```
#include <default_cfe_es_msgstruct.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 102 of file default\_cfe\_es\_msgstruct.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 104 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.15 CFE\_ES\_HousekeepingTlm Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

## Data Fields

- [CFE\\_MSG\\_TelemetryHeader\\_t TelemetryHeader](#)

*Telemetry header.*

- [CFE\\_ES\\_HousekeepingTlm\\_Payload\\_t Payload](#)

*Telemetry payload.*

### 11.15.1 Detailed Description

Definition at line 537 of file default\_cfe\_es\_msgstruct.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 540 of file default\_cfe\_es\_msgstruct.h.

#### 11.15.2.2 TelemetryHeader [CFE\\_MSG\\_TelemetryHeader\\_t CFE\\_ES\\_HousekeepingTlm::TelemetryHeader](#)

Telemetry header.

Definition at line 539 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.16 CFE\_ES\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_es_msgstruct.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 440 of file default\_cfe\_es\_msgstruct.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 506 of file default\_cfe\_es\_msgstruct.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 447 of file default\_cfe\_es\_msgstruct.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 449 of file default\_cfe\_es\_msgstruct.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 451 of file default\_cfe\_es\_msgstruct.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 455 of file default\_cfe\_es\_msgstruct.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 453 of file default\_cfe\_es\_msgstruct.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 442 of file default\_cfe\_es\_msgstruct.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 444 of file default\_cfe\_es\_msgstruct.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 486 of file default\_cfe\_es\_msgstruct.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 484 of file default\_cfe\_es\_msgstruct.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 531 of file default\_cfe\_es\_msgstruct.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 529 of file default\_cfe\_es\_msgstruct.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 533 of file default\_cfe\_es\_msgstruct.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 504 of file default\_cfe\_es\_msgstruct.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 457 of file default\_cfe\_es\_msgstruct.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 459 of file default\_cfe\_es\_msgstruct.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 463 of file default\_cfe\_es\_msgstruct.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 461 of file default\_cfe\_es\_msgstruct.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 524 of file default\_cfe\_es\_msgstruct.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 522 of file default\_cfe\_es\_msgstruct.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 520 of file default\_cfe\_es\_msgstruct.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 527 of file default\_cfe\_es\_msgstruct.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 515 of file default\_cfe\_es\_msgstruct.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 511 of file default\_cfe\_es\_msgstruct.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 509 of file default\_cfe\_es\_msgstruct.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_PerfTrigCntr`

Definition at line 513 of file default\_cfe\_es\_msgstruct.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 518 of file default\_cfe\_es\_msgstruct.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_ProcResetCntr`

Definition at line 502 of file default\_cfe\_es\_msgstruct.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 466 of file default\_cfe\_es\_msgstruct.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 468 of file default\_cfe\_es\_msgstruct.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 472 of file default\_cfe\_es\_msgstruct.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 470 of file default\_cfe\_es\_msgstruct.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 489 of file default\_cfe\_es\_msgstruct.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 491 of file default\_cfe\_es\_msgstruct.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 495 of file default\_cfe\_es\_msgstruct.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 493 of file `default_cfe_es_msgstruct.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 500 of file `default_cfe_es_msgstruct.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 498 of file `default_cfe_es_msgstruct.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 475 of file `default_cfe_es_msgstruct.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 479 of file `default_cfe_es_msgstruct.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 481 of file `default_cfe_es_msgstruct.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 477 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.17 CFE\_ES\_MemPoolStats Struct Reference

Memory Pool Statistics.

```
#include <default_cfe_es_extern_typedefs.h>
```

### Data Fields

- **CFE\_ES\_MemOffset\_t PoolSize**  
*Size of Memory Pool (in bytes)*
- **uint32 NumBlocksRequested**  
*Number of times a memory block has been allocated.*
- **uint32 CheckErrCtr**  
*Number of errors detected when freeing a memory block.*
- **CFE\_ES\_MemOffset\_t NumFreeBytes**  
*Number of bytes never allocated to a block.*
- **CFE\_ES\_BlockStats\_t BlockStats [CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS]**  
*Contains stats on each block size.*

### 11.17.1 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 553 of file default\_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 563 of file default\_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 559 of file default\_cfe\_es\_extern\_typedefs.h.

**11.17.2.3 NumBlocksRequested** [uint32](#) CFE\_ES\_MemPoolStats::NumBlocksRequested  
Number of times a memory block has been allocated.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_BlkREQ

Definition at line 557 of file default\_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 561 of file default\_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 555 of file default\_cfe\_es\_extern\_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_extern\_typedefs.h

## 11.18 CFE\_ES\_MemStatsTlm Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_TelemetryHeader\\_t TelemetryHeader](#)  
*Telemetry header.*
- [CFE\\_ES\\_PoolStatsTlm\\_Payload\\_t Payload](#)  
*Telemetry payload.*

### 11.18.1 Detailed Description

Definition at line 429 of file default\_cfe\_es\_msgstruct.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 432 of file default\_cfe\_es\_msgstruct.h.

**11.18.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_ES\_MemStatsTlm::TelemetryHeader  
Telemetry header.

Definition at line 431 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.19 CFE\_ES\_NoArgsCmd Struct Reference

Generic "no arguments" command.

```
#include <default_cfe_es_msgstruct.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 54 of file default\_cfe\_es\_msgstruct.h.

### 11.19.2 Field Documentation

#### 11.19.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_NoArgsCmd::CommandHeader

Command header.

Definition at line 58 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.20 CFE\_ES\_OneAppTlm Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

## Data Fields

- [CFE\\_MSG\\_TelemetryHeader\\_t](#) TelemetryHeader

*Telemetry header.*

- [CFE\\_ES\\_OneAppTlm\\_Payload\\_t](#) Payload

*Telemetry payload.*

### 11.20.1 Detailed Description

Definition at line 413 of file default\_cfe\_es\_msgstruct.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 416 of file default\_cfe\_es\_msgstruct.h.

**11.20.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_ES\_OneAppTlm::TelemetryHeader  
Telemetry header.

Definition at line 415 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.21 CFE\_ES\_OneAppTlm\_Payload Struct Reference

```
#include <default_cfe_es_msgstruct.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 408 of file default\_cfe\_es\_msgstruct.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 410 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.22 CFE\_ES\_OverWriteSysLogCmd Struct Reference

Overwrite/Discard System Log Configuration Command Payload.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader
  - Command header.*
- [CFE\\_ES\\_OverWriteSysLogCmd\\_Payload\\_t](#) Payload
  - Command payload.*

#### 11.22.1 Detailed Description

Overwrite/Discard System Log Configuration Command Payload.

Definition at line 141 of file default\_cfe\_es\_msgstruct.h.

#### 11.22.2 Field Documentation

**11.22.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_OverWriteSysLogCmd::CommandHeader  
Command header.  
Definition at line 143 of file default\_cfe\_es\_msgstruct.h.

**11.22.2.2 Payload** [CFE\\_ES\\_OverWriteSysLogCmd\\_Payload\\_t](#) CFE\_ES\_OverWriteSysLogCmd::Payload  
Command payload.  
Definition at line 144 of file default\_cfe\_es\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.23 CFE\_ES\_OverWriteSysLogCmd\_Payload Struct Reference

Overwrite/Discard System Log Configuration Command Payload.

```
#include <default_cfe_es_msgstruct.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 132 of file default\_cfe\_es\_msgstruct.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 134 of file default\_cfe\_es\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.24 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 105 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 110 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 109 of file cfe\_es\_api\_typedefs.h.

#### 11.24.2.3 Ptr void\* CFE\_ES\_PoolAlign::Ptr

Aligned pointer.

Definition at line 107 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 <default_cfe_es_msgstruct.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 422 of file default\_cfe\_es\_msgstruct.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 424 of file default\_cfe\_es\_msgstruct.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 426 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.26 CFE\_ES\_ReloadAppCmd Struct Reference

Reload Application Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_AppReloadCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.26.1 Detailed Description

Reload Application Command.

Definition at line 223 of file default\_cfe\_es\_msgstruct.h.

### 11.26.2 Field Documentation

#### 11.26.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_ReloadAppCmd::CommandHeader

Command header.

Definition at line 225 of file default\_cfe\_es\_msgstruct.h.

#### 11.26.2.2 Payload [CFE\\_ES\\_AppReloadCmd\\_Payload\\_t](#) CFE\_ES\_ReloadAppCmd::Payload

Command payload.

Definition at line 226 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.27 CFE\_ES\_RestartCmd Struct Reference

Restart cFE Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_RestartCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.27.1 Detailed Description

Restart cFE Command.

Definition at line 88 of file default\_cfe\_es\_msgstruct.h.

### 11.27.2 Field Documentation

**11.27.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_RestartCmd::CommandHeader  
Command header.

Definition at line 90 of file default\_cfe\_es\_msgstruct.h.

**11.27.2.2 Payload** [CFE\\_ES\\_RestartCmd\\_Payload\\_t](#) CFE\_ES\_RestartCmd::Payload  
Command payload.

Definition at line 91 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.28 CFE\_ES\_RestartCmd\_Payload Struct Reference

Restart cFE Command Payload.

```
#include <default_cfe_es_msgstruct.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 79 of file default\_cfe\_es\_msgstruct.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 81 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.29 CFE\_ES\_SendMemPoolStatsCmd Struct Reference

Send Memory Pool Statistics Command.

```
#include <default_cfe_es_msgstruct.h>
```

## Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_SendMemPoolStatsCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.29.1 Detailed Description

Send Memory Pool Statistics Command.

Definition at line 369 of file `default_cfe_es_msgstruct.h`.

### 11.29.2 Field Documentation

#### 11.29.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_SendMemPoolStatsCmd::CommandHeader

Command header.

Definition at line 371 of file `default_cfe_es_msgstruct.h`.

#### 11.29.2.2 Payload [CFE\\_ES\\_SendMemPoolStatsCmd\\_Payload\\_t](#) CFE\_ES\_SendMemPoolStatsCmd::Payload

Command payload.

Definition at line 372 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.30 CFE\_ES\_SendMemPoolStatsCmd\_Payload Struct Reference

Send Memory Pool Statistics Command Payload.

```
#include <default_cfe_es_msgstruct.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 360 of file `default_cfe_es_msgstruct.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 362 of file `default_cfe_es_msgstruct.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 363 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.31 CFE\_ES\_SetMaxPRCountCmd Struct Reference

Set Maximum Processor Reset Count Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_SetMaxPRCountCmd\\_Payload\\_t](#) Payload  
*Command payload.*

#### 11.31.1 Detailed Description

Set Maximum Processor Reset Count Command.

Definition at line 244 of file default\_cfe\_es\_msgstruct.h.

#### 11.31.2 Field Documentation

**11.31.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_SetMaxPRCountCmd::CommandHeader  
Command header.

Definition at line 246 of file default\_cfe\_es\_msgstruct.h.

**11.31.2.2 Payload** [CFE\\_ES\\_SetMaxPRCountCmd\\_Payload\\_t](#) CFE\_ES\_SetMaxPRCountCmd::Payload  
Command payload.

Definition at line 247 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.32 CFE\_ES\_SetMaxPRCountCmd\_Payload Struct Reference

Set Maximum Processor Reset Count Command Payload.

```
#include <default_cfe_es_msgstruct.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 235 of file default\_cfe\_es\_msgstruct.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 237 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.33 CFE\_ES\_SetPerfFilterMaskCmd Struct Reference

Set Performance Analyzer Filter Mask Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_ES_SetPerfFilterMaskCmd_Payload_t Payload`  
*Command payload.*

#### 11.33.1 Detailed Description

Set Performance Analyzer Filter Mask Command.

Definition at line 327 of file default\_cfe\_es\_msgstruct.h.

### 11.33.2 Field Documentation

#### 11.33.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_ES_SetPerfFilterMaskCmd::CommandHeader`

Command header.

Definition at line 329 of file default\_cfe\_es\_msgstruct.h.

#### 11.33.2.2 Payload `CFE_ES_SetPerfFilterMaskCmd_Payload_t CFE_ES_SetPerfFilterMaskCmd::Payload`

Command payload.

Definition at line 330 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.34 CFE\_ES\_SetPerfFilterMaskCmd\_Payload Struct Reference

Set Performance Analyzer Filter Mask Command Payload.

```
#include <default_cfe_es_msgstruct.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 318 of file default\_cfe\_es\_msgstruct.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 321 of file default\_cfe\_es\_msgstruct.h.

#### 11.34.2.2 FilterMaskNum `uint32` CFE\_ES\_SetPerfFilterMaskCmd\_Payload::FilterMaskNum

Index into array of Filter Masks.

Definition at line 320 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.35 CFE\_ES\_SetPerfTriggerMaskCmd Struct Reference

Set Performance Analyzer Trigger Mask Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t` CommandHeader  
*Command header.*
- `CFE_ES_SetPerfTrigMaskCmd_Payload_t` Payload  
*Command payload.*

### 11.35.1 Detailed Description

Set Performance Analyzer Trigger Mask Command.

Definition at line 348 of file default\_cfe\_es\_msgstruct.h.

### 11.35.2 Field Documentation

#### 11.35.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CFE\_ES\_SetPerfTriggerMaskCmd::CommandHeader

Command header.

Definition at line 350 of file default\_cfe\_es\_msgstruct.h.

#### 11.35.2.2 Payload `CFE_ES_SetPerfTrigMaskCmd_Payload_t` CFE\_ES\_SetPerfTriggerMaskCmd::Payload

Command payload.

Definition at line 351 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.36 CFE\_ES\_SetPerfTrigMaskCmd\_Payload Struct Reference

Set Performance Analyzer Trigger Mask Command Payload.

```
#include <default_cfe_es_msgstruct.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 339 of file default\_cfe\_es\_msgstruct.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 342 of file default\_cfe\_es\_msgstruct.h.

#### 11.36.2.2 TriggerMaskNum `uint32 CFE_ES_SetPerfTrigMaskCmd_Payload::TriggerMaskNum`

Index into array of Trigger Masks.

Definition at line 341 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default\\_cfe\\_es\\_msgstruct.h](#)

## 11.37 CFE\_ES\_StartApp Struct Reference

Start Application Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_ES_StartAppCmd_Payload_t Payload`  
*Command payload.*

### 11.37.1 Detailed Description

Start Application Command.

Definition at line 172 of file default\_cfe\_es\_msgstruct.h.

### 11.37.2 Field Documentation

**11.37.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_StartApp::CommandHeader  
Command header.  
Definition at line 174 of file default\_cfe\_es\_msgstruct.h.

**11.37.2.2 Payload** [CFE\\_ES\\_StartAppCmd\\_Payload\\_t](#) CFE\_ES\_StartApp::Payload  
Command payload.  
Definition at line 175 of file default\_cfe\_es\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.38 CFE\_ES\_StartAppCmd\_Payload Struct Reference

Start Application Command Payload.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- char [Application](#) [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]  
*Name of Application to be started.*
- char [AppEntryPoint](#) [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]  
*Symbolic name of Application's entry point.*
- char [AppFileName](#) [[CFE\\_MISSION\\_MAX\\_PATH\\_LEN](#)]  
*Full path and filename of Application's executable image.*
- [CFE\\_ES\\_MemOffset\\_t](#) [StackSize](#)  
*Desired stack size for the new application.*
- [CFE\\_ES\\_ExceptionAction\\_Enum\\_t](#) [ExceptionAction](#)  
*[CFE\\_ES\\_ExceptionAction\\_RESTART\\_APP](#)=On exception, restart Application, [CFE\\_ES\\_ExceptionAction\\_PROC\\_RESTART](#)=On exception, perform a Processor Reset*
- [CFE\\_ES\\_TaskPriority\\_Atom\\_t](#) [Priority](#)  
*The new Applications runtime priority.*

### 11.38.1 Detailed Description

Start Application Command Payload.

For command details, see [CFE\\_ES\\_START\\_APP\\_CC](#)

Definition at line 153 of file default\_cfe\_es\_msgstruct.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 156 of file default\_cfe\_es\_msgstruct.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 157 of file default\_cfe\_es\_msgstruct.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 155 of file `default_cfe_es_msgstruct.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_RESTART`=On exception, perform a Processor Reset

Definition at line 162 of file `default_cfe_es_msgstruct.h`.

**11.38.2.5 Priority** `CFE_ES_TaskPriority_Atom_t CFE_ES_StartAppCmd_Payload::Priority`

The new Applications runtime priority.

Definition at line 166 of file `default_cfe_es_msgstruct.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 160 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.39 CFE\_ES\_StartPerfCmd\_Payload Struct Reference

Start Performance Analyzer Command Payload.

```
#include <default_cfe_es_msgstruct.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 277 of file `default_cfe_es_msgstruct.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 279 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.40 CFE\_ES\_StartPerfDataCmd Struct Reference

Start Performance Analyzer Command.

```
#include <default_cfe_es_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_StartPerfCmd\\_Payload\\_t](#) Payload  
*Command payload.*

**11.40.1 Detailed Description**

Start Performance Analyzer Command.

Definition at line 285 of file default\_cfe\_es\_msgstruct.h.

**11.40.2 Field Documentation****11.40.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_StartPerfDataCmd::CommandHeader**

Command header.

Definition at line 287 of file default\_cfe\_es\_msgstruct.h.

**11.40.2.2 Payload [CFE\\_ES\\_StartPerfCmd\\_Payload\\_t](#) CFE\_ES\_StartPerfDataCmd::Payload**

Command payload.

Definition at line 288 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

**11.41 CFE\_ES\_StopPerfCmd\_Payload Struct Reference**

Stop Performance Analyzer Command Payload.

```
#include <default_cfe_es_msgstruct.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 297 of file default\_cfe\_es\_msgstruct.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 299 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.42 CFE\_ES\_StopPerfDataCmd Struct Reference

Stop Performance Analyzer Command.

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) `CommandHeader`  
*Command header.*
- [CFE\\_ES\\_StopPerfCmd\\_Payload\\_t](#) `Payload`  
*Command payload.*

### 11.42.1 Detailed Description

Stop Performance Analyzer Command.

Definition at line 306 of file default\_cfe\_es\_msgstruct.h.

### 11.42.2 Field Documentation

#### 11.42.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) `CFE_ES_StopPerfDataCmd::CommandHeader` Command header.

Definition at line 308 of file default\_cfe\_es\_msgstruct.h.

#### 11.42.2.2 Payload [CFE\\_ES\\_StopPerfCmd\\_Payload\\_t](#) `CFE_ES_StopPerfDataCmd::Payload` Command payload.

Definition at line 309 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.43 CFE\_ES\_TaskInfo Struct Reference

Task Information.

```
#include <default_cfe_es_extern_typedefs.h>
```

### Data Fields

- [CFE\\_ES\\_TaskId\\_t](#) `TaskId`  
*Task Id.*
- [uint32](#) `ExecutionCounter`  
*Task Execution Counter.*
- [char](#) `TaskName [CFE_MISSION_MAX_API_LEN]`  
*Task Name.*
- [CFE\\_ES\\_AppId\\_t](#) `Appld`  
*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 499 of file default\_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 504 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.43.2.2AppName [char](#) CFE\_ES\_TaskInfo::AppName[[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]

Parent Application Name.

Definition at line 505 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.43.2.3 ExecutionCounter [uint32](#) CFE\_ES\_TaskInfo::ExecutionCounter

Task Execution Counter.

Definition at line 502 of file default\_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 507 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.43.2.5 Spare [uint8](#) CFE\_ES\_TaskInfo::Spare[2]

Spare bytes for alignment

Definition at line 508 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.43.2.6 StackSize [CFE\\_ES\\_MemOffset\\_t](#) CFE\_ES\_TaskInfo::StackSize

Size of task stack

Definition at line 506 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.43.2.7 TaskId [CFE\\_ES\\_TaskId\\_t](#) CFE\_ES\_TaskInfo::TaskId

Task Id.

Definition at line 501 of file default\_cfe\_es\_extern\_typedefs.h.

**11.43.2.8 TaskName** char CFE\_ES\_TaskInfo::TaskName[CFE\_MISSION\_MAX\_API\_LEN]

Task Name.

Definition at line 503 of file default\_cfe\_es\_extern\_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_extern\_typedefs.h

## 11.44 CFE\_EVS\_AppDataCmd\_Payload Struct Reference

Write Event Services Application Information to File Command Payload.

```
#include <default_cfe_evs_msgstruct.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 89 of file default\_cfe\_evs\_msgstruct.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 91 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.45 CFE\_EVS\_AppNameBitMaskCmd Struct Reference

Generic App Name and Bitmask Command.

```
#include <default_cfe_evs_msgstruct.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 253 of file default\_cfe\_evs\_msgstruct.h.

#### 11.45.2 Field Documentation

**11.45.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_AppNameBitMaskCmd::CommandHeader  
Command header.  
Definition at line 255 of file default\_cfe\_evs\_msgstruct.h.

**11.45.2.2 Payload** [CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload\\_t](#) CFE\_EVS\_AppNameBitMaskCmd::Payload  
Command payload.  
Definition at line 256 of file default\_cfe\_evs\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.46 CFE\_EVS\_AppNameBitMaskCmd\_Payload Struct Reference

Generic App Name and Bitmask Command Payload.

```
#include <default_cfe_evs_msgstruct.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 243 of file default\_cfe\_evs\_msgstruct.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 245 of file default\_cfe\_evs\_msgstruct.h.

**11.46.2.2 BitMask** [uint8 CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload::BitMask](#)  
BitMask to use in the command.  
Definition at line 246 of file default\_cfe\_evs\_msgstruct.h.

**11.46.2.3 Spare** [uint8 CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload::Spare](#)  
Pad to even byte.  
Definition at line 247 of file default\_cfe\_evs\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.47 CFE\_EVS\_AppNameCmd Struct Reference

Generic App Name Command.

```
#include <default_cfe_evs_msgstruct.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 192 of file default\_cfe\_evs\_msgstruct.h.

### 11.47.2 Field Documentation

#### 11.47.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_AppNameCmd::CommandHeader

Command header.

Definition at line 194 of file default\_cfe\_evs\_msgstruct.h.

#### 11.47.2.2 Payload [CFE\\_EVS\\_AppNameCmd\\_Payload\\_t](#) CFE\_EVS\_AppNameCmd::Payload

Command payload.

Definition at line 195 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.48 CFE\_EVS\_AppNameCmd\_Payload Struct Reference

Generic App Name Command Payload.

```
#include <default_cfe_evs_msgstruct.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 184 of file default\_cfe\_evs\_msgstruct.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 186 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.49 CFE\_EVS\_AppNameEventIDCmd Struct Reference

Generic App Name and Event ID Command.

```
#include <default_cfe_evs_msgstruct.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 223 of file default\_cfe\_evs\_msgstruct.h.

### 11.49.2 Field Documentation

**11.49.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_AppNameEventIDCmd::CommandHeader  
Command header.

Definition at line 225 of file default\_cfe\_evs\_msgstruct.h.

**11.49.2.2 Payload** [CFE\\_EVS\\_AppNameEventIDCmd\\_Payload\\_t](#) CFE\_EVS\_AppNameEventIDCmd::Payload  
Command payload.

Definition at line 226 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.50 CFE\_EVS\_AppNameEventIDCmd\_Payload Struct Reference

Generic App Name and Event ID Command Payload.

```
#include <default_cfe_evs_msgstruct.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 214 of file default\_cfe\_evs\_msgstruct.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 216 of file default\_cfe\_evs\_msgstruct.h.

#### 11.50.2.2 EventID `uint16 CFE_EVS_AppNameEventIDCmd_Payload::EventID`

Event ID to use in the command.

Definition at line 217 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.51 CFE\_EVS\_AppNameEventIDMaskCmd Struct Reference

Generic App Name, Event ID, Mask Command.

```
#include <default_cfe_evs_msgstruct.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 284 of file default\_cfe\_evs\_msgstruct.h.

### 11.51.2 Field Documentation

#### 11.51.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_EVS_AppNameEventIDMaskCmd::CommandHeader`

Command header.

Definition at line 286 of file default\_cfe\_evs\_msgstruct.h.

#### 11.51.2.2 Payload `CFE_EVS_AppNameEventIDMaskCmd_Payload_t CFE_EVS_AppNameEventIDMaskCmd::Payload`

Command payload.

Definition at line 287 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.52 CFE\_EVS\_AppNameEventIDMaskCmd\_Payload Struct Reference

Generic App Name, Event ID, Mask Command Payload.

```
#include <default_cfe_evs_msgstruct.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 274 of file default\_cfe\_evs\_msgstruct.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 276 of file default\_cfe\_evs\_msgstruct.h.

#### 11.52.2.2 EventID [uint16 CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload::EventID](#)

Event ID to use in the command.

Definition at line 277 of file default\_cfe\_evs\_msgstruct.h.

#### 11.52.2.3 Mask [uint16 CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload::Mask](#)

Mask to use in the command.

Definition at line 278 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.53 CFE\_EVS\_AppTlmData Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_ES\\_AppId\\_t AppID](#)

*Numerical application identifier.*

- [uint16 AppMessageSentCounter](#)

*Application message sent counter.*

- [uint8 AppEnableStatus](#)

*Application event service enable status.*

- `uint8 AppMessageSquelchedCounter`

*Number of events squelched.*

### 11.53.1 Detailed Description

Definition at line 302 of file `default_cfe_evs_msgstruct.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 308 of file `default_cfe_evs_msgstruct.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 304 of file `default_cfe_evs_msgstruct.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 306 of file `default_cfe_evs_msgstruct.h`.

#### 11.53.2.4 `AppMessageSquelchedCounter` `uint8 CFE_EVS_AppTlmData::AppMessageSquelchedCounter`

Number of events squelched.

**Telemetry Mnemonic(s)** `$sc_$cpu_EVS_APP[CFE_PLATFORM_ES_MAX_APPLICATIONS].SQUELCHEDC`

Definition at line 310 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.54 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 <default_cfe_evs_msgstruct.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 161 of file default\_cfe\_evs\_msgstruct.h.

### 11.55.2 Field Documentation

#### 11.55.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_BitMaskCmd::CommandHeader

Command header.

Definition at line 163 of file default\_cfe\_evs\_msgstruct.h.

#### 11.55.2.2 Payload [CFE\\_EVS\\_BitMaskCmd\\_Payload\\_t](#) CFE\_EVS\_BitMaskCmd::Payload

Command payload.

Definition at line 164 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/evs/config/default\\_cfe\\_evs\\_msgstruct.h](#)

## 11.56 CFE\_EVS\_BitMaskCmd\_Payload Struct Reference

Generic Bitmask Command Payload.

```
#include <default_cfe_evs_msgstruct.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 152 of file default\_cfe\_evs\_msgstruct.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 154 of file default\_cfe\_evs\_msgstruct.h.

##### 11.56.2.2 Spare `uint8 CFE_EVS_BitMaskCmd_Payload::Spare`

Pad to even byte.

Definition at line 155 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.57 CFE\_EVS\_HousekeepingTlm Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`  
*Telemetry header.*
- `CFE_EVS_HousekeepingTlm_Payload_t Payload`  
*Telemetry payload.*

#### 11.57.1 Detailed Description

Definition at line 355 of file default\_cfe\_evs\_msgstruct.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 358 of file default\_cfe\_evs\_msgstruct.h.

**11.57.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_EVS\_HousekeepingTlm::TelemetryHeader  
Telemetry header.

Definition at line 357 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.58 CFE\_EVS\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_evs_msgstruct.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 317 of file default\_cfe\_evs\_msgstruct.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 351 of file default\_cfe\_evs\_msgstruct.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 319 of file default\_cfe\_evs\_msgstruct.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 321 of file default\_cfe\_evs\_msgstruct.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 342 of file default\_cfe\_evs\_msgstruct.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 332 of file default\_cfe\_evs\_msgstruct.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 334 of file default\_cfe\_evs\_msgstruct.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 339 of file default\_cfe\_evs\_msgstruct.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 323 of file default\_cfe\_evs\_msgstruct.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 337 of file default\_cfe\_evs\_msgstruct.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 325 of file default\_cfe\_evs\_msgstruct.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 330 of file default\_cfe\_evs\_msgstruct.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 344 of file default\_cfe\_evs\_msgstruct.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 346 of file default\_cfe\_evs\_msgstruct.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 348 of file default\_cfe\_evs\_msgstruct.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 328 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.59 CFE\_EVS\_LogFileCmd\_Payload Struct Reference

Write Event Log to File Command Payload.

```
#include <default_cfe_evs_msgstruct.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 69 of file default\_cfe\_evs\_msgstruct.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 71 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.60 CFE\_EVS\_LongEventTlm Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`  
*Telemetry header.*
- `CFE_EVS_LongEventTlm_Payload_t Payload`  
*Telemetry payload.*

### 11.60.1 Detailed Description

Definition at line 399 of file default\_cfe\_evs\_msgstruct.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 402 of file default\_cfe\_evs\_msgstruct.h.

**11.60.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_EVS\_LongEventTlm::TelemetryHeader  
Telemetry header.

Definition at line 401 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.61 CFE\_EVS\_LongEventTlm\_Payload Struct Reference

```
#include <default_cfe_evs_msgstruct.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 380 of file default\_cfe\_evs\_msgstruct.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 383 of file default\_cfe\_evs\_msgstruct.h.

**11.61.2.2 PacketID** `CFE_EVS_PacketID_t CFE_EVS_LongEventTlm_Payload::PacketID`  
Event packet information.

Definition at line 382 of file default\_cfe\_evs\_msgstruct.h.

**11.61.2.3 Spare1** `uint8 CFE_EVS_LongEventTlm_Payload::Spare1`  
Structure padding.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_SPARE1

Definition at line 385 of file default\_cfe\_evs\_msgstruct.h.

**11.61.2.4 Spare2** `uint8 CFE_EVS_LongEventTlm_Payload::Spare2`  
Structure padding.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_SPARE2

Definition at line 387 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.62 CFE\_EVS\_NoArgsCmd Struct Reference

Command with no additional arguments.

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

*Command header.*

### 11.62.1 Detailed Description

Command with no additional arguments.

Definition at line 48 of file default\_cfe\_evs\_msgstruct.h.

### 11.62.2 Field Documentation

**11.62.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_EVS_NoArgsCmd::CommandHeader`  
Command header.

Definition at line 52 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.63 CFE\_EVS\_PacketID Struct Reference

```
#include <default_cfe_evs_msgstruct.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 363 of file default\_cfe\_evs\_msgstruct.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 365 of file default\_cfe\_evs\_msgstruct.h.

**11.63.2.2 EventID** `uint16 CFE_EVS_PacketID::EventID`  
Numerical event identifier.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_EVENTID

Definition at line 367 of file default\_cfe\_evs\_msgstruct.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 369 of file default\_cfe\_evs\_msgstruct.h.

**11.63.2.4 ProcessorID** `uint32 CFE_EVS_PacketID::ProcessorID`  
Numerical processor identifier.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_PROCESSORID

Definition at line 373 of file default\_cfe\_evs\_msgstruct.h.

**11.63.2.5 SpacecraftID** `uint32 CFE_EVS_PacketID::SpacecraftID`  
Spacecraft identifier.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_SCID

Definition at line 371 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.64 CFE\_EVS\_SetEventFormatCode\_Payload Struct Reference

Set Event Format Mode Command Payload.

```
#include <default_cfe_evs_msgstruct.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 130 of file default\_cfe\_evs\_msgstruct.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 132 of file default\_cfe\_evs\_msgstruct.h.

**11.64.2.2 Spare** `uint8 CFE_EVS_SetEventFormatCode_Payload::Spare`

Pad to even byte.

Definition at line 133 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.65 CFE\_EVS\_SetEventFormatModeCmd Struct Reference

Set Event Format Mode Command.

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t CommandHeader**  
*Command header.*
- **CFE\_EVS\_SetEventFormatMode\_Payload\_t Payload**  
*Command payload.*

### 11.65.1 Detailed Description

Set Event Format Mode Command.

Definition at line 139 of file default\_cfe\_evs\_msgstruct.h.

### 11.65.2 Field Documentation

#### 11.65.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_SetEventFormatModeCmd::CommandHeader

Command header.

Definition at line 141 of file default\_cfe\_evs\_msgstruct.h.

#### 11.65.2.2 Payload [CFE\\_EVS\\_SetEventFormatMode\\_Payload\\_t](#) CFE\_EVS\_SetEventFormatModeCmd::Payload

Command payload.

Definition at line 142 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.66 CFE\_EVS\_SetLogMode\_Payload Struct Reference

Set Log Mode Command Payload.

```
#include <default_cfe_evs_msgstruct.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 109 of file default\_cfe\_evs\_msgstruct.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 111 of file default\_cfe\_evs\_msgstruct.h.

#### 11.66.2.2 Spare [uint8](#) CFE\_EVS\_SetLogMode\_Payload::Spare

Pad to even byte.

Definition at line 112 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.67 CFE\_EVS\_SetLogModeCmd Struct Reference

Set Log Mode Command.

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_SetLogMode\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.67.1 Detailed Description

Set Log Mode Command.

Definition at line 118 of file default\_cfe\_evs\_msgstruct.h.

### 11.67.2 Field Documentation

#### 11.67.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_SetLogModeCmd::CommandHeader

Command header.

Definition at line 120 of file default\_cfe\_evs\_msgstruct.h.

#### 11.67.2.2 Payload [CFE\\_EVS\\_SetLogMode\\_Payload\\_t](#) CFE\_EVS\_SetLogModeCmd::Payload

Command payload.

Definition at line 121 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.68 CFE\_EVS\_ShortEventTlm Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_TelemetryHeader\\_t](#) TelemetryHeader  
*Telemetry header.*
- [CFE\\_EVS\\_ShortEventTlm\\_Payload\\_t](#) Payload  
*Telemetry payload.*

### 11.68.1 Detailed Description

Definition at line 405 of file default\_cfe\_evs\_msgstruct.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 408 of file `default_cfe_evs_msgstruct.h`.

**11.68.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_EVS_ShortEventTlm::TelemetryHeader`  
Telemetry header.

Definition at line 407 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.69 CFE\_EVS\_ShortEventTlm\_Payload Struct Reference

```
#include <default_cfe_evs_msgstruct.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 394 of file `default_cfe_evs_msgstruct.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 396 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.70 CFE\_EVS\_WriteAppDataFileCmd Struct Reference

Write Event Services Application Information to File Command.

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

*Command header.*

- `CFE_EVS_AppDataCmd_Payload_t Payload`

*Command payload.*

#### 11.70.1 Detailed Description

Write Event Services Application Information to File Command.

Definition at line 97 of file `default_cfe_evs_msgstruct.h`.

## 11.70.2 Field Documentation

### 11.70.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_WriteAppDataFileCmd::CommandHeader

Command header.

Definition at line 99 of file default\_cfe\_evs\_msgstruct.h.

### 11.70.2.2 Payload [CFE\\_EVS\\_AppDataCmd\\_Payload\\_t](#) CFE\_EVS\_WriteAppDataFileCmd::Payload

Command payload.

Definition at line 100 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.71 CFE\_EVS\_WriteLogFileCmd Struct Reference

Write Event Log to File Command.

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_LogFileCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.71.1 Detailed Description

Write Event Log to File Command.

Definition at line 77 of file default\_cfe\_evs\_msgstruct.h.

## 11.71.2 Field Documentation

### 11.71.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_WriteLogFileCmd::CommandHeader

Command header.

Definition at line 79 of file default\_cfe\_evs\_msgstruct.h.

### 11.71.2.2 Payload [CFE\\_EVS\\_LogFileCmd\\_Payload\\_t](#) CFE\_EVS\_WriteLogFileCmd::Payload

Command payload.

Definition at line 80 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.72 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 123 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 131 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 127 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 130 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 133 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 125 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 134 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 <default_cfe_fs_filedef.h>
```

### Data Fields

- `uint32 ContentType`  
*Identifies the content type (=cFE1'=0x63464531)*
- `uint32 SubType`  
*Type of ContentType, if necessary.*
- `uint32 Length`  
*Length of this header to support external processing.*
- `uint32 SpacecraftID`  
*Spacecraft that generated the file.*
- `uint32 ProcessorID`  
*Processor that generated the file.*
- `uint32 ApplicationID`  
*Application that generated the file.*
- `uint32 TimeSeconds`  
*File creation timestamp (seconds)*
- `uint32 TimeSubSeconds`  
*File creation timestamp (sub-seconds)*
- `char Description [CFE_FS_HDR_DESC_MAX_LEN]`  
*File description.*

### 11.73.1 Detailed Description

Standard cFE File header structure definition.

Definition at line 181 of file default\_cfe\_fs\_filedef.h.

### 11.73.2 Field Documentation

#### 11.73.2.1 ApplicationID `uint32 CFE_FS_Header::ApplicationID`

Application that generated the file.

Definition at line 190 of file default\_cfe\_fs\_filedef.h.

#### 11.73.2.2 ContentType `uint32 CFE_FS_Header::ContentType`

Identifies the content type (=cFE1'=0x63464531)

Definition at line 183 of file default\_cfe\_fs\_filedef.h.

#### 11.73.2.3 Description `char CFE_FS_Header::Description[CFE_FS_HDR_DESC_MAX_LEN]`

File description.

Definition at line 195 of file default\_cfe\_fs\_filedef.h.

**11.73.2.4 Length** `uint32` `CFE_FS_Header::Length`

Length of this header to support external processing.

Definition at line 187 of file `default_cfe_fs_filedef.h`.

**11.73.2.5 ProcessorID** `uint32` `CFE_FS_Header::ProcessorID`

Processor that generated the file.

Definition at line 189 of file `default_cfe_fs_filedef.h`.

**11.73.2.6 SpacecraftID** `uint32` `CFE_FS_Header::SpacecraftID`

Spacecraft that generated the file.

Definition at line 188 of file `default_cfe_fs_filedef.h`.

**11.73.2.7 SubType** `uint32` `CFE_FS_Header::SubType`

Type of `ContentType`, if necessary.

Standard SubType definitions can be found [here](#)

Definition at line 184 of file `default_cfe_fs_filedef.h`.

**11.73.2.8 TimeSeconds** `uint32` `CFE_FS_Header::TimeSeconds`

File creation timestamp (seconds)

Definition at line 192 of file `default_cfe_fs_filedef.h`.

**11.73.2.9 TimeSubSeconds** `uint32` `CFE_FS_Header::TimeSubSeconds`

File creation timestamp (sub-seconds)

Definition at line 193 of file `default_cfe_fs_filedef.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/fs/config/default_cfe_fs_filedef.h`

## 11.74 CFE\_SB\_AllSubscriptionsTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_TelemetryHeader\_t TelemetryHeader**

*Telemetry header.*

- **CFE\_SB\_AllSubscriptionsTlm\_Payload\_t Payload**

*Telemetry payload.*

### 11.74.1 Detailed Description

Definition at line 356 of file `default_cfe_sb_msgstruct.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 359 of file default\_cfe\_sb\_msgstruct.h.

**11.74.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_SB\_AllSubscriptionsTlm::TelemetryHeader  
Telemetry header.

Definition at line 358 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.75 CFE\_SB\_AllSubscriptionsTlm\_Payload Struct Reference

```
#include <default_cfe_sb_msgstruct.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 348 of file default\_cfe\_sb\_msgstruct.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 352 of file default\_cfe\_sb\_msgstruct.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 353 of file default\_cfe\_sb\_msgstruct.h.

**11.75.2.3 PktSegment** `uint32` `CFE_SB_AllSubscriptionsTlm_Payload::PktSegment`  
Pkt number(starts at 1) in the series.  
Definition at line 350 of file default\_cfe\_sb\_msgstruct.h.

**11.75.2.4 TotalSegments** `uint32` `CFE_SB_AllSubscriptionsTlm_Payload::TotalSegments`  
Total number of pkts needed to complete the request.  
Definition at line 351 of file default\_cfe\_sb\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default\\_cfe\\_sb\\_msgstruct.h](#)

## 11.76 CFE\_SB\_HousekeepingTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`  
*Telemetry header.*
- `CFE_SB_HousekeepingTlm_Payload_t Payload`  
*Telemetry payload.*

### 11.76.1 Detailed Description

Definition at line 166 of file default\_cfe\_sb\_msgstruct.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 169 of file default\_cfe\_sb\_msgstruct.h.

**11.76.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_SB_HousekeepingTlm::TelemetryHeader`  
Telemetry header.  
Definition at line 168 of file default\_cfe\_sb\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default\\_cfe\\_sb\\_msgstruct.h](#)

## 11.77 CFE\_SB\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_sb_msgstruct.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 GetPipIdByNameErrorCounter`  
*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 123 of file default\_cfe\_sb\_msgstruct.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 127 of file default\_cfe\_sb\_msgstruct.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 129 of file default\_cfe\_sb\_msgstruct.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 140 of file default\_cfe\_sb\_msgstruct.h.

**11.77.2.4 DuplicateSubscriptionsCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::DuplicateSubscriptions`←  
Counter  
Count of duplicate subscriptions.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_DupSubCnt

Definition at line 146 of file default\_cfe\_sb\_msgstruct.h.

**11.77.2.5 GetPipeIdByNameErrorCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::GetPipeIdByName`←  
ErrorCounter  
Count of errors in get pipe id by name API.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_GetPipeIDByNameEC

Definition at line 148 of file default\_cfe\_sb\_msgstruct.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 138 of file default\_cfe\_sb\_msgstruct.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 161 of file default\_cfe\_sb\_msgstruct.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 158 of file default\_cfe\_sb\_msgstruct.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 155 of file `default_cfe_sb_msgstruct.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 136 of file `default_cfe_sb_msgstruct.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 133 of file `default_cfe_sb_msgstruct.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 131 of file `default_cfe_sb_msgstruct.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 144 of file `default_cfe_sb_msgstruct.h`.

**11.77.2.14 PipeOverflowErrorCounter** `uint16` `CFE_SB_HousekeepingTlm_Payload::PipeOverflowError←`  
Counter  
Count of pipe overflow errors.

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_PipeOvrEC`

Definition at line 153 of file `default_cfe_sb_msgstruct.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 150 of file `default_cfe_sb_msgstruct.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 142 of file default\_cfe\_sb\_msgstruct.h.

**11.77.2.17 UnmarkedMem** `uint32 CFE_SB_HousekeepingTlm_Payload::UnmarkedMem`  
cfg param CFE\_PLATFORM\_SB\_BUFSIZE minus Peak Memory in use

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_UnMarkedMem

Definition at line 164 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.78 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 [CS\\_AppPipe\(\)](#), and [CS\\_ProcessCmd\(\)](#).

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 <default_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 104 of file [default\\_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 106 of file [default\\_cfe\\_sb\\_extern\\_typedefs.h](#).

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default\\_cfe\\_sb\\_extern\\_typedefs.h](#)

## 11.80 CFE\_SB\_MsgMapFileEntry Struct Reference

SB Map File Entry.

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_SB\\_MsgId\\_t](#) MsgId
  - Message Id which has been subscribed to.*
- [CFE\\_SB\\_RoutId\\_Atom\\_t](#) Index
  - Routing raw index value (0 based, not Route ID)*

#### 11.80.1 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 294 of file [default\\_cfe\\_sb\\_msgstruct.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 297 of file default\_cfe\_sb\_msgstruct.h.

### 11.80.2.2 MsgId `CFE_SB_MsgId_t` `CFE_SB_MsgMapFileEntry::MsgId`

Message Id which has been subscribed to.

Definition at line 296 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.81 CFE\_SB\_PipeDepthStats Struct Reference

SB Pipe Depth Statistics.

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_SB_Pipeld_t Pipeld`

*Pipe Id associated with the stats below.*

- `uint16 MaxQueueDepth`

*Number of messages the pipe can hold.*

- `uint16 CurrentQueueDepth`

*Number of messages currently on the pipe.*

- `uint16 PeakQueueDepth`

*Peak number of messages that have been on the pipe.*

- `uint16 Spare`

*Spare word to ensure alignment.*

### 11.81.1 Detailed Description

SB Pipe Depth Statistics.

Used in SB Statistics Telemetry Packet `CFE_SB_StatsTlm_t`

Definition at line 177 of file default\_cfe\_sb\_msgstruct.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 183 of file default\_cfe\_sb\_msgstruct.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 181 of file default\_cfe\_sb\_msgstruct.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 185 of file default\_cfe\_sb\_msgstruct.h.

**11.81.2.4 PipelId** `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 179 of file default\_cfe\_sb\_msgstruct.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 187 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.82 CFE\_SB\_PipeInfoEntry Struct Reference

SB Pipe Information File Entry.

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_SB_PipeId_t PipeId`
- `CFE_ES_AppId_t AppId`
- `char PipeName [CFE_MISSION_MAX_API_LEN]`
- `charAppName [CFE_MISSION_MAX_API_LEN]`
- `uint16 MaxQueueDepth`
- `uint16 CurrentQueueDepth`
- `uint16 PeakQueueDepth`
- `uint16 SendErrors`
- `uint8 Opts`
- `uint8 Spare [3]`

### 11.82.1 Detailed Description

SB Pipe Information File Entry.

This statistics structure is output as part of the CFE SB "Send Pipe Info" command (CFE\_SB\_SEND\_PIPE\_INFO\_CC). Previous versions of CFE simply wrote the internal CFE\_SB\_PipeD\_t object to the file, but this also contains information such as pointers which are not relevant outside the running CFE process.

By defining the pipe info structure separately, it also provides some independence, such that the internal CFE\_SB\_PipeD\_t definition can evolve without changing the binary format of the information file.

Definition at line 206 of file default\_cfe\_sb\_msgstruct.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 209 of file default\_cfe\_sb\_msgstruct.h.

#### 11.82.2.2AppName `char` CFE\_SB\_PipeInfoEntry::AppName [`CFE_MISSION_MAX_API_LEN`]

The Name of the application that owns the pipe

Definition at line 211 of file default\_cfe\_sb\_msgstruct.h.

#### 11.82.2.3 CurrentQueueDepth `uint16` CFE\_SB\_PipeInfoEntry::CurrentQueueDepth

The current depth of the pipe

Definition at line 213 of file default\_cfe\_sb\_msgstruct.h.

#### 11.82.2.4 MaxQueueDepth `uint16` CFE\_SB\_PipeInfoEntry::MaxQueueDepth

The allocated depth of the pipe (max capacity)

Definition at line 212 of file default\_cfe\_sb\_msgstruct.h.

#### 11.82.2.5 Opts `uint8` CFE\_SB\_PipeInfoEntry::Opts

Pipe options set (bitmask)

Definition at line 216 of file default\_cfe\_sb\_msgstruct.h.

#### 11.82.2.6 PeakQueueDepth `uint16` CFE\_SB\_PipeInfoEntry::PeakQueueDepth

The peak depth of the pipe (high watermark)

Definition at line 214 of file default\_cfe\_sb\_msgstruct.h.

#### 11.82.2.7 PipeId `CFE_SB_PipeId_t` CFE\_SB\_PipeInfoEntry::PipeId

The runtime ID of the pipe

Definition at line 208 of file default\_cfe\_sb\_msgstruct.h.

#### 11.82.2.8 PipeName `char` CFE\_SB\_PipeInfoEntry::PipeName [`CFE_MISSION_MAX_API_LEN`]

The Name of the pipe

Definition at line 210 of file default\_cfe\_sb\_msgstruct.h.

**11.82.2.9 SendErrors** `uint16 CFE_SB_PipeInfoEntry::SendErrors`

Number of errors when writing to this pipe

Definition at line 215 of file default\_cfe\_sb\_msgstruct.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 217 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default\\_cfe\\_sb\\_msgstruct.h](#)

## 11.83 CFE\_SB\_Qos\_t Struct Reference

Quality Of Service Type Definition.

```
#include <default_cfe_sb_extern_typedefs.h>
```

### Data Fields

- `uint8 Priority`  
*Specify high(1) or low(0) message priority for off-board routing, currently unused.*
- `uint8 Reliability`  
*Specify high(1) or low(0) message transfer reliability for off-board routing, currently unused.*

### 11.83.1 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 121 of file default\_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 123 of file default\_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 124 of file default\_cfe\_sb\_extern\_typedefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default\\_cfe\\_sb\\_extern\\_typedefs.h](#)

## 11.84 CFE\_SB\_RouteCmd Struct Reference

Enable/Disable Route Command.

```
#include <default_cfe_sb_msgstruct.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 104 of file default\_cfe\_sb\_msgstruct.h.

### 11.84.2 Field Documentation

#### 11.84.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_RouteCmd::CommandHeader

Command header.

Definition at line 106 of file default\_cfe\_sb\_msgstruct.h.

#### 11.84.2.2 Payload [CFE\\_SB\\_RouteCmd\\_Payload\\_t](#) CFE\_SB\_RouteCmd::Payload

Command payload.

Definition at line 107 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.85 CFE\_SB\_RouteCmd\_Payload Struct Reference

Enable/Disable Route Command Payload.

```
#include <default_cfe_sb_msgstruct.h>
```

## Data Fields

- [CFE\\_SB\\_MsgId\\_t](#) MsgId  
*Message ID of route to be enabled or disabled [CFE\\_SB\\_MsgId\\_t](#).*
- [CFE\\_SB\\_Pipeld\\_t](#) Pipe  
*Pipe ID of route to be enabled or disabled [CFE\\_SB\\_Pipeld\\_t](#).*
- [uint8](#) Spare  
*Spare byte to make command even number of bytes.*

### 11.85.1 Detailed Description

Enable/Disable Route Command Payload.

This structure contains a definition used by two SB commands, 'Enable Route' [CFE\\_SB\\_ENABLE\\_ROUTE\\_CC](#) and 'Disable Route' [CFE\\_SB\\_DISABLE\\_ROUTE\\_CC](#). A route is the destination pipe for a particular message and is therefore defined as a MsgId and Pipeld combination.

Definition at line 94 of file default\_cfe\_sb\_msgstruct.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 96 of file default\_cfe\_sb\_msgstruct.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 97 of file default\_cfe\_sb\_msgstruct.h.

**11.85.2.3 Spare** [uint8](#) CFE\_SB\_RouteCmd\_Payload::Spare  
Spare byte to make command even number of bytes.  
Definition at line 98 of file default\_cfe\_sb\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.86 CFE\_SB\_RoutingFileEntry Struct Reference

SB Routing File Entry.

```
#include <default_cfe_sb_msgstruct.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 279 of file default\_cfe\_sb\_msgstruct.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 285 of file default\_cfe\_sb\_msgstruct.h.

**11.86.2.2 MsgCnt** `uint16` `CFE_SB_RoutingFileEntry::MsgCnt`

Number of msgs with this MsgId sent to this PipeId.

Definition at line 284 of file default\_cfe\_sb\_msgstruct.h.

**11.86.2.3 MsgId** `CFE_SB_MsgId_t` `CFE_SB_RoutingFileEntry::MsgId`

Message Id portion of the route.

Definition at line 281 of file default\_cfe\_sb\_msgstruct.h.

**11.86.2.4 PipeId** `CFE_SB_PipeId_t` `CFE_SB_RoutingFileEntry::PipeId`

Pipe Id portion of the route.

Definition at line 282 of file default\_cfe\_sb\_msgstruct.h.

**11.86.2.5 PipeName** `char` `CFE_SB_RoutingFileEntry::PipeName[CFE_MISSION_MAX_API_LEN]`

Pipe Depth Statistics.

Definition at line 286 of file default\_cfe\_sb\_msgstruct.h.

**11.86.2.6 State** `uint8` `CFE_SB_RoutingFileEntry::State`

Route Enabled or Disabled.

Definition at line 283 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.87 CFE\_SB\_SingleSubscriptionTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`  
*Telemetry header.*
- `CFE_SB_SingleSubscriptionTlm_Payload_t Payload`  
*Telemetry payload.*

### 11.87.1 Detailed Description

Definition at line 318 of file default\_cfe\_sb\_msgstruct.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 321 of file default\_cfe\_sb\_msgstruct.h.

**11.87.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_SB\_SingleSubscriptionTlm::TelemetryHeader  
Telemetry header.

Definition at line 320 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.88 CFE\_SB\_SingleSubscriptionTlm\_Payload Struct Reference

```
#include <default_cfe_sb_msgstruct.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 310 of file default\_cfe\_sb\_msgstruct.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 313 of file default\_cfe\_sb\_msgstruct.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 315 of file default\_cfe\_sb\_msgstruct.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 314 of file default\_cfe\_sb\_msgstruct.h.

**11.88.2.4 SubType** `uint8` `CFE_SB_SingleSubscriptionTlm_Payload::SubType`  
Subscription or Unsubscription.

Definition at line 312 of file `default_cfe_sb_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgstruct.h`

## 11.89 CFE\_SB\_StatsTlm Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`  
*Telemetry header.*
- `CFE_SB_StatsTlm_Payload_t Payload`  
*Telemetry payload.*

#### 11.89.1 Detailed Description

Definition at line 268 of file `default_cfe_sb_msgstruct.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 271 of file `default_cfe_sb_msgstruct.h`.

**11.89.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_SB_StatsTlm::TelemetryHeader`  
Telemetry header.

Definition at line 270 of file `default_cfe_sb_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgstruct.h`

## 11.90 CFE\_SB\_StatsTlm\_Payload Struct Reference

```
#include <default_cfe_sb_msgstruct.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 225 of file default\_cfe\_sb\_msgstruct.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 245 of file default\_cfe\_sb\_msgstruct.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\_SMMMDALW

Definition at line 231 of file default\_cfe\_sb\_msgstruct.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 261 of file `default_cfe_sb_msgstruct.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 238 of file `default_cfe_sb_msgstruct.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 252 of file `default_cfe_sb_msgstruct.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 241 of file `default_cfe_sb_msgstruct.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 227 of file `default_cfe_sb_msgstruct.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 243 of file `default_cfe_sb_msgstruct.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 229 of file `default_cfe_sb_msgstruct.h`.

**11.90.2.10 PeakPipesInUse** `uint32` `CFE_SB_StatsTlm_Payload::PeakPipesInUse`  
Peak number of pipes since last reboot.

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMPIU`

Definition at line 236 of file `default_cfe_sb_msgstruct.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_SMPSSIU`

Definition at line 258 of file `default_cfe_sb_msgstruct.h`.

**11.90.2.12 PeakSubscriptionsInUse** `uint32` `CFE_SB_StatsTlm_Payload::PeakSubscriptionsInUse`  
Peak number of subscriptions.

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMPSSIU`

Definition at line 250 of file `default_cfe_sb_msgstruct.h`.

**11.90.2.13 PipeDepthStats** `CFE_SB_PipeDepthStats_t` `CFE_SB_StatsTlm_Payload::PipeDepthStats[CFE_MISSION_SB_MAX_PIPE]`  
Pipe Depth Statistics `CFE_SB_PipeDepthStats_t`.

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMPDS[CFE_PLATFORM_SB_MAX_PIPES]`

Definition at line 264 of file `default_cfe_sb_msgstruct.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 234 of file `default_cfe_sb_msgstruct.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 256 of file `default_cfe_sb_msgstruct.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 248 of file `default_cfe_sb_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgstruct.h`

## 11.91 CFE\_SB\_SubEntries Struct Reference

SB Previous Subscriptions Entry.

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_SB\\_MsgId\\_t MsgId](#)  
*MsgId portion of the subscription.*
- [CFE\\_SB\\_Qos\\_t Qos](#)  
*Qos portion of the subscription.*
- [CFE\\_SB\\_PipeId\\_t Pipe](#)  
*PipeId portion of the subscription.*

### 11.91.1 Detailed Description

SB Previous Subscriptions Entry.

This structure defines an entry used in the CFE\_SB\_PrevSubsPkt\_t Intended to be used primarily by Software Bus Networking Application (SBN)

Used in structure definition [CFE\\_SB\\_AllSubscriptionsTim\\_t](#)

Definition at line 332 of file default\_cfe\_sb\_msgstruct.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 334 of file default\_cfe\_sb\_msgstruct.h.

#### 11.91.2.2 Pipe [CFE\\_SB\\_PipeId\\_t](#) CFE\_SB\_SubEntries::Pipe

PipeId portion of the subscription.

Definition at line 336 of file default\_cfe\_sb\_msgstruct.h.

#### 11.91.2.3 Qos [CFE\\_SB\\_Qos\\_t](#) CFE\_SB\_SubEntries::Qos

Qos portion of the subscription.

Definition at line 335 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.92 CFE\_SB\_WriteFileInfoCmd Struct Reference

Write File Info Command.

```
#include <default_cfe_sb_msgstruct.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 73 of file default\_cfe\_sb\_msgstruct.h.

### 11.92.2 Field Documentation

#### 11.92.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_WriteFileInfoCmd::CommandHeader

Command header.

Definition at line 75 of file default\_cfe\_sb\_msgstruct.h.

#### 11.92.2.2 Payload [CFE\\_SB\\_WriteFileInfoCmd\\_Payload\\_t](#) CFE\_SB\_WriteFileInfoCmd::Payload

Command payload.

Definition at line 76 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.93 CFE\_SB\_WriteFileInfoCmd\_Payload Struct Reference

Write File Info Command Payload.

```
#include <default_cfe_sb_msgstruct.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 65 of file default\_cfe\_sb\_msgstruct.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 67 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.94 CFE\_TBL\_AbortLoadCmd Struct Reference

Abort Load Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

## Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TBL\\_AbortLoadCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.94.1 Detailed Description

Abort Load Command.

Definition at line 249 of file default\_cfe\_tbl\_msgstruct.h.

### 11.94.2 Field Documentation

#### 11.94.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_AbortLoadCmd::CommandHeader

Command header.

Definition at line 251 of file default\_cfe\_tbl\_msgstruct.h.

#### 11.94.2.2 Payload [CFE\\_TBL\\_AbortLoadCmd\\_Payload\\_t](#) CFE\_TBL\_AbortLoadCmd::Payload

Command payload.

Definition at line 252 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.95 CFE\_TBL\_AbortLoadCmd\_Payload Struct Reference

Abort Load Command Payload.

```
#include <default_cfe_tbl_msgstruct.h>
```

## Data Fields

- char [TableIdentifier](#) [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 239 of file default\_cfe\_tbl\_msgstruct.h.

### 11.95.2 Field Documentation

#### 11.95.2.1 TableIdentifier char CFE\_TBL\_AbortLoadCmd\_Payload::TableIdentifier [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 241 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.96 CFE\_TBL\_ActivateCmd Struct Reference

Activate Table Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TBL\\_ActivateCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.96.1 Detailed Description

Activate Table Command.

Definition at line 160 of file default\_cfe\_tbl\_msgstruct.h.

### 11.96.2 Field Documentation

#### 11.96.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_ActivateCmd::CommandHeader

Command header.

Definition at line 162 of file default\_cfe\_tbl\_msgstruct.h.

#### 11.96.2.2 Payload [CFE\\_TBL\\_ActivateCmd\\_Payload\\_t](#) CFE\_TBL\_ActivateCmd::Payload

Command payload.

Definition at line 163 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.97 CFE\_TBL\_ActivateCmd\_Payload Struct Reference

Activate Table Command Payload.

```
#include <default_cfe_tbl_msgstruct.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 150 of file default\_cfe\_tbl\_msgstruct.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 152 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.98 CFE\_TBL\_DelCDSCmd\_Payload Struct Reference

Delete Critical Table CDS Command Payload.

```
#include <default_cfe_tbl_msgstruct.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 216 of file default\_cfe\_tbl\_msgstruct.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 218 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.99 CFE\_TBL\_DeleteCDSCmd Struct Reference

Delete Critical Table CDS Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) **CommandHeader**

*Command header.*

- [CFE\\_TBL\\_DelCDSCmd\\_Payload\\_t](#) **Payload**

*Command payload.*

### 11.99.1 Detailed Description

Delete Critical Table CDS Command.

Definition at line 228 of file default\_cfe\_tbl\_msgstruct.h.

## 11.99.2 Field Documentation

### 11.99.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_DeleteCDSCmd::CommandHeader

Command header.

Definition at line 230 of file default\_cfe\_tbl\_msgstruct.h.

### 11.99.2.2 Payload [CFE\\_TBL\\_DelCDSCmd\\_Payload\\_t](#) CFE\_TBL\_DeleteCDSCmd::Payload

Command payload.

Definition at line 231 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.100 CFE\_TBL\_DumpCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TBL\\_DumpCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.100.1 Detailed Description

/brief Dump Table Command

Definition at line 112 of file default\_cfe\_tbl\_msgstruct.h.

### 11.100.2 Field Documentation

#### 11.100.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_DumpCmd::CommandHeader

Command header.

Definition at line 114 of file default\_cfe\_tbl\_msgstruct.h.

#### 11.100.2.2 Payload [CFE\\_TBL\\_DumpCmd\\_Payload\\_t](#) CFE\_TBL\_DumpCmd::Payload

Command payload.

Definition at line 115 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.101 CFE\_TBL\_DumpCmd\_Payload Struct Reference

Dump Table Command Payload.

```
#include <default_cfe_tbl_msgstruct.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 93 of file default\_cfe\_tbl\_msgstruct.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 95 of file default\_cfe\_tbl\_msgstruct.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 104 of file default\_cfe\_tbl\_msgstruct.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 101 of file default\_cfe\_tbl\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.102 CFE\_TBL\_DumpRegistryCmd Struct Reference

Dump Registry Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

## Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_TBL_DumpRegistryCmd_Payload_t Payload`  
*Command payload.*

### 11.102.1 Detailed Description

Dump Registry Command.

Definition at line 182 of file default\_cfe\_tbl\_msgstruct.h.

### 11.102.2 Field Documentation

**11.102.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_DumpRegistryCmd::CommandHeader  
Command header.

Definition at line 184 of file default\_cfe\_tbl\_msgstruct.h.

**11.102.2.2 Payload** [CFE\\_TBL\\_DumpRegistryCmd\\_Payload\\_t](#) CFE\_TBL\_DumpRegistryCmd::Payload  
Command payload.

Definition at line 185 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.103 CFE\_TBL\_DumpRegistryCmd\_Payload Struct Reference

Dump Registry Command Payload.

```
#include <default_cfe_tbl_msgstruct.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 171 of file default\_cfe\_tbl\_msgstruct.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 173 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.104 CFE\_TBL\_File\_Hdr Struct Reference

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

```
#include <default_cfe_tbl_extern_typedefs.h>
```

## Data Fields

- `uint32 Reserved`
- `uint32 Offset`
- `uint32 NumBytes`
- `char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`

### 11.104.1 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.

#### Note

The Offset and NumBytes fields in the table header are to 32 bits for backward compatibility with existing CFE versions. This means that even on 64-bit CPUs, individual table files will be limited to 4GiB in size.

Definition at line 64 of file default\_cfe\_tbl\_extern\_typedefs.h.

### 11.104.2 Field Documentation

#### 11.104.2.1 NumBytes `uint32 CFE_TBL_File_Hdr::NumBytes`

Number of bytes to load into table

Definition at line 68 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 11.104.2.2 Offset `uint32 CFE_TBL_File_Hdr::Offset`

Byte Offset at which load should commence

Definition at line 67 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 11.104.2.3 Reserved `uint32 CFE_TBL_File_Hdr::Reserved`

Future Use: NumTblSegments in File?

Definition at line 66 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 11.104.2.4 TableName `char CFE_TBL_File_Hdr::TableName[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`

Fully qualified name of table to load

Definition at line 69 of file default\_cfe\_tbl\_extern\_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_extern\_typedefs.h

## 11.105 CFE\_TBL\_FileDef Struct Reference

Table File summary object.

```
#include <cfe_tbl_filedef.h>
```

## Data Fields

- char **ObjectName** [64]  
*Name of instantiated variable that contains desired table image.*
- char **TableName** [[CFE\\_MISSION\\_TBL\\_MAX\\_FULL\\_NAME\\_LEN](#)]  
*Name of Table as defined onboard.*
- char **Description** [[CFE\\_FS\\_HDR\\_DESC\\_MAX\\_LEN](#)]  
*Description of table image that is included in cFE File Header.*
- char **TgtFilename** [[CFE\\_MISSION\\_MAX\\_FILE\\_LEN](#)]  
*Default filename to be used for output of elf2cfetbl utility.*
- **uint32 ObjectSize**  
*Size, in bytes, of instantiated object.*

### 11.105.1 Detailed Description

Table File summary object.

The definition of the file definition metadata that can be used by external tools (e.g. elf2cfetbl) to generate CFE table data files.

Definition at line 58 of file `cfe_tbl_filedef.h`.

### 11.105.2 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.

This is a free-form text string that can be any meaningful value

Definition at line 94 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.

##### Note

For consistency and future compatibility with auto-generated table files and table definitions, the "ObjectName" should match the table struct typedef name without the "\_t" suffix. For example, the limit checker action table (ADT) is defined by a type called "LC\_ADT\_t", the ObjectName should be "LC\_ADT".

This naming convention allows the type name to be inferred from the ObjectName (and vice-versa) without having to directly specify both the type name and object name here.

Although the traditional elf2cfetbl tool does not currently do any type checking, future tool versions may add more robust type verification and therefore need to know the type name as well as the object name.

Definition at line 76 of file `cfe_tbl_filedef.h`.

#### 11.105.2.3 **ObjectSize** `uint32 CFE_TBL_FileDef::ObjectSize`

Size, in bytes, of instantiated object.

This may be used by tools to check for consistency between the actual defined table size and the expected table size.

This is set automatically via the `CFE_TBL_FILEDEF` macro.

Definition at line 112 of file `cfe_tbl_filedef.h`.

**11.105.2.4 TableName** char CFE\_TBL\_FileDef::TableName[CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN]

Name of Table as defined onboard.

This should be in the form of "APP\_NAME.TABLE\_NAME" where APP\_NAME matches what the app is named at runtime (the 4th column of cfe\_es\_startup.scr) and TABLE\_NAME matches the 2nd parameter of the call to [CFE\\_TBL\\_Register\(\)](#). Preferably the TABLE\_NAME should also match the ObjectName here in this structure, although this is not strictly required, it helps keep things consistent.

Definition at line 87 of file [cfe\\_tbl\\_filedef.h](#).

**11.105.2.5 TgtFilename** char CFE\_TBL\_FileDef::TgtFilename[CFE\_MISSION\_MAX\_FILE\_LEN]

Default filename to be used for output of elf2cfetbl utility.

This must match the expected table file name, which is the name of the source file but the ".c" extension replaced with ".tbl". This is the filename only - do not include a directory/path name here, it can be copied to any runtime directory on the target by external scripts, but should not be renamed.

Definition at line 104 of file [cfe\\_tbl\\_filedef.h](#).

The documentation for this struct was generated from the following file:

- [cfe/modules/core\\_api/fsw/inc/cfe\\_tbl\\_filedef.h](#)

**11.106 CFE\_TBL\_HousekeepingTlm Struct Reference**

```
#include <default_cfe_tbl_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_TelemetryHeader\\_t TelemetryHeader](#)  
*Telemetry header.*
- [CFE\\_TBL\\_HousekeepingTlm\\_Payload\\_t Payload](#)  
*Telemetry payload.*

**11.106.1 Detailed Description**

Definition at line 348 of file [default\\_cfe\\_tbl\\_msgstruct.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 351 of file [default\\_cfe\\_tbl\\_msgstruct.h](#).

**11.106.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_TBL\_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 350 of file [default\\_cfe\\_tbl\\_msgstruct.h](#).

The documentation for this struct was generated from the following file:

- [cfe/modules/tbl/config/default\\_cfe\\_tbl\\_msgstruct.h](#)

**11.107 CFE\_TBL\_HousekeepingTlm\_Payload Struct Reference**

```
#include <default_cfe_tbl_msgstruct.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 289 of file default\_cfe\_tbl\_msgstruct.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 316 of file default\_cfe\_tbl\_msgstruct.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 332 of file default\_cfe\_tbl\_msgstruct.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 294 of file default\_cfe\_tbl\_msgstruct.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 296 of file default\_cfe\_tbl\_msgstruct.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 322 of file default\_cfe\_tbl\_msgstruct.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 342 of file default\_cfe\_tbl\_msgstruct.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 340 of file default\_cfe\_tbl\_msgstruct.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 344 of file default\_cfe\_tbl\_msgstruct.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 338 of file default\_cfe\_tbl\_msgstruct.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 336 of file default\_cfe\_tbl\_msgstruct.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 312 of file default\_cfe\_tbl\_msgstruct.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 314 of file default\_cfe\_tbl\_msgstruct.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 318 of file default\_cfe\_tbl\_msgstruct.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 334 of file default\_cfe\_tbl\_msgstruct.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 330 of file `default_cfe_tbl_msgstruct.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 304 of file `default_cfe_tbl_msgstruct.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 302 of file `default_cfe_tbl_msgstruct.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 324 of file `default_cfe_tbl_msgstruct.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 320 of file `default_cfe_tbl_msgstruct.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 310 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

## 11.108 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`.  
Referenced by `CS_ComputeTables()`.

**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 <default_cfe_tbl_msgstruct.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 82 of file `default_cfe_tbl_msgstruct.h`.

**11.109.2 Field Documentation****11.109.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_LoadCmd::CommandHeader**

Command header.

Definition at line 84 of file `default_cfe_tbl_msgstruct.h`.

**11.109.2.2 Payload [CFE\\_TBL\\_LoadCmd\\_Payload\\_t](#) CFE\_TBL\_LoadCmd::Payload**

Command payload.

Definition at line 85 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

**11.110 CFE\_TBL\_LoadCmd\_Payload Struct Reference**

Load Table Command Payload.

```
#include <default_cfe_tbl_msgstruct.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 74 of file `default_cfe_tbl_msgstruct.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 76 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

## 11.111 CFE\_TBL\_NoArgsCmd Struct Reference

Generic "no arguments" command.

```
#include <default_cfe_tbl_msgstruct.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 54 of file default\_cfe\_tbl\_msgstruct.h.

### 11.111.2 Field Documentation

#### 11.111.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_NoArgsCmd::CommandHeader

Command header.

Definition at line 58 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.112 CFE\_TBL\_NotifyCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)

*Command header.*

- [CFE\\_TBL\\_NotifyCmd\\_Payload\\_t Payload](#)

*Command payload.*

### 11.112.1 Detailed Description

/brief Table Management Notification Command

Definition at line 276 of file default\_cfe\_tbl\_msgstruct.h.

### 11.112.2 Field Documentation

**11.112.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_NotifyCmd::CommandHeader  
Command header.  
Definition at line 278 of file default\_cfe\_tbl\_msgstruct.h.

**11.112.2.2 Payload** [CFE\\_TBL\\_NotifyCmd\\_Payload\\_t](#) CFE\_TBL\_NotifyCmd::Payload  
Command payload.  
Definition at line 279 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.113 CFE\_TBL\_NotifyCmd\_Payload Struct Reference

Table Management Notification Command Payload.  
`#include <default_cfe_tbl_msgstruct.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 268 of file default\_cfe\_tbl\_msgstruct.h.

### 11.113.2 Field Documentation

**11.113.2.1 Parameter** [uint32](#) CFE\_TBL\_NotifyCmd\_Payload::Parameter  
Application specified command parameter.  
Definition at line 270 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.114 CFE\_TBL\_SendRegistryCmd Struct Reference

Send Table Registry Command.  
`#include <default_cfe_tbl_msgstruct.h>`

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CFE\\_TBL\\_SendRegistryCmd\\_Payload\\_t Payload](#)  
*Command payload.*

### 11.114.1 Detailed Description

Send Table Registry Command.

Definition at line 205 of file default\_cfe\_tbl\_msgstruct.h.

### 11.114.2 Field Documentation

**11.114.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_SendRegistryCmd::CommandHeader  
Command header.

Definition at line 207 of file default\_cfe\_tbl\_msgstruct.h.

**11.114.2.2 Payload** [CFE\\_TBL\\_SendRegistryCmd\\_Payload\\_t](#) CFE\_TBL\_SendRegistryCmd::Payload  
Command payload.

Definition at line 208 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.115 CFE\_TBL\_SendRegistryCmd\_Payload Struct Reference

Send Table Registry Command Payload.

```
#include <default_cfe_tbl_msgstruct.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 193 of file default\_cfe\_tbl\_msgstruct.h.

### 11.115.2 Field Documentation

**11.115.2.1 TableName** char CFE\_TBL\_SendRegistryCmd\_Payload::TableName [[CFE\\_MISSION\\_TBL\\_MAX\\_FULL\\_NAME\\_LEN](#)]  
Full Name of Table whose registry entry is to be telemetered.

ASCII string containing full table name identifier of table whose registry entry is to be telemetered via  
[CFE\\_TBL\\_TableRegistryTlm\\_t](#)

Definition at line 195 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.116 CFE\_TBL\_TableRegistryTlm Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_TelemetryHeader\\_t](#) TelemetryHeader  
*Telemetry header.*
- [CFE\\_TBL\\_TblRegPacket\\_Payload\\_t](#) Payload  
*Telemetry payload.*

**11.116.1 Detailed Description**

Definition at line 395 of file `default_cfe_tbl_msgstruct.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 398 of file `default_cfe_tbl_msgstruct.h`.

**11.116.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_TBL\_TableRegistryTlm::TelemetryHeader  
Telemetry header.

Definition at line 397 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/[default\\_cfe\\_tbl\\_msgstruct.h](#)

**11.117 CFE\_TBL\_TblRegPacket\_Payload Struct Reference**

```
#include <default_cfe_tbl_msgstruct.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 357 of file default\_cfe\_tbl\_msgstruct.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 363 of file default\_cfe\_tbl\_msgstruct.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 391 of file default\_cfe\_tbl\_msgstruct.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 361 of file default\_cfe\_tbl\_msgstruct.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 389 of file `default_cfe_tbl_msgstruct.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 381 of file `default_cfe_tbl_msgstruct.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 379 of file `default_cfe_tbl_msgstruct.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 371 of file `default_cfe_tbl_msgstruct.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 373 of file `default_cfe_tbl_msgstruct.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 365 of file `default_cfe_tbl_msgstruct.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 385 of file `default_cfe_tbl_msgstruct.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 377 of file default\_cfe\_tbl\_msgstruct.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 383 of file default\_cfe\_tbl\_msgstruct.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 387 of file default\_cfe\_tbl\_msgstruct.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 359 of file default\_cfe\_tbl\_msgstruct.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 375 of file default\_cfe\_tbl\_msgstruct.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 369 of file default\_cfe\_tbl\_msgstruct.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 367 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.118 CFE\_TBL\_ValidateCmd Struct Reference

Validate Table Command.

```
#include <default_cfe_tbl_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TBL\\_ValidateCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.118.1 Detailed Description

Validate Table Command.

Definition at line 139 of file default\_cfe\_tbl\_msgstruct.h.

### 11.118.2 Field Documentation

#### 11.118.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_ValidateCmd::CommandHeader

Command header.

Definition at line 141 of file default\_cfe\_tbl\_msgstruct.h.

#### 11.118.2.2 Payload [CFE\\_TBL\\_ValidateCmd\\_Payload\\_t](#) CFE\_TBL\_ValidateCmd::Payload

Command payload.

Definition at line 142 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.119 CFE\_TBL\_ValidateCmd\_Payload Struct Reference

Validate Table Command Payload.

```
#include <default_cfe_tbl_msgstruct.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 123 of file default\_cfe\_tbl\_msgstruct.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 125 of file `default_cfe_tbl_msgstruct.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 131 of file `default_cfe_tbl_msgstruct.h`.  
The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

## 11.120 CFE\_TIME\_DiagnosticTlm Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_TelemetryHeader\_t TelemetryHeader**  
*Telemetry header.*
- **CFE\_TIME\_DiagnosticTlm\_Payload\_t Payload**  
*Telemetry payload.*

### 11.120.1 Detailed Description

Definition at line 435 of file `default_cfe_time_msgstruct.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 438 of file `default_cfe_time_msgstruct.h`.

**11.120.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_TIME_DiagnosticTlm::TelemetryHeader`  
Telemetry header.  
Definition at line 437 of file `default_cfe_time_msgstruct.h`.  
The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

## 11.121 CFE\_TIME\_DiagnosticTlm\_Payload Struct Reference

```
#include <default_cfe_time_msgstruct.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 289 of file default\_cfe\_time\_msgstruct.h.

### 11.121.2 Field Documentation

**11.121.2.1 AtToneDelay** `CFE_TIME_SysTime_t` `CFE_TIME_DiagnosticTlm_Payload::AtToneDelay`  
Adjustment for slow tone detection.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DLlatentS`, `$sc_$cpu_TIME_DLlatentSs`

Definition at line 298 of file `default_cfe_time_msgstruct.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_DTVlaidS`, `$sc_$cpu_TIME_DTVlaidSs`

Definition at line 300 of file `default_cfe_time_msgstruct.h`.

**11.121.2.3 AtToneLeapSeconds** `int16` `CFE_TIME_DiagnosticTlm_Payload::AtToneLeapSeconds`  
Leap Seconds at time of tone.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DLepS`

Definition at line 303 of file `default_cfe_time_msgstruct.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 294 of file `default_cfe_time_msgstruct.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 296 of file `default_cfe_time_msgstruct.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 327 of file `default_cfe_time_msgstruct.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 325 of file `default_cfe_time_msgstruct.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 331 of file default\_cfe\_time\_msgstruct.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 329 of file default\_cfe\_time\_msgstruct.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 305 of file default\_cfe\_time\_msgstruct.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 341 of file default\_cfe\_time\_msgstruct.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_DLCLALS, $sc_$cpu_TIME_DLCLASS`

Definition at line 313 of file default\_cfe\_time\_msgstruct.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 315 of file default\_cfe\_time\_msgstruct.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 317 of file default\_cfe\_time\_msgstruct.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 319 of file default\_cfe\_time\_msgstruct.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 431 of file default\_cfe\_time\_msgstruct.h.

**11.121.2.17 DelayDirection** `int16` `CFE_TIME_DiagnosticTlm_Payload::DelayDirection`  
Client latency adjustment direction.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_DLlatentDir

Definition at line 351 of file default\_cfe\_time\_msgstruct.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 335 of file default\_cfe\_time\_msgstruct.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 389 of file default\_cfe\_time\_msgstruct.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 391 of file default\_cfe\_time\_msgstruct.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 411 of file default\_cfe\_time\_msgstruct.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 417 of file default\_cfe\_time\_msgstruct.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 409 of file default\_cfe\_time\_msgstruct.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 359 of file default\_cfe\_time\_msgstruct.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 349 of file default\_cfe\_time\_msgstruct.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 357 of file default\_cfe\_time\_msgstruct.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 347 of file default\_cfe\_time\_msgstruct.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 333 of file default\_cfe\_time\_msgstruct.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 311 of file `default_cfe_time_msgstruct.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 379 of file `default_cfe_time_msgstruct.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 367 of file `default_cfe_time_msgstruct.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 381 of file `default_cfe_time_msgstruct.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 383 of file `default_cfe_time_msgstruct.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 373 of file `default_cfe_time_msgstruct.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 375 of file `default_cfe_time_msgstruct.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 423 of file `default_cfe_time_msgstruct.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 377 of file `default_cfe_time_msgstruct.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 365 of file `default_cfe_time_msgstruct.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 385 of file `default_cfe_time_msgstruct.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 425 of file `default_cfe_time_msgstruct.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 387 of file `default_cfe_time_msgstruct.h`.

**11.121.2.42 VirtualMET** `uint32` `CFE_TIME_DiagnosticTlm_Payload::VirtualMET`  
Software MET.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DLLogicalMET`

Definition at line 397 of file `default_cfe_time_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

## 11.122 CFE\_TIME\_HousekeepingTlm Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_TelemetryHeader\\_t TelemetryHeader](#)  
*Telemetry header.*
- [CFE\\_TIME\\_HousekeepingTlm\\_Payload\\_t Payload](#)  
*Telemetry payload.*

### 11.122.1 Detailed Description

Definition at line 278 of file default\_cfe\_time\_msgstruct.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 281 of file default\_cfe\_time\_msgstruct.h.

**11.122.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_TIME\_HousekeepingTlm::TelemetryHeader  
Telemetry header.

Definition at line 280 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/[default\\_cfe\\_time\\_msgstruct.h](#)

## 11.123 CFE\_TIME\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_time_msgstruct.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](#)

- **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 220 of file default\_cfe\_time\_msgstruct.h.

### 11.123.2 Field Documentation

**11.123.2.1 ClockStateAPI** `CFE_TIME_ClockState_Enum_t` `CFE_TIME_HousekeepingTlm_Payload::ClockState` ← API State.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_DAPIState

Definition at line 235 of file default\_cfe\_time\_msgstruct.h.

**11.123.2.2 ClockStateFlags** `uint16` `CFE_TIME_HousekeepingTlm_Payload::ClockStateFlags` State Flags.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_StateFlag, \$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 233 of file default\_cfe\_time\_msgstruct.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 225 of file default\_cfe\_time\_msgstruct.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 227 of file default\_cfe\_time\_msgstruct.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 241 of file default\_cfe\_time\_msgstruct.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 261 of file default\_cfe\_time\_msgstruct.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 271 of file default\_cfe\_time\_msgstruct.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 247 of file default\_cfe\_time\_msgstruct.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 252 of file default\_cfe\_time\_msgstruct.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 263 of file default\_cfe\_time\_msgstruct.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 273 of file default\_cfe\_time\_msgstruct.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 249 of file default\_cfe\_time\_msgstruct.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 254 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.124 CFE\_TIME\_LeapsCmd\_Payload Struct Reference

Set leap seconds command payload.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `int16 LeapSeconds`

#### 11.124.1 Detailed Description

Set leap seconds command payload.

Definition at line 65 of file default\_cfe\_time\_msgstruct.h.

#### 11.124.2 Field Documentation

**11.124.2.1 LeapSeconds** `int16` `CFE_TIME_LeapsCmd_Payload::LeapSeconds`  
Definition at line 67 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.125 CFE\_TIME\_NoArgsCmd Struct Reference

Generic no argument command.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

#### 11.125.1 Detailed Description

Generic no argument command.

Definition at line 44 of file default\_cfe\_time\_msgstruct.h.

## 11.125.2 Field Documentation

### 11.125.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_NoArgsCmd::CommandHeader

Command header.

Definition at line 48 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.126 CFE\_TIME\_OneHzAdjustmentCmd Struct Reference

Generic seconds, subseconds adjustment command.

```
#include <default_cfe_time_msgstruct.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 181 of file default\_cfe\_time\_msgstruct.h.

## 11.126.2 Field Documentation

### 11.126.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_OneHzAdjustmentCmd::CommandHeader

Command header.

Definition at line 183 of file default\_cfe\_time\_msgstruct.h.

### 11.126.2.2 Payload [CFE\\_TIME\\_OneHzAdjustmentCmd\\_Payload\\_t](#) CFE\_TIME\_OneHzAdjustmentCmd::Payload

Command payload.

Definition at line 184 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.127 CFE\_TIME\_OneHzAdjustmentCmd\_Payload Struct Reference

Generic seconds, subseconds command payload.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- uint32 Seconds
- uint32 Subseconds

### 11.127.1 Detailed Description

Generic seconds, subseconds command payload.

Definition at line 172 of file default\_cfe\_time\_msgstruct.h.

### 11.127.2 Field Documentation

#### 11.127.2.1 Seconds `uint32` CFE\_TIME\_OneHzAdjustmentCmd\_Payload::Seconds

Definition at line 174 of file default\_cfe\_time\_msgstruct.h.

#### 11.127.2.2 Subseconds `uint32` CFE\_TIME\_OneHzAdjustmentCmd\_Payload::Subseconds

Definition at line 175 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.128 CFE\_TIME\_SetLeapSecondsCmd Struct Reference

Set leap seconds command.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader
  - Command header.*
- [CFE\\_TIME\\_LeapsCmd\\_Payload\\_t](#) Payload
  - Command payload.*

### 11.128.1 Detailed Description

Set leap seconds command.

Definition at line 73 of file default\_cfe\_time\_msgstruct.h.

### 11.128.2 Field Documentation

#### 11.128.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CFE\_TIME\_SetLeapSecondsCmd::CommandHeader

Command header.

Definition at line 75 of file default\_cfe\_time\_msgstruct.h.

#### 11.128.2.2 Payload `CFE_TIME_LeapsCmd_Payload_t` CFE\_TIME\_SetLeapSecondsCmd::Payload

Command payload.

Definition at line 76 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.129 CFE\_TIME\_SetSignalCmd Struct Reference

Set tone signal source command.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_SignalCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.129.1 Detailed Description

Set tone signal source command.

Definition at line 132 of file default\_cfe\_time\_msgstruct.h.

### 11.129.2 Field Documentation

#### 11.129.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetSignalCmd::CommandHeader

Command header.

Definition at line 134 of file default\_cfe\_time\_msgstruct.h.

#### 11.129.2.2 Payload [CFE\\_TIME\\_SignalCmd\\_Payload\\_t](#) CFE\_TIME\_SetSignalCmd::Payload

Command payload.

Definition at line 135 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.130 CFE\_TIME\_SetSourceCmd Struct Reference

Set time data source command.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_SourceCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.130.1 Detailed Description

Set time data source command.

Definition at line 113 of file default\_cfe\_time\_msgstruct.h.

### 11.130.2 Field Documentation

**11.130.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetSourceCmd::CommandHeader  
Command header.

Definition at line 115 of file default\_cfe\_time\_msgstruct.h.

**11.130.2.2 Payload** [CFE\\_TIME\\_SourceCmd\\_Payload\\_t](#) CFE\_TIME\_SetSourceCmd::Payload  
Command payload.

Definition at line 116 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.131 CFE\_TIME\_SetStateCmd Struct Reference

Set clock state command.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_StateCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.131.1 Detailed Description

Set clock state command.

Definition at line 94 of file default\_cfe\_time\_msgstruct.h.

### 11.131.2 Field Documentation

**11.131.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetStateCmd::CommandHeader  
Command header.

Definition at line 96 of file default\_cfe\_time\_msgstruct.h.

**11.131.2.2 Payload** [CFE\\_TIME\\_StateCmd\\_Payload\\_t](#) CFE\_TIME\_SetStateCmd::Payload

Command payload.

Definition at line 97 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.132 CFE\_TIME\_SignalCmd\_Payload Struct Reference

Set tone signal source command payload.

```
#include <default_cfe_time_msgstruct.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 122 of file default\_cfe\_time\_msgstruct.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 124 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.133 CFE\_TIME\_SourceCmd\_Payload Struct Reference

Set time data source command payload.

```
#include <default_cfe_time_msgstruct.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 103 of file default\_cfe\_time\_msgstruct.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 105 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.134 CFE\_TIME\_StateCmd\_Payload Struct Reference

Set clock state command payload.

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_TIME_ClockState_Enum_t ClockState`

`CFE_TIME_ClockState_INVALID`=Spacecraft time has not been accurately set, `CFE_TIME_ClockState_VALID`=Spacecraft clock has been accurately set, `CFE_TIME_ClockState_FLYWHEEL`=Force into FLYWHEEL mode

### 11.134.1 Detailed Description

Set clock state command payload.

Definition at line 82 of file default\_cfe\_time\_msgstruct.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 84 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.135 CFE\_TIME\_SysTime Struct Reference

Data structure used to hold system time values.

```
#include <default_cfe_time_extern_typedefs.h>
```

### Data Fields

- `uint32 Seconds`  
*Number of seconds since epoch.*
- `uint32 Subseconds`  
*Number of subseconds since epoch (LSB =  $2^{-32}$  seconds)*

### 11.135.1 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 41 of file default\_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 43 of file default\_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 44 of file default\_cfe\_time\_extern\_typedefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_extern\_typedefs.h

## 11.136 CFE\_TIME\_TimeCmd Struct Reference

Generic seconds, microseconds argument command.

```
#include <default_cfe_time_msgstruct.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 150 of file default\_cfe\_time\_msgstruct.h.

### 11.136.2 Field Documentation

#### 11.136.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CFE\_TIME\_TimeCmd::CommandHeader

Command header.

Definition at line 152 of file default\_cfe\_time\_msgstruct.h.

#### 11.136.2.2 Payload `CFE_TIME_TimeCmd_Payload_t` CFE\_TIME\_TimeCmd::Payload

Command payload.

Definition at line 153 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.137 CFE\_TIME\_TimeCmd\_Payload Struct Reference

Generic seconds, microseconds command payload.

```
#include <default_cfe_time_msgstruct.h>
```

**Data Fields**

- `uint32 Seconds`
- `uint32 MicroSeconds`

**11.137.1 Detailed Description**

Generic seconds, microseconds command payload.

Definition at line 141 of file default\_cfe\_time\_msgstruct.h.

**11.137.2 Field Documentation****11.137.2.1 MicroSeconds `uint32 CFE_TIME_TimeCmd_Payload::MicroSeconds`**

Definition at line 144 of file default\_cfe\_time\_msgstruct.h.

**11.137.2.2 Seconds `uint32 CFE_TIME_TimeCmd_Payload::Seconds`**

Definition at line 143 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

**11.138 CFE\_TIME\_ToneDataCmd Struct Reference**

Time at tone data command.

```
#include <default_cfe_time_msgstruct.h>
```

**Data Fields**

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_TIME_ToneDataCmd_Payload_t Payload`  
*Command payload.*

**11.138.1 Detailed Description**

Time at tone data command.

Definition at line 209 of file default\_cfe\_time\_msgstruct.h.

**11.138.2 Field Documentation****11.138.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_TIME_ToneDataCmd::CommandHeader`**

Command header.

Definition at line 211 of file default\_cfe\_time\_msgstruct.h.

**11.138.2.2 Payload `CFE_TIME_ToneDataCmd_Payload_t CFE_TIME_ToneDataCmd::Payload`**

Command payload.

Definition at line 212 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.139 CFE\_TIME\_ToneDataCmd\_Payload Struct Reference

Time at tone data command payload.

```
#include <default_cfe_time_msgstruct.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 198 of file default\_cfe\_time\_msgstruct.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 202 of file default\_cfe\_time\_msgstruct.h.

#### 11.139.2.2 AtToneMET `CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneMET`

MET at time of tone.

Definition at line 200 of file default\_cfe\_time\_msgstruct.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 203 of file default\_cfe\_time\_msgstruct.h.

#### 11.139.2.4 AtToneSTCF `CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneSTCF`

STCF at time of tone.

Definition at line 201 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.140 CS\_AppData\_t Struct Reference

CS global data structure.

```
#include <cs_app.h>
```

## Data Fields

- **CS\_HkPacket\_t HkPacket**  
*Housekeeping telemetry packet.*
- **char PipeName [CS\_CMD\_PIPE\_NAME\_LEN]**  
*Command pipe name.*
- **uint16 PipeDepth**  
*Command pipe depth.*
- **uint16 ChildTaskTable**  
*Table for the child task to process.*
- **uint16 ChildTaskEntryID**  
*Entry in table for child task to process.*
- **CFE\_ES\_TaskId\_t ChildTaskID**  
*Task ID for the child task.*
- **uint32 MaxBytesPerCycle**  
*Max number of bytes to process in a cycle.*
- **uint32 RunStatus**  
*Application run status.*
- **CS\_Res\_EepromMemory\_Table\_Entry\_t \* RecomputeEepromMemoryEntryPtr**  
*Pointer to an entry to recompute in the EEPROM or Memory table.*
- **CS\_Res\_App\_Table\_Entry\_t \* RecomputeAppEntryPtr**  
*Pointer to an entry to recompute in the Application table.*
- **CS\_Res\_Tables\_Table\_Entry\_t \* RecomputeTablesEntryPtr**  
*Pointer to an entry to recompute in the Tables table.*
- **CFE\_SB\_Pipeld\_t CmdPipe**  
*Command pipe ID.*
- **CFE\_TBL\_Handle\_t DefEepromTableHandle**  
*Handle to the EEPROM definition table.*
- **CFE\_TBL\_Handle\_t ResEepromTableHandle**  
*Handle to the EEPROM results table.*
- **CFE\_TBL\_Handle\_t DefMemoryTableHandle**  
*Handle to the Memory definition table.*
- **CFE\_TBL\_Handle\_t ResMemoryTableHandle**  
*Handle to the Memory results table.*
- **CFE\_TBL\_Handle\_t DefTablesTableHandle**  
*Handle to the Tables definition table.*
- **CFE\_TBL\_Handle\_t ResTablesTableHandle**  
*Handle to the Tables results table.*
- **CFE\_TBL\_Handle\_t DefAppTableHandle**  
*Handle to the Apps definition table.*
- **CFE\_TBL\_Handle\_t ResAppTableHandle**  
*Handle to the Apps results table.*
- **CS\_Def\_EepromMemory\_Table\_Entry\_t \* DefEepromTblPtr**  
*Pointer to the EEPROM definition table.*
- **CS\_Res\_EepromMemory\_Table\_Entry\_t \* ResEepromTblPtr**  
*Pointer to the EEPROM results table.*
- **CS\_Def\_EepromMemory\_Table\_Entry\_t \* DefMemoryTblPtr**  
*Pointer to the Memory definition table.*

- `CS_Res_EepromMemory_Table_Entry_t * ResMemoryTblPtr`  
*Pointer to the Memory results table.*
- `CS_Def_Tables_Table_Entry_t * DefTablesTblPtr`  
*Pointer to the Tables definition table.*
- `CS_Res_Tables_Table_Entry_t * ResTablesTblPtr`  
*Pointer to the Tables results table.*
- `CS_Def_App_Table_Entry_t * DefAppTblPtr`  
*Pointer to the Apps definition table.*
- `CS_Res_App_Table_Entry_t * ResAppTblPtr`  
*Pointer to the Apps results table.*
- `CS_Res_EepromMemory_Table_Entry_t OSCodeSeg`  
*OS code segment 'table'.*
- `CS_Res_EepromMemory_Table_Entry_t CfeCoreCodeSeg`  
*cFE core code segment 'table'*
- `CS_Def_EepromMemory_Table_Entry_t DefaultEepromDefTable [CS_MAX_NUM_EEPROM_TABLE_ENTRIES]`  
*Default EEPROM definition table.*
- `CS_Def_EepromMemory_Table_Entry_t DefaultMemoryDefTable [CS_MAX_NUM_MEMORY_TABLE_ENTRIES]`  
*Default Memory definition table.*
- `CS_Def_Tables_Table_Entry_t DefaultTablesDefTable [CS_MAX_NUM_TABLES_TABLE_ENTRIES]`  
*Default Tables definition table.*
- `CS_Def_App_Table_Entry_t DefaultAppDefTable [CS_MAX_NUM_APP_TABLE_ENTRIES]`  
*Default Apps definition table.*
- `CS_Res_Tables_Table_Entry_t * EepResTablesTblPtr`  
*CS results entry for the CS eeprom.*
- `CS_Res_Tables_Table_Entry_t * MemResTablesTblPtr`  
*CS results entry for the CS memory.*
- `CS_Res_Tables_Table_Entry_t * AppResTablesTblPtr`  
*CS results entry for the CS apps.*
- `CS_Res_Tables_Table_Entry_t * TblResTablesTblPtr`  
*CS results table entry for the CS tables.*
- `CFE_ES_CDSHandle_t DataStoreHandle`  
*Handle to critical data store created by CS.*

### 11.140.1 Detailed Description

CS global data structure.

Definition at line 106 of file cs\_app.h.

### 11.140.2 Field Documentation

#### 11.140.2.1 AppResTablesTblPtr `CS_Res_Tables_Table_Entry_t* CS_AppData_t::AppResTablesTblPtr`

CS results entry for the CS apps.

Definition at line 176 of file cs\_app.h.

Referenced by `CS_DisableNameAppCmd()`, `CS_EnableNameAppCmd()`, `CS_ProcessNewAppDefinitionTable()`, `CS_ProcessNewTablesDefinitionTable()`, and `CS_RecomputeAppChildTask()`.

**11.140.2.2 CfeCoreCodeSeg** `CS_Res_EepromMemory_Table_Entry_t` `CS_AppData_t::CfeCoreCodeSeg`  
cFE core code segment 'table'  
Definition at line 160 of file `cs_app.h`.  
Referenced by `CS_BackgroundCfeCore()`, `CS_InitSegments()`, `CS_RecomputeBaselineCfeCoreCmd()`, `CS_ReportBaselineCfeCoreCmd()`, and `CS_ZeroCfeCoreTempValues()`.

**11.140.2.3 ChildTaskEntryID** `uint16` `CS_AppData_t::ChildTaskEntryID`  
Entry in table for child task to process.  
Definition at line 118 of file `cs_app.h`.  
Referenced by `CS_RecomputeBaselineCfeCoreCmd()`, `CS_RecomputeBaselineEepromCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeBaselineOSCmd()`, and `CS_RecomputeEepromMemoryChildTask()`.

**11.140.2.4 ChildTaskID** `CFE_ES_TaskId_t` `CS_AppData_t::ChildTaskID`  
Task ID for the child task.  
Definition at line 119 of file `cs_app.h`.  
Referenced by `CS_CancelOneShotCmd()`, `CS_OneShotChildTask()`, and `CS_OneShotCmd()`.

**11.140.2.5 ChildTaskTable** `uint16` `CS_AppData_t::ChildTaskTable`  
Table for the child task to process.  
Definition at line 117 of file `cs_app.h`.  
Referenced by `CS_HandleRoutineTableUpdates()`, `CS_RecomputeBaselineAppCmd()`, `CS_RecomputeBaselineCfeCoreCmd()`, `CS_RecomputeBaselineEepromCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeBaselineOSCmd()`, `CS_RecomputeBaselineTablesCmd()`, and `CS_RecomputeEepromMemoryChildTask()`.

**11.140.2.6 CmdPipe** `CFE_SB_PipeId_t` `CS_AppData_t::CmdPipe`  
Command pipe ID.  
Definition at line 133 of file `cs_app.h`.  
Referenced by `CS_AppMain()`, and `CS_SbInit()`.

**11.140.2.7 DataStoreHandle** `CFE_ES_CDSHandle_t` `CS_AppData_t::DataStoreHandle`  
Handle to critical data store created by CS.  
Definition at line 180 of file `cs_app.h`.  
Referenced by `CS_CreateRestoreStatesFromCDS()`, and `CS_UpdateCDS()`.

**11.140.2.8 DefAppTableHandle** `CFE_TBL_Handle_t` `CS_AppData_t::DefAppTableHandle`  
Handle to the Apps definition table.  
Definition at line 144 of file `cs_app.h`.  
Referenced by `CS_DisableNameAppCmd()`, `CS_EnableNameAppCmd()`, `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, `CS_ProcessNewTablesDefinitionTable()`, and `CS_RecomputeAppChildTask()`.

**11.140.2.9 DefAppTblPtr** `CS_Def_App_Table_Entry_t*` `CS_AppData_t::DefAppTblPtr`  
Pointer to the Apps definition table.  
Definition at line 156 of file `cs_app.h`.  
Referenced by `CS_GetAppDefTblEntryByName()`, `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, and `CS_RecomputeAppChildTask()`.

**11.140.2.10 DefaultAppDefTable** `CS_Def_App_Table_Entry_t CS_AppData_t::DefaultAppDefTable[CS_MAX_NUM_APP_TABLE_ENTRIES]`  
Default Apps definition table.  
Definition at line 169 of file cs\_app.h.  
Referenced by CS\_InitAllTables(), and CS\_InitializeDefaultTables().

**11.140.2.11 DefaultEepromDefTable** `CS_Def_EepromMemory_Table_Entry_t CS_AppData_t::DefaultEepromDefTable[CS_MAX_NUM_EEPROM_TABLE_ENTRIES]`  
Default EEPROM definition table.  
Definition at line 163 of file cs\_app.h.  
Referenced by CS\_InitAllTables(), and CS\_InitializeDefaultTables().

**11.140.2.12 DefaultMemoryDefTable** `CS_Def_EepromMemory_Table_Entry_t CS_AppData_t::DefaultMemoryDefTable[CS_MAX_NUM_MEMORY_TABLE_ENTRIES]`  
Default Memory definition table.  
Definition at line 165 of file cs\_app.h.  
Referenced by CS\_InitAllTables(), and CS\_InitializeDefaultTables().

**11.140.2.13 DefaultTablesDefTable** `CS_Def_Tables_Table_Entry_t CS_AppData_t::DefaultTablesDefTable[CS_MAX_NUM_TABLES_TABLE_ENTRIES]`  
Default Tables definition table.  
Definition at line 167 of file cs\_app.h.  
Referenced by CS\_InitAllTables(), and CS\_InitializeDefaultTables().

**11.140.2.14 DefEepromTableHandle** `CFE_TBL_Handle_t CS_AppData_t::DefEepromTableHandle`  
Handle to the EEPROM definition table.  
Definition at line 135 of file cs\_app.h.  
Referenced by CS\_DisableEntryIDEEPROMCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), CS\_ProcessNewTablesDefinitionTable(), and CS\_RecomputeEEPROMMemoryChildTask().

**11.140.2.15 DefEepromTblPtr** `CS_Def_EepromMemory_Table_Entry_t* CS_AppData_t::DefEepromTblPtr`  
Pointer to the EEPROM definition table.  
Definition at line 147 of file cs\_app.h.  
Referenced by CS\_DisableEntryIDEEPROMCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), and CS\_RecomputeEEPROMMemoryChildTask().

**11.140.2.16 DefMemoryTableHandle** `CFE_TBL_Handle_t CS_AppData_t::DefMemoryTableHandle`  
Handle to the Memory definition table.  
Definition at line 138 of file cs\_app.h.  
Referenced by CS\_DisableEntryIDMEMORYCmd(), CS\_EnableEntryIDMEMORYCmd(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), CS\_ProcessNewTablesDefinitionTable(), and CS\_RecomputeEEPROMMemoryChildTask().

**11.140.2.17 DefMemoryTblPtr** `CS_Def_EepromMemory_Table_Entry_t* CS_AppData_t::DefMemoryTblPtr`  
Pointer to the Memory definition table.

Definition at line 150 of file cs\_app.h.

Referenced by CS\_DisableEntryIDMemoryCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), and CS\_RecomputeEepromMemoryChildTask().

#### 11.140.2.18 DefTablesTableHandle [CFE\\_TBL\\_Handle\\_t](#) CS\_AppData\_t::DefTablesTableHandle

Handle to the Tables definition table.

Definition at line 141 of file cs\_app.h.

Referenced by CS\_DisableNameTablesCmd(), CS\_EnableNameTablesCmd(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), CS\_ProcessNewTablesDefinitionTable(), and CS\_RecomputeTablesChildTask().

#### 11.140.2.19 DefTablesTblPtr [CS\\_Def\\_Tables\\_Table\\_Entry\\_t\\*](#) CS\_AppData\_t::DefTablesTblPtr

Pointer to the Tables definition table.

Definition at line 153 of file cs\_app.h.

Referenced by CS\_GetTableDefTblEntryByName(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), and CS\_RecomputeTablesChildTask().

#### 11.140.2.20 EepResTablesTblPtr [CS\\_Res\\_Tables\\_Table\\_Entry\\_t\\*](#) CS\_AppData\_t::EepResTablesTblPtr

CS results entry for the CS eeprom.

Definition at line 174 of file cs\_app.h.

Referenced by CS\_DisableEntryIDEEPROMCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_ProcessNewEEPROMMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), and CS\_RecomputeEEPROMMemoryChildTask().

#### 11.140.2.21 HkPacket [CS\\_HkPacket\\_t](#) CS\_AppData\_t::HkPacket

Housekeeping telemetry packet.

Definition at line 112 of file cs\_app.h.

Referenced by CS\_AppInit(), CS\_AppPipe(), CS\_BackgroundApp(), CS\_BackgroundCfeCore(), CS\_BackgroundCheckCycle(), CS\_BackgroundEEPROM(), CS\_BackgroundMemory(), CS\_BackgroundOS(), CS\_BackgroundTables(), CS\_CancelOneShotCmd(), CS\_CheckRecomputeOneshot(), CS\_CreateRestoreStatesFromCDS(), CS\_DisableAllCSCmd(), CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEEPROMCmd(), CS\_DisableEntryIDEEPROMCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_EnableAllCSCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEEPROMCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_EnableOSCmd(), CS\_EnableTablesCmd(), CS\_FindEnabledAppEntry(), CS\_FindEnabledEEPROMEntry(), CS\_FindEnabledMemoryEntry(), CS\_FindEnabledTablesEntry(), CS\_GetEntryIDEEPROMCmd(), CS\_GetEntryIDMemoryCmd(), CS\_GoToNextTable(), CS\_HandleRoutineTableUpdates(), CS\_HousekeepingCmd(), CS\_InitAllTables(), CS\_NoopCmd(), CS\_OneShotChildTask(), CS\_OneShotCmd(), CS\_ProcessCmd(), CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEEPROMMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), CS\_RecomputeAppChildTask(), CS\_RecomputeBaselineAppCmd(), CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEEPROMCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), CS\_RecomputeBaselineTablesCmd(), CS\_RecomputeEEPROMMemoryChildTask(), CS\_RecomputeTablesChildTask(), CS\_ReportBaselineAppCmd(), CS\_ReportBaselineCfeCoreCmd(), CS\_ReportBaselineEntryIDEEPROMCmd(), CS\_ReportBaselineEntryIDMemoryCmd(), CS\_ReportBaselineOSCmd(), CS\_ReportBaselineTablesCmd(), CS\_ResetCmd(), CS\_SblInit(), CS\_TableInit(), CS\_UpdateCDS(), and CS\_VerifyCmdLength().

#### 11.140.2.22 MaxBytesPerCycle [uint32](#) CS\_AppData\_t::MaxBytesPerCycle

Max number of bytes to process in a cycle.

Definition at line 121 of file cs\_app.h.

Referenced by CS\_AppInit(), CS\_ComputeApp(), CS\_ComputeEepromMemory(), CS\_ComputeTables(), and CS\_OneShotCmd().

#### **11.140.2.23 MemResTablesTblPtr** [CS\\_Res\\_Tables\\_Table\\_Entry\\_t\\*](#) CS\_AppData\_t::MemResTablesTblPtr

CS results entry for the CS memory.

Definition at line 175 of file cs\_app.h.

Referenced by CS\_DisableEntryIDMemoryCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), and CS\_RecomputeEepromMemoryChildTask().

#### **11.140.2.24 OSCodeSeg** [CS\\_Res\\_EepromMemory\\_Table\\_Entry\\_t](#) CS\_AppData\_t::OSCodeSeg

OS code segment 'table'.

Definition at line 159 of file cs\_app.h.

Referenced by CS\_BackgroundOS(), CS\_InitSegments(), CS\_RecomputeBaselineOSCmd(), CS\_ReportBaselineOSCmd(), and CS\_ZeroOSTempValues().

#### **11.140.2.25 PipeDepth** [uint16](#) CS\_AppData\_t::PipeDepth

Command pipe depth.

Definition at line 115 of file cs\_app.h.

Referenced by CS\_SblInit().

#### **11.140.2.26 PipeName** [char](#) CS\_AppData\_t::PipeName[[CS\\_CMD\\_PIPE\\_NAME\\_LEN](#)]

Command pipe name.

Definition at line 114 of file cs\_app.h.

Referenced by CS\_SblInit().

#### **11.140.2.27 RecomputeAppEntryPtr** [CS\\_Res\\_App\\_Table\\_Entry\\_t\\*](#) CS\_AppData\_t::RecomputeAppEntryPtr

Pointer to an entry to recompute in the Application table.

Definition at line 128 of file cs\_app.h.

Referenced by CS\_RecomputeAppChildTask(), and CS\_RecomputeBaselineAppCmd().

#### **11.140.2.28 RecomputeEepromMemoryEntryPtr** [CS\\_Res\\_EepromMemory\\_Table\\_Entry\\_t\\*](#) CS\_AppData\_t::RecomputeEepromMemoryEntryPtr

Pointer to an entry to recompute in the EEPROM or Memory table.

Definition at line 125 of file cs\_app.h.

Referenced by CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), and CS\_RecomputeEepromMemoryChildTask().

#### **11.140.2.29 RecomputeTablesEntryPtr** [CS\\_Res\\_Tables\\_Table\\_Entry\\_t\\*](#) CS\_AppData\_t::RecomputeTablesEntryPtr

Pointer to an entry to recompute in the Tables table.

Definition at line 130 of file cs\_app.h.

Referenced by CS\_RecomputeBaselineTablesCmd(), and CS\_RecomputeTablesChildTask().

**11.140.2.30 ResAppTableHandle** `CFE_TBL_Handle_t` `CS_AppData_t::ResAppTableHandle`  
Handle to the Apps results table.

Definition at line 145 of file `cs_app.h`.

Referenced by `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, and `CS_ProcessNewTablesDefinitionTable()`.

**11.140.2.31 ResAppTblPtr** `CS_Res_App_Table_Entry_t*` `CS_AppData_t::ResAppTblPtr`  
Pointer to the Apps results table.

Definition at line 157 of file `cs_app.h`.

Referenced by `CS_BackgroundApp()`, `CS_FindEnabledAppEntry()`, `CS_GetAppResTblEntryByName()`, `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, and `CS_ZeroAppTempValues()`.

**11.140.2.32 ResEepromTableHandle** `CFE_TBL_Handle_t` `CS_AppData_t::ResEepromTableHandle`  
Handle to the EEPROM results table.

Definition at line 136 of file `cs_app.h`.

Referenced by `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, and `CS_ProcessNewTablesDefinitionTable()`.

**11.140.2.33 ResEepromTblPtr** `CS_Res_EepromMemory_Table_Entry_t*` `CS_AppData_t::ResEepromTblPtr`  
Pointer to the EEPROM results table.

Definition at line 148 of file `cs_app.h`.

Referenced by `CS_BackgroundEeprom()`, `CS_DisableEntryIDEepromCmd()`, `CS_EnableEntryIDEepromCmd()`, `CS_FindEnabledEepromEntry()`, `CS_GetEntryIDEepromCmd()`, `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, `CS_RecomputeBaselineEepromCmd()`, `CS_ReportBaselineEntryIDEepromCmd()`, and `CS_ZeroEepromTempValues()`.

**11.140.2.34 ResMemoryTableHandle** `CFE_TBL_Handle_t` `CS_AppData_t::ResMemoryTableHandle`  
Handle to the Memory results table.

Definition at line 139 of file `cs_app.h`.

Referenced by `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, and `CS_ProcessNewTablesDefinitionTable()`.

**11.140.2.35 ResMemoryTblPtr** `CS_Res_EepromMemory_Table_Entry_t*` `CS_AppData_t::ResMemoryTblPtr`  
Pointer to the Memory results table.

Definition at line 151 of file `cs_app.h`.

Referenced by `CS_BackgroundMemory()`, `CS_DisableEntryIDMemoryCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_FindEnabledMemoryEntry()`, `CS_GetEntryIDMemoryCmd()`, `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_ReportBaselineEntryIDMemoryCmd()`, and `CS_ZeroMemoryTempValues()`.

**11.140.2.36 ResTablesTableHandle** `CFE_TBL_Handle_t` `CS_AppData_t::ResTablesTableHandle`  
Handle to the Tables results table.

Definition at line 142 of file `cs_app.h`.

Referenced by `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, and `CS_ProcessNewTablesDefinitionTable()`.

**11.140.2.37 ResTablesTblPtr** `CS_Res_Tables_Table_Entry_t*` `CS_AppData_t::ResTablesTblPtr`  
Pointer to the Tables results table.

Definition at line 154 of file `cs_app.h`.

Referenced by CS\_BackgroundTables(), CS\_FindEnabledTablesEntry(), CS\_GetTableResTblEntryByName(), CS\_HandleRoutineTableUpdates(), CS\_HandleTableUpdate(), CS\_InitAllTables(), and CS\_ZeroTablesTempValues().

#### 11.140.2.38 RunStatus `uint32 CS_AppData_t::RunStatus`

Application run status.

Definition at line 123 of file cs\_app.h.

Referenced by CS\_AppInit(), and CS\_AppMain().

#### 11.140.2.39 TblResTablesTblPtr `CS_Res_Tables_Table_Entry_t* CS_AppData_t::TblResTablesTblPtr`

CS results table entry for the CS tables.

Definition at line 177 of file cs\_app.h.

Referenced by CS\_DisableNameTablesCmd(), CS\_EnableNameTablesCmd(), CS\_ProcessNewTablesDefinitionTable(), and CS\_RecomputeTablesChildTask().

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/src/cs_app.h`

### 11.141 CS\_AppNameCmd\_Payload\_t Struct Reference

Payload for commanding by app name.

```
#include <cs_msg.h>
```

#### Data Fields

- `char Name [OS_MAX_API_NAME]`

*App name to perform a command on.*

#### 11.141.1 Detailed Description

Payload for commanding by app name.

Definition at line 113 of file cs\_msg.h.

#### 11.141.2 Field Documentation

##### 11.141.2.1 Name `char CS_AppNameCmd_Payload_t::Name [OS_MAX_API_NAME]`

App name to perform a command on.

Definition at line 115 of file cs\_msg.h.

Referenced by CS\_DisableNameAppCmd(), CS\_EnableNameAppCmd(), CS\_RecomputeBaselineAppCmd(), and CS\_ReportBaselineAppCmd().

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_msg.h`

### 11.142 CS\_AppNameCmd\_t Struct Reference

Command type for commanding by app name.

```
#include <cs_msg.h>
```

**Data Fields**

- CFE\_MSG\_CommandHeader\_t CmdHeader
- CS\_AppNameCmd\_Payload\_t Payload

**11.142.1 Detailed Description**

Command type for commanding by app name.

For command details see e [CS\\_ENABLE\\_NAME\\_APP\\_CC](#), [CS\\_DISABLE\\_NAME\\_APP\\_CC](#), [CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#), [CS\\_REPORT\\_BASELINE\\_APP\\_CC](#)

Definition at line 184 of file cs\_msg.h.

**11.142.2 Field Documentation****11.142.2.1 CmdHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CS\_AppNameCmd\_t::CmdHeader

Definition at line 186 of file cs\_msg.h.

**11.142.2.2 Payload** [CS\\_AppNameCmd\\_Payload\\_t](#) CS\_AppNameCmd\_t::Payload

Definition at line 187 of file cs\_msg.h.

Referenced by CS\_DisableNameAppCmd(), CS\_EnableNameAppCmd(), CS\_RecomputeBaselineAppCmd(), and CS\_ReportBaselineAppCmd().

The documentation for this struct was generated from the following file:

- apps/cs/fsw/inc/[cs\\_msg.h](#)

**11.143 CS\_Def\_App\_Table\_Entry\_t Struct Reference**

Data structure for the App definition table.

```
#include <cs_tbldefs.h>
```

**Data Fields**

- uint16 State  
*Uses the CS\_STATE\_... defines from above.*
- char Name [[OS\\_MAX\\_API\\_NAME](#)]  
*name of the app*

**11.143.1 Detailed Description**

Data structure for the App definition table.

Definition at line 104 of file cs\_tbldefs.h.

**11.143.2 Field Documentation****11.143.2.1 Name** char CS\_Def\_App\_Table\_Entry\_t::Name [[OS\\_MAX\\_API\\_NAME](#)]

name of the app

Definition at line 107 of file cs\_tbldefs.h.

Referenced by CS\_GetAppDefTblEntryByName(), CS\_InitializeDefaultTables(), CS\_ProcessNewAppDefinitionTable(), CS\_RecomputeAppChildTask(), and CS\_ValidateAppChecksumDefinitionTable().

**11.143.2.2 State** `uint16 CS_Def_App_Table_Entry_t::State`

Uses the CS\_STATE\_... defines from above.

Definition at line 106 of file cs\_tbldefs.h.

Referenced by CS\_DisableNameAppCmd(), CS\_EnableNameAppCmd(), CS\_GetAppDefTblEntryByName(), CS\_InitializeDefaultTables(), CS\_ProcessNewAppDefinitionTable(), CS\_RecomputeAppChildTask(), and CS\_ValidateAppChecksumDefinitionTable().

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_tbldefs.h`

## 11.144 CS\_Def\_EepromMemory\_Table\_Entry\_t Struct Reference

Data structure for the EEPROM or Memory definition table.

```
#include <cs_tbldefs.h>
```

### Data Fields

- `cpuaddr StartAddress`  
*The Start address to Checksum.*
- `uint16 State`  
*Uses the CS\_STATE\_... defines from above.*
- `uint16 Filler16`
- `uint32 NumBytesToChecksum`  
*Padding.*

### 11.144.1 Detailed Description

Data structure for the EEPROM or Memory definition table.

Definition at line 69 of file cs\_tbldefs.h.

### 11.144.2 Field Documentation

#### 11.144.2.1 Filler16 `uint16 CS_Def_EepromMemory_Table_Entry_t::Filler16`

Definition at line 77 of file cs\_tbldefs.h.

#### 11.144.2.2 NumBytesToChecksum `uint32 CS_Def_EepromMemory_Table_Entry_t::NumBytesToChecksum`

Padding.

<

The number of Bytes to Checksum

Definition at line 78 of file cs\_tbldefs.h.

Referenced by CS\_InitializeDefaultTables(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ValidateEepromChecksumDefinitionTable(), and CS\_ValidateMemoryChecksumDefinitionTable().

#### 11.144.2.3 StartAddress `cpuaddr CS_Def_EepromMemory_Table_Entry_t::StartAddress`

The Start address to Checksum.

Definition at line 75 of file cs\_tbldefs.h.

Referenced by CS\_InitializeDefaultTables(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_RecomputeEepromMemoryChildTask(), CS\_ValidateEepromChecksumDefinitionTable(), and CS\_ValidateMemoryChecksumDefinitionTable().

**11.144.2.4 State** `uint16 CS_Def_EepromMemory_Table_Entry_t::State`

Uses the CS\_STATE\_... defines from above.

Definition at line 76 of file cs\_tbldefs.h.

Referenced by CS\_DisableEntryIDEEPROMCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_InitializeDefaultTables(), CS\_ProcessNewEEPROMMemoryDefinitionTable(), CS\_RecomputeEEPROMMemoryChildTask(), CS\_ValidateEEPROMChecksumDefinitionTable(), and CS\_ValidateMemoryChecksumDefinitionTable().

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_tbldefs.h`

**11.145 CS\_Def\_Tables\_Table\_Entry\_t Struct Reference**

Data structure for the Tables definition table.

```
#include <cs_tbldefs.h>
```

**Data Fields**

- `uint16 State`  
*Uses the CS\_STATE\_... defines from above.*
- `char Name [CFE_TBL_MAX_FULL_NAME_LEN]`  
*name of the table*

**11.145.1 Detailed Description**

Data structure for the Tables definition table.

Definition at line 95 of file cs\_tbldefs.h.

**11.145.2 Field Documentation****11.145.2.1 Name** `char CS_Def_Tables_Table_Entry_t::Name [CFE_TBL_MAX_FULL_NAME_LEN]`

name of the table

Definition at line 98 of file cs\_tbldefs.h.

Referenced by CS\_GetTableDefTblEntryByName(), CS\_InitializeDefaultTables(), CS\_ProcessNewTablesDefinitionTable(), CS\_RecomputeTablesChildTask(), and CS\_ValidateTablesChecksumDefinitionTable().

**11.145.2.2 State** `uint16 CS_Def_Tables_Table_Entry_t::State`

Uses the CS\_STATE\_... defines from above.

Definition at line 97 of file cs\_tbldefs.h.

Referenced by CS\_DisableNameTablesCmd(), CS\_EnableNameTablesCmd(), CS\_GetTableDefTblEntryByName(), CS\_InitializeDefaultTables(), CS\_ProcessNewTablesDefinitionTable(), CS\_RecomputeTablesChildTask(), and CS\_ValidateTablesChecksumDefinitionTable().

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_tbldefs.h`

**11.146 CS\_EntryCmd\_Payload\_t Struct Reference**

Payload for commands using Memory or EEPROM tables.

```
#include <cs_msg.h>
```

## Data Fields

- `uint32 EntryID`

*EntryID to perform a command on.*

### 11.146.1 Detailed Description

Payload for commands using Memory or EEPROM tables.

Definition at line 97 of file `cs_msg.h`.

### 11.146.2 Field Documentation

#### 11.146.2.1 `EntryID` `uint32 CS_EntryCmd_Payload_t::EntryID`

EntryID to perform a command on.

Definition at line 99 of file `cs_msg.h`.

Referenced by `CS_DisableEntryIDEEPROMCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_EnableEntryIDEEPROMCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_RecomputeBaselineEEPROMCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_ReportBaselineEntryIDEEPROMCmd()`, and `CS_ReportBaselineEntryIDMemoryCmd()`.

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_msg.h`

## 11.147 CS\_EntryCmd\_t Struct Reference

Command type for commands using Memory or EEPROM tables.

```
#include <cs_msg.h>
```

## Data Fields

- `CFE_MSG_CommandHeader_t CmdHeader`
- `CS_EntryCmd_Payload_t Payload`

### 11.147.1 Detailed Description

Command type for commands using Memory or EEPROM tables.

For command details see `CS_ENABLE_ENTRY_EEPROM_CC`, `CS_DISABLE_ENTRY_EEPROM_CC`, `CS_ENABLE_ENTRY_MEMORY_CC`, `CS_DISABLE_ENTRY_MEMORY_CC`

Definition at line 160 of file `cs_msg.h`.

### 11.147.2 Field Documentation

#### 11.147.2.1 `CmdHeader` `CFE_MSG_CommandHeader_t CS_EntryCmd_t::CmdHeader`

Definition at line 162 of file `cs_msg.h`.

#### 11.147.2.2 `Payload` `CS_EntryCmd_Payload_t CS_EntryCmd_t::Payload`

Definition at line 163 of file `cs_msg.h`.

Referenced by `CS_DisableEntryIDEEPROMCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_EnableEntryIDEEPROMCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_RecomputeBaselineEEPROMCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_ReportBaselineEntryIDEEPROMCmd()`, and `CS_ReportBaselineEntryIDMemoryCmd()`.

The documentation for this struct was generated from the following file:

- apps/cs/fsw/inc/cs\_msg.h

## 11.148 CS\_GetEntryIDCmd\_Payload\_t Struct Reference

Get entry ID command payload.

```
#include <cs_msg.h>
```

### Data Fields

- `cptuaddr` Address

*Address to get the ID for.*

#### 11.148.1 Detailed Description

Get entry ID command payload.

Definition at line 89 of file cs\_msg.h.

#### 11.148.2 Field Documentation

##### 11.148.2.1 Address `cptuaddr` CS\_GetEntryIDCmd\_Payload\_t::Address

Address to get the ID for.

Definition at line 91 of file cs\_msg.h.

Referenced by CS\_GetEntryIDEepromCmd(), and CS\_GetEntryIDMemoryCmd().

The documentation for this struct was generated from the following file:

- apps/cs/fsw/inc/cs\_msg.h

## 11.149 CS\_GetEntryIDCmd\_t Struct Reference

Get entry ID command.

```
#include <cs_msg.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t` CmdHeader
- `CS_GetEntryIDCmd_Payload_t` Payload

#### 11.149.1 Detailed Description

Get entry ID command.

For command details see CS\_GET\_ENTRY\_ID\_EEPROM\_CC, [CS\\_GET\\_ENTRY\\_ID\\_MEMORY\\_CC](#)

Definition at line 148 of file cs\_msg.h.

#### 11.149.2 Field Documentation

##### 11.149.2.1 CmdHeader `CFE_MSG_CommandHeader_t` CS\_GetEntryIDCmd\_t::CmdHeader

Definition at line 150 of file cs\_msg.h.

**11.149.2.2 Payload** `CS_GetEntryIDCmd_Payload_t` `CS_GetEntryIDCmd_t::Payload`  
Definition at line 151 of file `cs_msg.h`.

Referenced by `CS_GetEntryIDEepromCmd()`, and `CS_GetEntryIDMemoryCmd()`.  
The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_msg.h`

## 11.150 CS\_HkPacket\_Payload\_t Struct Reference

Housekeeping Payload Structure.

```
#include <cs_msg.h>
```

### Data Fields

- `uint8 CmdCounter`  
*CS Application Command Counter.*
- `uint8 CmdErrCounter`  
*CS Application Command Error Counter.*
- `uint8 ChecksumState`  
*CS Application global checksum state.*
- `uint8 EepromCSState`  
*CS EEPROM table checksum state.*
- `uint8 MemoryCSState`  
*CS Memory table checksum state.*
- `uint8 AppCSState`  
*CS App table checksum state.*
- `uint8 TablesCSState`  
*CS Tables table checksum state.*
- `uint8 OSCSState`  
*OS code segment checksum state.*
- `uint8 CfeCoreCSState`  
*cFE Core code segment checksum stat e*
- `uint8 RecomputeInProgress`  
*CS "Recompute In Progress" flag.*
- `uint8 OneShotInProgress`  
*CS "OneShot In Progress" flag.*
- `uint8 Filler8`  
*8 bit padding*
- `uint16 EepromCSErrCounter`  
*EEPROM miscompare counter.*
- `uint16 MemoryCSErrCounter`  
*Memory miscompare counter.*
- `uint16 AppCSErrCounter`  
*App miscompare counter.*
- `uint16 TablesCSErrCounter`  
*Tables miscompare counter.*
- `uint16 CfeCoreCSErrCounter`  
*cFE core miscompare counter*
- `uint16 OSCSErrCounter`

- **uint16 CurrentCSTable**  
*Current table being checksummed.*
- **uint16 CurrentEntryInTable**  
*Current entry ID in table being checksummed.*
- **uint32 EepromBaseline**  
*Baseline checksum for all of EEPROM.*
- **uint32 OSBaseline**  
*Baseline checksum for the OS code segment.*
- **uint32 CfeCoreBaseline**  
*Baseline checksum for the cFE core.*
- **cpuaddr LastOneShotAddress**  
*Address used in last one shot checksum command.*
- **uint32 LastOneShotSize**  
*Size used in the last one shot checksum command.*
- **uint32 LastOneShotMaxBytesPerCycle**  
*Max bytes per cycle for last one shot checksum command.*
- **uint32 LastOneShotChecksum**  
*Checksum of the last one shot checksum command.*
- **uint32 PassCounter**  
*Number of times CS has passed through all of its tables.*

### 11.150.1 Detailed Description

Housekeeping Payload Structure.

Definition at line 38 of file cs\_msg.h.

### 11.150.2 Field Documentation

**11.150.2.1 AppCSErrCounter** `uint16 CS_HkPacket_Payload_t::AppCSErrCounter`  
App miscompare counter.

Definition at line 54 of file cs\_msg.h.

Referenced by CS\_BackgroundApp(), and CS\_ResetCmd().

**11.150.2.2 AppCSState** `uint8 CS_HkPacket_Payload_t::AppCSState`  
CS App table checksum state.

Definition at line 45 of file cs\_msg.h.

Referenced by CS\_AppInit(), CS\_BackgroundApp(), CS\_CreateRestoreStatesFromCDS(), CS\_DisableAppCmd(), C↔S\_EnableAppCmd(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), CS\_ProcessNewAppDefinitionTable(), C↔S\_TableInit(), and CS\_UpdateCDS().

**11.150.2.3 CfeCoreBaseline** `uint32 CS_HkPacket_Payload_t::CfeCoreBaseline`  
Baseline checksum for the cFE core.

Definition at line 62 of file cs\_msg.h.

Referenced by CS\_BackgroundCfeCore(), and CS\_RecomputeEepromMemoryChildTask().

**11.150.2.4 CfeCoreCSErrCounter** `uint16 CS_HkPacket_Payload_t::CfeCoreCSErrCounter`  
cFE core miscompare counter  
Definition at line 56 of file `cs_msg.h`.  
Referenced by `CS_BackgroundCfeCore()`, and `CS_ResetCmd()`.

**11.150.2.5 CfeCoreCSState** `uint8 CS_HkPacket_Payload_t::CfeCoreCSState`  
cFE Core code segment checksum stat e  
Definition at line 48 of file `cs_msg.h`.  
Referenced by `CS_AppInit()`, `CS_BackgroundCfeCore()`, `CS_CreateRestoreStatesFromCDS()`, `CS_DisableCfeCoreCmd()`, `CS_EnableCfeCoreCmd()`, and `CS_UpdateCDS()`.

**11.150.2.6 ChecksumState** `uint8 CS_HkPacket_Payload_t::ChecksumState`  
CS Application global checksum state.  
Definition at line 42 of file `cs_msg.h`.  
Referenced by `CS_AppInit()`, `CS_BackgroundCheckCycle()`, `CS_DisableAllCSCmd()`, and `CS_EnableAllCSCmd()`.

**11.150.2.7 CmdCounter** `uint8 CS_HkPacket_Payload_t::CmdCounter`  
CS Application Command Counter.  
Definition at line 40 of file `cs_msg.h`.  
Referenced by `CS_CancelOneShotCmd()`, `CS_DisableAllCSCmd()`, `CS_DisableAppCmd()`, `CS_DisableCfeCoreCmd()`, `CS_DisableEepromCmd()`, `CS_DisableEntryIDEepromCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_DisableMemoryCmd()`, `CS_DisableNameAppCmd()`, `CS_DisableNameTablesCmd()`, `CS_DisableOSCmd()`, `CS_DisableTablesCmd()`, `CS_EnableAllCSCmd()`, `CS_EnableAppCmd()`, `CS_EnableCfeCoreCmd()`, `CS_EnableEepromCmd()`, `CS_EnableEntryIDEepromCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_EnableMemoryCmd()`, `CS_EnableNameAppCmd()`, `CS_EnableNameTablesCmd()`, `CS_EnableOSCmd()`, `CS_EnableTablesCmd()`, `CS_GetEntryIDEepromCmd()`, `CS_GetEntryIDMemoryCmd()`, `CS_NoopCmd()`, `CS_OneShotCmd()`, `CS_RecomputeBaselineAppCmd()`, `CS_RecomputeBaselineCfeCoreCmd()`, `CS_RecomputeBaselineEepromCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeBaselineOSCmd()`, `CS_RecomputeBaselineTablesCmd()`, `CS_ReportBaselineAppCmd()`, `CS_ReportBaselineCfeCoreCmd()`, `CS_ReportBaselineEntryIDEepromCmd()`, `CS_ReportBaselineEntryIDMemoryCmd()`, `CS_ReportBaselineOSCmd()`, `CS_ReportBaselineTablesCmd()`, and `CS_ResetCmd()`.

**11.150.2.8 CmdErrCounter** `uint8 CS_HkPacket_Payload_t::CmdErrCounter`  
CS Application Command Error Counter.  
Definition at line 41 of file `cs_msg.h`.  
Referenced by `CS_AppPipe()`, `CS_CancelOneShotCmd()`, `CS_CheckRecomputeOneshot()`, `CS_DisableEntryIDEepromCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_DisableNameAppCmd()`, `CS_DisableNameTablesCmd()`, `CS_EnableEntryIDEepromCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_EnableNameAppCmd()`, `CS_EnableNameTablesCmd()`, `CS_OneShotCmd()`, `CS_ProcessCmd()`, `CS_RecomputeBaselineAppCmd()`, `CS_RecomputeBaselineCfeCoreCmd()`, `CS_RecomputeBaselineEepromCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeBaselineOSCmd()`, `CS_RecomputeBaselineTablesCmd()`, `CS_ReportBaselineAppCmd()`, `CS_ReportBaselineEntryIDEepromCmd()`, `CS_ReportBaselineEntryIDMemoryCmd()`, `CS_ReportBaselineOSCmd()`, `CS_ReportBaselineTablesCmd()`, `CS_ResetCmd()`, and `CS_VerifyCmdLength()`.

**11.150.2.9 CurrentCSTable** `uint16 CS_HkPacket_Payload_t::CurrentCSTable`  
Current table being checksummed.  
Definition at line 58 of file `cs_msg.h`.  
Referenced by `CS_AppInit()`, `CS_BackgroundCheckCycle()`, and `CS_GoToNextTable()`.

**11.150.2.10 CurrentEntryInTable** `uint16` CS\_HkPacket\_Payload\_t::CurrentEntryInTable

Current entry ID in table being checksummed.

Definition at line 59 of file cs\_msg.h.

Referenced by CS\_AppInit(), CS\_BackgroundApp(), CS\_BackgroundCfeCore(), CS\_BackgroundCheckCycle(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), CS\_BackgroundTables(), CS\_FindEnabledAppEntry(), CS\_FindEnabledEepromEntry(), CS\_FindEnabledMemoryEntry(), CS\_FindEnabledTablesEntry(), and CS\_GoToNextTable().

**11.150.2.11 EepromBaseline** `uint32` CS\_HkPacket\_Payload\_t::EepromBaseline

Baseline checksum for all of EEPROM.

Definition at line 60 of file cs\_msg.h.

Referenced by CS\_BackgroundEeprom().

**11.150.2.12 EepromCSErrCounter** `uint16` CS\_HkPacket\_Payload\_t::EepromCSErrCounter

EEPROM miscompare counter.

Definition at line 52 of file cs\_msg.h.

Referenced by CS\_BackgroundEeprom(), and CS\_ResetCmd().

**11.150.2.13 EepromCSState** `uint8` CS\_HkPacket\_Payload\_t::EepromCSState

CS EEPROM table checksum state.

Definition at line 43 of file cs\_msg.h.

Referenced by CS\_AppInit(), CS\_BackgroundEeprom(), CS\_CreateRestoreStatesFromCDS(), CS\_DisableEepromCmd(), CS\_EnableEepromCmd(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_TableInit(), and CS\_UpdateCDS().

**11.150.2.14 Filler8** `uint8` CS\_HkPacket\_Payload\_t::Filler8

8 bit padding

Definition at line 51 of file cs\_msg.h.

**11.150.2.15 LastOneShotAddress** `cpuaddr` CS\_HkPacket\_Payload\_t::LastOneShotAddress

Address used in last one shot checksum command.

Definition at line 63 of file cs\_msg.h.

Referenced by CS\_OneShotChildTask(), and CS\_OneShotCmd().

**11.150.2.16 LastOneShotChecksum** `uint32` CS\_HkPacket\_Payload\_t::LastOneShotChecksum

Checksum of the last one shot checksum command.

Definition at line 66 of file cs\_msg.h.

Referenced by CS\_OneShotChildTask(), and CS\_OneShotCmd().

**11.150.2.17 LastOneShotMaxBytesPerCycle** `uint32` CS\_HkPacket\_Payload\_t::LastOneShotMaxBytesPerCycle

Max bytes per cycle for last one shot checksum command.

Definition at line 65 of file cs\_msg.h.

Referenced by CS\_OneShotChildTask(), and CS\_OneShotCmd().

**11.150.2.18 LastOneShotSize** `uint32 CS_HkPacket_Payload_t::LastOneShotSize`  
Size used in the last one shot checksum command.

Definition at line 64 of file `cs_msg.h`.

Referenced by `CS_OneShotChildTask()`, and `CS_OneShotCmd()`.

**11.150.2.19 MemoryCSErrCounter** `uint16 CS_HkPacket_Payload_t::MemoryCSErrCounter`

Memory miscompare counter.

Definition at line 53 of file `cs_msg.h`.

Referenced by `CS_BackgroundMemory()`, and `CS_ResetCmd()`.

**11.150.2.20 MemoryCSState** `uint8 CS_HkPacket_Payload_t::MemoryCSState`

CS Memory table checksum state.

Definition at line 44 of file `cs_msg.h`.

Referenced by `CS_AppInit()`, `CS_BackgroundMemory()`, `CS_CreateRestoreStatesFromCDS()`, `CS_DisableMemoryCmd()`, `CS_EnableMemoryCmd()`, `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, `CS_TableInit()`, and `CS_UpdateCDS()`.

**11.150.2.21 OneShotInProgress** `uint8 CS_HkPacket_Payload_t::OneShotInProgress`

CS "OneShot In Progress" flag.

Definition at line 50 of file `cs_msg.h`.

Referenced by `CS_AppInit()`, `CS_BackgroundCheckCycle()`, `CS_CancelOneShotCmd()`, `CS_CheckRecomputeOneshot()`, `CS_HandleRoutineTableUpdates()`, `CS_OneShotChildTask()`, `CS_OneShotCmd()`, `CS_RecomputeBaselineAppCmd()`, `CS_RecomputeBaselineCfeCoreCmd()`, `CS_RecomputeBaselineEepromCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeBaselineOSCmd()`, and `CS_RecomputeBaselineTablesCmd()`.

**11.150.2.22 OSBaseline** `uint32 CS_HkPacket_Payload_t::OSBaseline`

Baseline checksum for the OS code segment.

Definition at line 61 of file `cs_msg.h`.

Referenced by `CS_BackgroundOS()`, and `CS_RecomputeEepromMemoryChildTask()`.

**11.150.2.23 OSCSErrCounter** `uint16 CS_HkPacket_Payload_t::OSCSErrCounter`

OS code segment miscopmare counter.

Definition at line 57 of file `cs_msg.h`.

Referenced by `CS_BackgroundOS()`, and `CS_ResetCmd()`.

**11.150.2.24 OSCSState** `uint8 CS_HkPacket_Payload_t::OSCSState`

OS code segment checksum state.

Definition at line 47 of file `cs_msg.h`.

Referenced by `CS_AppInit()`, `CS_BackgroundOS()`, `CS_CreateRestoreStatesFromCDS()`, `CS_DisableOSCmd()`, `CS_EnableOSCmd()`, and `CS_UpdateCDS()`.

**11.150.2.25 PassCounter** `uint32 CS_HkPacket_Payload_t::PassCounter`

Number of times CS has passed through all of its tables.

Definition at line 67 of file `cs_msg.h`.

Referenced by `CS_BackgroundCheckCycle()`, `CS_GoToNextTable()`, and `CS_ResetCmd()`.

**11.150.2.26 RecomputeInProgress** `uint8 CS_HkPacket_Payload_t::RecomputeInProgress`

CS "Recompute In Progress" flag.

Definition at line 49 of file `cs_msg.h`.

Referenced by `CS_AppInit()`, `CS_BackgroundCheckCycle()`, `CS_CancelOneShotCmd()`, `CS_CheckRecomputeOneshot()`, `CS_HandleRoutineTableUpdates()`, `CS_OneShotCmd()`, `CS_RecomputeAppChildTask()`, `CS_RecomputeBaselineAppCmd()`, `CS_RecomputeBaselineCfeCoreCmd()`, `CS_RecomputeBaselineEepromCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeBaselineOSCmd()`, `CS_RecomputeBaselineTablesCmd()`, `CS_RecomputeEepromMemoryChildTask()`, and `CS_RecomputeTablesChildTask()`.

**11.150.2.27 TablesCSErrCounter** `uint16 CS_HkPacket_Payload_t::TablesCSErrCounter`

Tables miscompare counter.

Definition at line 55 of file `cs_msg.h`.

Referenced by `CS_BackgroundTables()`, and `CS_ResetCmd()`.

**11.150.2.28 TablesCSState** `uint8 CS_HkPacket_Payload_t::TablesCSState`

CS Tables table checksum state.

Definition at line 46 of file `cs_msg.h`.

Referenced by `CS_AppInit()`, `CS_BackgroundTables()`, `CS_CreateRestoreStatesFromCDS()`, `CS_DisableTablesCmd()`, `CS_EnableTablesCmd()`, `CS_HandleRoutineTableUpdates()`, `CS_InitAllTables()`, `CS_ProcessNewTablesDefinitionTable()`, `CS_TableInit()`, and `CS_UpdateCDS()`.

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_msg.h`

## 11.151 CS\_HkPacket\_t Struct Reference

Housekeeping Packet Structure.

```
#include <cs_msg.h>
```

### Data Fields

- [CFE\\_MSG\\_TelemetryHeader\\_t TlmHeader](#)  
*cFE SB Tlm Msg Hdr*
- [CS\\_HkPacket\\_Payload\\_t Payload](#)  
*CS HK Payload.*

### 11.151.1 Detailed Description

Housekeeping Packet Structure.

Definition at line 73 of file `cs_msg.h`.

### 11.151.2 Field Documentation

**11.151.2.1 Payload** `CS_HkPacket_Payload_t CS_HkPacket_t::Payload`

CS HK Payload.

Definition at line 76 of file `cs_msg.h`.

Referenced by `CS_AppInit()`, `CS_AppPipe()`, `CS_BackgroundApp()`, `CS_BackgroundCfeCore()`, `CS_BackgroundCheckCycle()`, `CS_BackgroundEeprom()`, `CS_BackgroundMemory()`, `CS_BackgroundOS()`, `CS_BackgroundTables()`,

CS\_CancelOneShotCmd(), CS\_CheckRecomputeOneshot(), CS\_CreateRestoreStatesFromCDS(), CS\_DisableAllSCmd(), CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEepromCmd(), CS\_DisableEntryIDEEPROMCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_EnableAllCSCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEepromCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_EnableOSCmd(), CS\_EnableTablesCmd(), CS\_FindEnabledAppEntry(), CS\_FindEnabledEepromEntry(), CS\_FindEnabledMemoryEntry(), CS\_FindEnabledTablesEntry(), CS\_GetEntryIDEEPROMCmd(), CS\_GetEntryIDMemoryCmd(), CS\_GoToNextTable(), CS\_HandleRoutineTableUpdates(), CS\_InitAllTables(), CS\_NoopCmd(), CS\_OneShotChildTask(), CS\_OneShotCmd(), CS\_ProcessCmd(), CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), CS\_RecomputeAppChildTask(), CS\_RecomputeBaselineAppCmd(), CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), CS\_RecomputeBaselineTablesCmd(), CS\_RecomputeEepromMemoryChildTask(), CS\_RecomputeTablesChildTask(), CS\_ReportBaselineAppCmd(), CS\_ReportBaselineCfeCoreCmd(), CS\_ReportBaselineEntryIDEEPROMCmd(), CS\_ReportBaselineEntryIDMemoryCmd(), CS\_ReportBaselineOSCmd(), CS\_ReportBaselineTablesCmd(), CS\_ResetCmd(), CS\_TableInit(), CS\_UpdateCDS(), and CS\_VerifyCmdLength().

### 11.151.2.2 TlmHeader [CFE\\_MSG\\_TelemetryHeader\\_t](#) CS\_HkPacket\_t::TlmHeader

cFE SB Tlm Msg Hdr

Definition at line 75 of file cs\_msg.h.

Referenced by CS\_HousekeepingCmd(), and CS\_SblInit().

The documentation for this struct was generated from the following file:

- [apps/cs/fsw/inc/cs\\_msg.h](#)

## 11.152 CS\_NoArgsCmd\_t Struct Reference

No arguments command data type.

```
#include <cs_msg.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CmdHeader

### 11.152.1 Detailed Description

No arguments command data type.

For command details see [CS\\_NOOP\\_CC](#) [CS\\_RESET\\_CC](#), [CS\\_ENABLE\\_ALL\\_CS\\_CC](#), [CS\\_DISABLE\\_ALL\\_CS\\_CC](#), [CS\\_ENABLE\\_CFE\\_CORE\\_CC](#), [CS\\_DISABLE\\_CFE\\_CORE\\_CC](#), [CS\\_ENABLE\\_OS\\_CC](#), [CS\\_DISABLE\\_OS\\_CC](#), [CS\\_ENABLE\\_EEPROM\\_CC](#), [CS\\_DISABLE\\_EEPROM\\_CC](#), [CS\\_ENABLE\\_MEMORY\\_CC](#), [CS\\_DISABLE\\_MEMORY\\_CC](#), [CS\\_ENABLE\\_TABLES\\_CC](#), [CS\\_DISABLE\\_TABLES\\_CC](#), [CS\\_ENABLE\\_APPS\\_CC](#), [CS\\_DISABLE\\_APPS\\_CC](#), [CS\\_CANCEL\\_ONE\\_SHOT\\_CC](#)

Definition at line 138 of file cs\_msg.h.

### 11.152.2 Field Documentation

#### 11.152.2.1 CmdHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CS\_NoArgsCmd\_t::CmdHeader

Definition at line 140 of file cs\_msg.h.

Referenced by CS\_BackgroundCheckCycle(), and CS\_HousekeepingCmd().

The documentation for this struct was generated from the following file:

- [apps/cs/fsw/inc/cs\\_msg.h](#)

## 11.153 CS\_OneShotCmd\_Payload\_t Struct Reference

Payload for sending one shot calculation.

```
#include <cs_msg.h>
```

### Data Fields

- [cpuaddr Address](#)  
*Address to start checksum.*
- [uint32 Size](#)  
*Number of bytes to checksum.*
- [uint32 MaxBytesPerCycle](#)  
*Max Number of bytes to compute per cycle. Value of Zero to use platform config value.*

### 11.153.1 Detailed Description

Payload for sending one shot calculation.

Definition at line 121 of file cs\_msg.h.

### 11.153.2 Field Documentation

#### 11.153.2.1 Address [cpuaddr CS\\_OneShotCmd\\_Payload\\_t::Address](#)

Address to start checksum.

Definition at line 123 of file cs\_msg.h.

Referenced by CS\_OneShotCmd().

#### 11.153.2.2 MaxBytesPerCycle [uint32 CS\\_OneShotCmd\\_Payload\\_t::MaxBytesPerCycle](#)

Max Number of bytes to compute per cycle. Value of Zero to use platform config value.

Definition at line 125 of file cs\_msg.h.

Referenced by CS\_OneShotCmd().

#### 11.153.2.3 Size [uint32 CS\\_OneShotCmd\\_Payload\\_t::Size](#)

Number of bytes to checksum.

Definition at line 124 of file cs\_msg.h.

Referenced by CS\_OneShotCmd().

The documentation for this struct was generated from the following file:

- [apps/cs/fsw/inc/cs\\_msg.h](#)

## 11.154 CS\_OneShotCmd\_t Struct Reference

Command type for sending one shot calculation.

```
#include <cs_msg.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CmdHeader](#)
- [CS\\_OneShotCmd\\_Payload\\_t Payload](#)

### 11.154.1 Detailed Description

Command type for sending one shot calculation.  
For command details see [CS\\_ONE\\_SHOT\\_CC](#)  
Definition at line 195 of file cs\_msg.h.

### 11.154.2 Field Documentation

#### 11.154.2.1 CmdHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CS\_OneShotCmd\_t::CmdHeader

Definition at line 197 of file cs\_msg.h.

#### 11.154.2.2 Payload [CS\\_OneShotCmd\\_Payload\\_t](#) CS\_OneShotCmd\_t::Payload

Definition at line 198 of file cs\_msg.h.

Referenced by CS\_OneShotCmd().

The documentation for this struct was generated from the following file:

- [apps/cs/fsw/inc/cs\\_msg.h](#)

## 11.155 CS\_Res\_App\_Table\_Entry\_t Struct Reference

Data structure for the app result table.

```
#include <cs_tbldefs.h>
```

### Data Fields

- [cpuaddr StartAddress](#)  
*The Start address to Checksum.*
- [uint16 State](#)  
*Uses the CS\_STATE\_... defines from above.*
- [uint16 ComputedYet](#)  
*Have we computed an Integrity value yet.*
- [uint32 NumBytesToChecksum](#)  
*The number of Bytes to Checksum.*
- [uint32 ComparisonValue](#)  
*The Memory Integrity Value.*
- [uint32 ByteOffset](#)  
*Where a previous unfinished calc left off.*
- [uint32 TempChecksumValue](#)  
*The unfinished calculation.*
- [char Name \[OS\\_MAX\\_API\\_NAME\]](#)  
*name of the app*

### 11.155.1 Detailed Description

Data structure for the app result table.  
Definition at line 131 of file cs\_tbldefs.h.

### 11.155.2 Field Documentation

**11.155.2.1 ByteOffset** `uint32` `CS_Res_App_Table_Entry_t::ByteOffset`

Where a previous unfinished calc left off.

Definition at line 138 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeApp()`, `CS_DisableNameAppCmd()`, `CS_ProcessNewAppDefinitionTable()`, `CS_RecomputeAppChildTask()`, and `CS_ZeroAppTempValues()`.

**11.155.2.2 ComparisonValue** `uint32` `CS_Res_App_Table_Entry_t::ComparisonValue`

The Memory Integrity Value.

Definition at line 137 of file `cs_tbldefs.h`.

Referenced by `CS_BackgroundApp()`, `CS_ComputeApp()`, `CS_ProcessNewAppDefinitionTable()`, and `CS_ReportBaselineAppCmd()`.

**11.155.2.3 ComputedYet** `uint16` `CS_Res_App_Table_Entry_t::ComputedYet`

Have we computed an Integrity value yet.

Definition at line 135 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeApp()`, `CS_ProcessNewAppDefinitionTable()`, `CS_RecomputeAppChildTask()`, and `CS_ReportBaselineAppCmd()`.

**11.155.2.4 Name** `char` `CS_Res_App_Table_Entry_t::Name[OS_MAX_API_NAME]`

name of the app

Definition at line 140 of file `cs_tbldefs.h`.

Referenced by `CS_BackgroundApp()`, `CS_ComputeApp()`, `CS_GetAppResTblEntryByName()`, `CS_ProcessNewAppDefinitionTable()`, and `CS_RecomputeAppChildTask()`.

**11.155.2.5 NumBytesToChecksum** `uint32` `CS_Res_App_Table_Entry_t::NumBytesToChecksum`

The number of Bytes to Checksum.

Definition at line 136 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeApp()`, and `CS_ProcessNewAppDefinitionTable()`.

**11.155.2.6 StartAddress** `cpuaddr` `CS_Res_App_Table_Entry_t::StartAddress`

The Start address to Checksum.

Definition at line 133 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeApp()`, and `CS_ProcessNewAppDefinitionTable()`.

**11.155.2.7 State** `uint16` `CS_Res_App_Table_Entry_t::State`

Uses the `CS_STATE_...` defines from above.

Definition at line 134 of file `cs_tbldefs.h`.

Referenced by `CS_DisableNameAppCmd()`, `CS_EnableNameAppCmd()`, `CS_FindEnabledAppEntry()`, `CS_GetAppResTblEntryByName()`, `CS_ProcessNewAppDefinitionTable()`, and `CS_RecomputeAppChildTask()`.

**11.155.2.8 TempChecksumValue** `uint32` `CS_Res_App_Table_Entry_t::TempChecksumValue`

The unfinished calculation.

Definition at line 139 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeApp()`, `CS_DisableNameAppCmd()`, `CS_ProcessNewAppDefinitionTable()`, `CS_RecomputeAppChildTask()`, and `CS_ZeroAppTempValues()`.

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_tbldefs.h`

## 11.156 CS\_Res\_EepromMemory\_Table\_Entry\_t Struct Reference

Data structure for the Eeporom or Memory results table.

```
#include <cs_tbldefs.h>
```

### Data Fields

- `cpuaddr StartAddress`  
*The Start address to Checksum.*
- `uint16 State`  
*Uses the CS\_STATE\_... defines from above.*
- `uint16 ComputedYet`  
*Have we computed an Integrity value yet.*
- `uint32 NumBytesToChecksum`  
*The number of Bytes to Checksum.*
- `uint32 ComparisonValue`  
*The Memory Integrity Value.*
- `uint32 ByteOffset`  
*Where a previous unfinished calc left off.*
- `uint32 TempChecksumValue`  
*The unfinished caluculation.*
- `uint32 Filler32`  
*Padding.*

### 11.156.1 Detailed Description

Data structure for the Eeporom or Memory results table.

Definition at line 80 of file `cs_tbldefs.h`.

### 11.156.2 Field Documentation

#### 11.156.2.1 ByteOffset `uint32 CS_Res_EepromMemory_Table_Entry_t::ByteOffset`

Where a previous unfinished calc left off.

Definition at line 87 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeEepromMemory()`, `CS_DisableEntryIDEepromCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_InitSegments()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, `CS_RecomputeEepromMemoryChildTask()`, `CS_ZeroCfeCoreTempValues()`, `CS_ZeroEepromTempValues()`, `CS_ZeroMemoryTempValues()`, and `CS_ZeroOS←TempValues()`.

#### 11.156.2.2 ComparisonValue `uint32 CS_Res_EepromMemory_Table_Entry_t::ComparisonValue`

The Memory Integrity Value.

Definition at line 86 of file `cs_tbldefs.h`.

Referenced by `CS_BackgroundCfeCore()`, `CS_BackgroundEeprom()`, `CS_BackgroundMemory()`, `CS_Background←OS()`, `CS_ComputeEepromMemory()`, `CS_InitSegments()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, `CS←ReportBaselineCfeCoreCmd()`, `CS_ReportBaselineEntryIDEepromCmd()`, `CS_ReportBaselineEntryIDMemoryCmd()`, and `CS_ReportBaselineOSCmd()`.

**11.156.2.3 ComputedYet** `uint16` `CS_Res_EepromMemory_Table_Entry_t::ComputedYet`

Have we computed an Integrity value yet.

Definition at line 84 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeEepromMemory()`, `CS_InitSegments()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, `CS_RecomputeEepromMemoryChildTask()`, `CS_ReportBaselineCfeCoreCmd()`, `CS_ReportBaselineEntryIDEEPROMCmd()`, `CS_ReportBaselineEntryIDMemoryCmd()`, and `CS_ReportBaselineOSCmd()`.

**11.156.2.4 Filler32** `uint32` `CS_Res_EepromMemory_Table_Entry_t::Filler32`

Padding.

Definition at line 89 of file `cs_tbldefs.h`.

**11.156.2.5 NumBytesToChecksum** `uint32` `CS_Res_EepromMemory_Table_Entry_t::NumBytesToChecksum`

The number of Bytes to Checksum.

Definition at line 85 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeEepromMemory()`, `CS_GetEntryIDEEPROMCmd()`, `CS_GetEntryIDMemoryCmd()`, `CS_InitSegments()`, and `CS_ProcessNewEepromMemoryDefinitionTable()`.

**11.156.2.6 StartAddress** `cpuaddr` `CS_Res_EepromMemory_Table_Entry_t::StartAddress`

The Start address to Checksum.

Definition at line 82 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeEepromMemory()`, `CS_GetEntryIDEEPROMCmd()`, `CS_GetEntryIDMemoryCmd()`, `CS_InitSegments()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, and `CS_RecomputeEepromMemoryChildTask()`.

**11.156.2.7 State** `uint16` `CS_Res_EepromMemory_Table_Entry_t::State`

Uses the `CS_STATE_...` defines from above.

Definition at line 83 of file `cs_tbldefs.h`.

Referenced by `CS_BackgroundCfeCore()`, `CS_BackgroundOS()`, `CS_DisableEntryIDEEPROMCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_EnableEntryIDEEPROMCmd()`, `CS_EnableEntryIDMemoryCmd()`, `CS_FindEnabledEEPROMEntry()`, `CS_FindEnabledMemoryEntry()`, `CS_GetEntryIDEEPROMCmd()`, `CS_GetEntryIDMemoryCmd()`, `CS_InitSegments()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, `CS_RecomputeBaselineEEPROMCmd()`, `CS_RecomputeBaselineMemoryCmd()`, `CS_RecomputeEepromMemoryChildTask()`, `CS_ReportBaselineEntryIDEEPROMCmd()`, and `CS_ReportBaselineEntryIDMemoryCmd()`.

**11.156.2.8 TempChecksumValue** `uint32` `CS_Res_EepromMemory_Table_Entry_t::TempChecksumValue`

The unfinished calculation.

Definition at line 88 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeEepromMemory()`, `CS_DisableEntryIDEEPROMCmd()`, `CS_DisableEntryIDMemoryCmd()`, `CS_InitSegments()`, `CS_ProcessNewEepromMemoryDefinitionTable()`, `CS_RecomputeEepromMemoryChildTask()`, `CS_ZeroCfeCoreTempValues()`, `CS_ZeroEEPROMTempValues()`, `CS_ZeroMemoryTempValues()`, and `CS_ZeroOSTempValues()`.

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_tbldefs.h`

**11.157 CS\_Res\_Tables\_Table\_Entry\_t Struct Reference**

Data structure for the Tables result table.

```
#include <cs_tbldefs.h>
```

## Data Fields

- `cpuaddr StartAddress`

*The Start address to Checksum.*
- `uint16 State`

*Uses the CS\_STATE\_... defines from above.*
- `uint16 ComputedYet`

*Have we computed an Integrity value yet.*
- `uint32 NumBytesToChecksum`

*The number of Bytes to Checksum.*
- `uint32 ComparisonValue`

*The Memory Integrity Value.*
- `uint32 ByteOffset`

*Where a previous unfinished calc left off.*
- `uint32 TempChecksumValue`

*The unfinished calculation.*
- `CFE_TBL_Handle_t TblHandle`

*handle received from CFE\_TBL*
- `bool IsCSOwner`

*Is CS the original owner of this table.*
- `bool Filler8`

*Padding.*
- `char Name [CFE_TBL_MAX_FULL_NAME_LEN]`

*name of the table*

### 11.157.1 Detailed Description

Data structure for the Tables result table.

Definition at line 113 of file cs\_tbldefs.h.

### 11.157.2 Field Documentation

#### 11.157.2.1 ByteOffset `uint32 CS_Res_Tables_Table_Entry_t::ByteOffset`

Where a previous unfinished calc left off.

Definition at line 120 of file cs\_tbldefs.h.

Referenced by `CS_ComputeTables()`, `CS_DisableNameTablesCmd()`, `CS_ProcessNewTablesDefinitionTable()`, `CS_RecomputeTablesChildTask()`, `CS_ResetTablesTblResultEntry()`, and `CS_ZeroTablesTempValues()`.

#### 11.157.2.2 ComparisonValue `uint32 CS_Res_Tables_Table_Entry_t::ComparisonValue`

The Memory Integrity Value.

Definition at line 119 of file cs\_tbldefs.h.

Referenced by `CS_BackgroundTables()`, `CS_ComputeTables()`, `CS_ProcessNewTablesDefinitionTable()`, and `CS_ReportBaselineTablesCmd()`.

**11.157.2.3 ComputedYet** `uint16` `CS_Res_Tables_Table_Entry_t::ComputedYet`

Have we computed an Integrity value yet.

Definition at line 117 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeTables()`, `CS_ProcessNewTablesDefinitionTable()`, `CS_RecomputeTablesChildTask()`, `C←S_ReportBaselineTablesCmd()`, and `CS_ResetTablesTblResultEntry()`.

**11.157.2.4 Filler8** `bool` `CS_Res_Tables_Table_Entry_t::Filler8`

Padding.

Definition at line 124 of file `cs_tbldefs.h`.

**11.157.2.5 IsCSOwner** `bool` `CS_Res_Tables_Table_Entry_t::IsCSOwner`

Is CS the original owner of this table.

Definition at line 123 of file `cs_tbldefs.h`.

Referenced by `CS_HandleTableUpdate()`, and `CS_ProcessNewTablesDefinitionTable()`.

**11.157.2.6 Name** `char` `CS_Res_Tables_Table_Entry_t::Name[CFE_TBL_MAX_FULL_NAME_LEN]`

name of the table

Definition at line 125 of file `cs_tbldefs.h`.

Referenced by `CS_AttemptTableReshare()`, `CS_BackgroundTables()`, `CS_ComputeTables()`, `CS_GetTableResTbl←EntryByName()`, `CS_ProcessNewTablesDefinitionTable()`, and `CS_RecomputeTablesChildTask()`.

**11.157.2.7 NumBytesToChecksum** `uint32` `CS_Res_Tables_Table_Entry_t::NumBytesToChecksum`

The number of Bytes to Checksum.

Definition at line 118 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeTables()`, and `CS_ProcessNewTablesDefinitionTable()`.

**11.157.2.8 StartAddress** `cpuaddr` `CS_Res_Tables_Table_Entry_t::StartAddress`

The Start address to Checksum.

Definition at line 115 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeTables()`, and `CS_ProcessNewTablesDefinitionTable()`.

**11.157.2.9 State** `uint16` `CS_Res_Tables_Table_Entry_t::State`

Uses the `CS_STATE_...` defines from above.

Definition at line 116 of file `cs_tbldefs.h`.

Referenced by `CS_DisableNameTablesCmd()`, `CS_EnableNameTablesCmd()`, `CS_FindEnabledTablesEntry()`, `CS←GetTableResTblEntryByName()`, `CS_ProcessNewTablesDefinitionTable()`, and `CS_RecomputeTablesChildTask()`.

**11.157.2.10 TblHandle** `CFE_TBL_Handle_t` `CS_Res_Tables_Table_Entry_t::TblHandle`

handle received from `CFE_TBL`

Definition at line 122 of file `cs_tbldefs.h`.

Referenced by `CS_AttemptTableReshare()`, `CS_ComputeTables()`, `CS_HandleTableUpdate()`, and `CS_ProcessNew←TablesDefinitionTable()`.

**11.157.2.11 TempChecksumValue** `uint32 CS_Res_Tables_Table_Entry_t::TempChecksumValue`

The unfinished calculation.

Definition at line 121 of file `cs_tbldefs.h`.

Referenced by `CS_ComputeTables()`, `CS_DisableNameTablesCmd()`, `CS_ProcessNewTablesDefinitionTable()`, `CS_RecomputeTablesChildTask()`, `CS_ResetTablesTblResultEntry()`, and `CS_ZeroTablesTempValues()`.

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_tbldefs.h`

**11.158 CS\_TableNameCmd\_Payload\_t Struct Reference**

Payload for commanding by table name.

```
#include <cs_msg.h>
```

**Data Fields**

- `char Name [CFE_TBL_MAX_FULL_NAME_LEN]`

*Table name to perform a command on.*

**11.158.1 Detailed Description**

Payload for commanding by table name.

Definition at line 105 of file `cs_msg.h`.

**11.158.2 Field Documentation****11.158.2.1 Name** `char CS_TableNameCmd_Payload_t::Name [CFE_TBL_MAX_FULL_NAME_LEN]`

Table name to perform a command on.

Definition at line 107 of file `cs_msg.h`.

Referenced by `CS_DisableNameTablesCmd()`, `CS_EnableNameTablesCmd()`, `CS_RecomputeBaselineTablesCmd()`, and `CS_ReportBaselineTablesCmd()`.

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_msg.h`

**11.159 CS\_TableNameCmd\_t Struct Reference**

Command type for commanding by table name.

```
#include <cs_msg.h>
```

**Data Fields**

- `CFE_MSG_CommandHeader_t CmdHeader`
- `CS_TableNameCmd_Payload_t Payload`

**11.159.1 Detailed Description**

Command type for commanding by table name.

For command details see `CS_ENABLE_NAME_TABLE_CC`, `CS_DISABLE_NAME_TABLE_CC`, `CS_RECOMPUTE_BASELINE_TABLE_CC`, `CS_REPORT_BASELINE_TABLE_CC`

Definition at line 172 of file `cs_msg.h`.

## 11.159.2 Field Documentation

**11.159.2.1 CmdHeader** `CFE_MSG_CommandHeader_t CS_TableNameCmd_t::CmdHeader`  
Definition at line 174 of file `cs_msg.h`.

**11.159.2.2 Payload** `CS_TableNameCmd_Payload_t CS_TableNameCmd_t::Payload`  
Definition at line 175 of file `cs_msg.h`.

Referenced by `CS_DisableNameTablesCmd()`, `CS_EnableNameTablesCmd()`, `CS_RecomputeBaselineTablesCmd()`, and `CS_ReportBaselineTablesCmd()`.

The documentation for this struct was generated from the following file:

- `apps/cs/fsw/inc/cs_msg.h`

## 11.160 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`

## 11.160.1 Detailed Description

OSAL binary semaphore properties.

Definition at line 39 of file `osapi-binsem.h`.

## 11.160.2 Field Documentation

**11.160.2.1 creator** `osal_id_t OS_bin_sem_prop_t::creator`  
Definition at line 42 of file `osapi-binsem.h`.

**11.160.2.2 name** `char OS_bin_sem_prop_t::name[OS_MAX_API_NAME]`  
Definition at line 41 of file `osapi-binsem.h`.

**11.160.2.3 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`

## 11.161 OS\_condvar\_prop\_t Struct Reference

OSAL condition variable properties.

```
#include <osapi-condvar.h>
```

**Data Fields**

- char `name` [`OS_MAX_API_NAME`]
- `osal_id_t` `creator`

**11.161.1 Detailed Description**

OSAL condition variable properties.

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

**11.161.2 Field Documentation****11.161.2.1 `creator` `osal_id_t` `OS_condvar_prop_t::creator`**

Definition at line 37 of file osapi-condvar.h.

**11.161.2.2 `name` char `OS_condvar_prop_t::name[OS_MAX_API_NAME]`**

Definition at line 36 of file osapi-condvar.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-condvar.h`

**11.162 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`

**11.162.1 Detailed Description**

OSAL counting semaphore properties.

Definition at line 32 of file osapi-countsem.h.

**11.162.2 Field Documentation****11.162.2.1 `creator` `osal_id_t` `OS_count_sem_prop_t::creator`**

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

**11.162.2.2 `name` char `OS_count_sem_prop_t::name[OS_MAX_API_NAME]`**

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

**11.162.2.3 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](#)

## 11.163 os\_dirent\_t Struct Reference

Directory entry.

```
#include <osapi-dir.h>
```

### Data Fields

- char [FileName \[OS\\_MAX\\_FILE\\_NAME\]](#)

#### 11.163.1 Detailed Description

Directory entry.

Definition at line 32 of file osapi-dir.h.

#### 11.163.2 Field Documentation

##### 11.163.2.1 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](#)

## 11.164 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\]](#)

#### 11.164.1 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.

#### 11.164.2 Field Documentation

**11.164.2.1 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](#)

## 11.165 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`

#### 11.165.1 Detailed Description

OSAL file properties.

Definition at line 49 of file osapi-file.h.

#### 11.165.2 Field Documentation

**11.165.2.1 IsValid** `uint8 OS_file_prop_t::IsValid`

Definition at line 53 of file osapi-file.h.

**11.165.2.2 Path** `char OS_file_prop_t::Path[OS_MAX_PATH_LEN]`

Definition at line 51 of file osapi-file.h.

**11.165.2.3 User** `osal_id_t OS_file_prop_t::User`

Definition at line 52 of file osapi-file.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-file.h](#)

## 11.166 os\_fsinfo\_t Struct Reference

OSAL file system info.

```
#include <osapi-filesystem.h>
```

### Data Fields

- `uint32 MaxFds`  
*Total number of file descriptors.*
- `uint32 FreeFds`  
*Total number that are free.*
- `uint32 MaxVolumes`  
*Maximum number of volumes.*
- `uint32 FreeVolumes`  
*Total number of volumes free.*

### 11.166.1 Detailed Description

OSAL file system info.

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

### 11.166.2 Field Documentation

#### 11.166.2.1 FreeFds `uint32` `os_fsinfo_t::FreeFds`

Total number that are free.

Definition at line 38 of file osapi-filesystems.h.

#### 11.166.2.2 FreeVolumes `uint32` `os_fsinfo_t::FreeVolumes`

Total number of volumes free.

Definition at line 40 of file osapi-filesystems.h.

#### 11.166.2.3 MaxFds `uint32` `os_fsinfo_t::MaxFds`

Total number of file descriptors.

Definition at line 37 of file osapi-filesystems.h.

#### 11.166.2.4 MaxVolumes `uint32` `os_fsinfo_t::MaxVolumes`

Maximum number of volumes.

Definition at line 39 of file osapi-filesystems.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-filesystems.h`

## 11.167 **os\_fstat\_t** Struct Reference

File system status.

```
#include <osapi-file.h>
```

### Data Fields

- `uint32 FileModeBits`
- `OS_time_t FileTime`
- `size_t FileSize`

### 11.167.1 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.

## 11.167.2 Field Documentation

**11.167.2.1 FileModeBits** `uint32 os_fstat_t::FileModeBits`  
Definition at line 66 of file osapi-file.h.

**11.167.2.2 FileSize** `size_t os_fstat_t::FileSize`  
Definition at line 68 of file osapi-file.h.

**11.167.2.3 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`

## 11.168 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`

### 11.168.1 Detailed Description

OSAL heap properties.

See also

[OS\\_HeapGetInfo\(\)](#)

Definition at line 36 of file osapi-heap.h.

### 11.168.2 Field Documentation

**11.168.2.1 free\_blocks** `osal_blockcount_t OS_heap_prop_t::free_blocks`  
Definition at line 39 of file osapi-heap.h.

**11.168.2.2 free\_bytes** `size_t OS_heap_prop_t::free_bytes`  
Definition at line 38 of file osapi-heap.h.

**11.168.2.3 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`

## 11.169 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`

### 11.169.1 Detailed Description

OSAL module address properties.

Definition at line 78 of file osapi-module.h.

### 11.169.2 Field Documentation

#### 11.169.2.1 `bss_address` `cpuaddr` OS\_module\_address\_t::bss\_address

Definition at line 86 of file osapi-module.h.

#### 11.169.2.2 `bss_size` `cpuaddr` OS\_module\_address\_t::bss\_size

Definition at line 87 of file osapi-module.h.

#### 11.169.2.3 `code_address` `cpuaddr` OS\_module\_address\_t::code\_address

Definition at line 82 of file osapi-module.h.

#### 11.169.2.4 `code_size` `cpuaddr` OS\_module\_address\_t::code\_size

Definition at line 83 of file osapi-module.h.

#### 11.169.2.5 `data_address` `cpuaddr` OS\_module\_address\_t::data\_address

Definition at line 84 of file osapi-module.h.

#### 11.169.2.6 `data_size` `cpuaddr` OS\_module\_address\_t::data\_size

Definition at line 85 of file osapi-module.h.

#### 11.169.2.7 `flags` `uint32` OS\_module\_address\_t::flags

Definition at line 81 of file osapi-module.h.

**11.169.2.8 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`

## 11.170 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`

### 11.170.1 Detailed Description

OSAL module properties.

Definition at line 91 of file osapi-module.h.

### 11.170.2 Field Documentation

**11.170.2.1 addr** `OS_module_address_t OS_module_prop_t::addr`

Definition at line 97 of file osapi-module.h.

**11.170.2.2 entry\_point** `cpuaddr OS_module_prop_t::entry_point`

Definition at line 93 of file osapi-module.h.

**11.170.2.3 filename** `char OS_module_prop_t::filename[OS_MAX_PATH_LEN]`

Definition at line 95 of file osapi-module.h.

**11.170.2.4 host\_module\_id** `cpuaddr OS_module_prop_t::host_module_id`

Definition at line 94 of file osapi-module.h.

**11.170.2.5 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`

## 11.171 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`

**11.171.1 Detailed Description**

OSAL mutex properties.

Definition at line 32 of file osapi-mutex.h.

**11.171.2 Field Documentation****11.171.2.1 creator `osal_id_t` `OS_mut_sem_prop_t::creator`**

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

**11.171.2.2 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`

**11.172 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`

**11.172.1 Detailed Description**

OSAL queue properties.

Definition at line 32 of file osapi-queue.h.

**11.172.2 Field Documentation****11.172.2.1 creator `osal_id_t` `OS_queue_prop_t::creator`**

Definition at line 35 of file osapi-queue.h.

**11.172.2.2 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`

## 11.173 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.*

### 11.173.1 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.

### 11.173.2 Field Documentation

#### 11.173.2.1 ActualLength size\_t OS\_SockAddr\_t::ActualLength

Length of the actual address data.

Definition at line 111 of file osapi-sockets.h.

#### 11.173.2.2 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

## 11.174 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.*

### 11.174.1 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.

## 11.174.2 Field Documentation

### 11.174.2.1 AlignPtr void\* OS\_SockAddrData\_t::AlignPtr

Ensures pointer alignment.

Definition at line 99 of file osapi-sockets.h.

### 11.174.2.2 AlignU32 uint32 OS\_SockAddrData\_t::AlignU32

Ensures uint32 alignment.

Definition at line 98 of file osapi-sockets.h.

### 11.174.2.3 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](#)

## 11.175 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.*

### 11.175.1 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.

## 11.175.2 Field Documentation

### 11.175.2.1 **creator** [osal\\_id\\_t](#) OS\_socket\_prop\_t::creator

OSAL TaskID which opened the socket.

Definition at line 125 of file osapi-sockets.h.

### 11.175.2.2 **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](#)

## 11.176 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](#)

#### 11.176.1 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.

#### 11.176.2 Field Documentation

##### 11.176.2.1 Address void(\* OS\_static\_symbol\_record\_t::Address) (void)

Definition at line 116 of file osapi-module.h.

##### 11.176.2.2 Module const char\* OS\_static\_symbol\_record\_t::Module

Definition at line 117 of file osapi-module.h.

##### 11.176.2.3 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](#)

## 11.177 OS\_statvfs\_t Struct Reference

```
#include <osapi-filesystem.h>
```

### Data Fields

- size\_t [block\\_size](#)
- [osal\\_blockcount\\_t](#) [total\\_blocks](#)
- [osal\\_blockcount\\_t](#) [blocks\\_free](#)

#### 11.177.1 Detailed Description

Definition at line 49 of file osapi-filesystem.h.

## 11.177.2 Field Documentation

### 11.177.2.1 **block\_size** size\_t OS\_statvfs\_t::block\_size

Block size of underlying FS

Definition at line 51 of file osapi-filesystems.h.

### 11.177.2.2 **blocks\_free** osal\_blockcount\_t OS\_statvfs\_t::blocks\_free

Available blocks in underlying FS

Definition at line 53 of file osapi-filesystems.h.

### 11.177.2.3 **total\_blocks** osal\_blockcount\_t OS\_statvfs\_t::total\_blocks

Total blocks in underlying FS

Definition at line 52 of file osapi-filesystems.h.

The documentation for this struct was generated from the following file:

- osal/src/os/inc/[osapi-filesystems.h](#)

## 11.178 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**

### 11.178.1 Detailed Description

OSAL task properties.

Definition at line 57 of file osapi-task.h.

## 11.178.2 Field Documentation

### 11.178.2.1 **creator** [osal\\_id\\_t](#) OS\_task\_prop\_t::creator

Definition at line 60 of file osapi-task.h.

### 11.178.2.2 **name** char OS\_task\_prop\_t::name[OS\_MAX\_API\_NAME]

Definition at line 59 of file osapi-task.h.

### 11.178.2.3 **priority** [osal\\_priority\\_t](#) OS\_task\_prop\_t::priority

Definition at line 62 of file osapi-task.h.

**11.178.2.4 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](#)

## 11.179 OS\_time\_t Struct Reference

OSAL time interval structure.

```
#include <osapi-clock.h>
```

### Data Fields

- [int64 ticks](#)

#### 11.179.1 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.

#### 11.179.2 Field Documentation

##### 11.179.2.1 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](#)

## 11.180 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](#)

### 11.180.1 Detailed Description

Time base properties.

Definition at line 37 of file osapi-timebase.h.

### 11.180.2 Field Documentation

#### 11.180.2.1 accuracy `uint32` OS\_timebase\_prop\_t::accuracy

Definition at line 43 of file osapi-timebase.h.

#### 11.180.2.2 creator `osal_id_t` OS\_timebase\_prop\_t::creator

Definition at line 40 of file osapi-timebase.h.

#### 11.180.2.3 freerun\_time `uint32` OS\_timebase\_prop\_t::freerun\_time

Definition at line 42 of file osapi-timebase.h.

#### 11.180.2.4 name `char` OS\_timebase\_prop\_t::name[OS\_MAX\_API\_NAME]

Definition at line 39 of file osapi-timebase.h.

#### 11.180.2.5 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](#)

## 11.181 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

### 11.181.1 Detailed Description

Timer properties.

Definition at line 37 of file osapi-timer.h.

### 11.181.2 Field Documentation

**11.181.2.1 accuracy** `uint32 OS_timer_prop_t::accuracy`

Definition at line 43 of file osapi-timer.h.

**11.181.2.2 creator** `osal_id_t OS_timer_prop_t::creator`

Definition at line 40 of file osapi-timer.h.

**11.181.2.3 interval\_time** `uint32 OS_timer_prop_t::interval_time`

Definition at line 42 of file osapi-timer.h.

**11.181.2.4 name** `char OS_timer_prop_t::name[OS_MAX_API_NAME]`

Definition at line 39 of file osapi-timer.h.

**11.181.2.5 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/cs/docs/dox\\_src/cfs\\_cs.dox](#) File Reference

### 12.2 [apps/cs/fsw/inc/cs\\_events.h](#) File Reference

#### Macros

- `#define CS_INIT_INF_EID 1`  
*CS Initialization Event ID.*
- `#define CS_NOOP_INF_EID 2`  
*CS No-op Command Event ID.*
- `#define CS_RESET_DBG_EID 3`  
*CS Reset Counters Command Event ID.*
- `#define CS_DISABLE_ALL_INF_EID 4`  
*CS Disable Checksumming Command Event ID.*
- `#define CS_ENABLE_ALL_INF_EID 5`  
*CS Enable Checksumming Command Event ID.*
- `#define CS_DISABLE_CFECORE_INF_EID 6`  
*CS Disable cFE Core Checksumming Command Event ID.*
- `#define CS_ENABLE_CFECORE_INF_EID 7`  
*CS Enable cFE Core Checksumming Command Event ID.*
- `#define CS_DISABLE_OS_INF_EID 8`  
*CS Disable OS Code Checksumming Command Event ID.*
- `#define CS_ENABLE_OS_INF_EID 9`  
*CS Enable OS Code Checksumming Command Event ID.*
- `#define CS_BASELINE_CFECORE_INF_EID 10`  
*CS Baseline cFE Core Checksum Event ID.*

- #define CS\_NO\_BASELINE\_CFECORE\_INF\_EID 11  
*CS Baseline cFE Core Checksum Pending Event ID.*
- #define CS\_BASELINE\_OS\_INF\_EID 12  
*CS Baseline OS Code Checksum Event ID.*
- #define CS\_NO\_BASELINE\_OS\_INF\_EID 13  
*CS Baseline OS Code Checksum Pending Event ID.*
- #define CS\_RECOMPUTE\_CFECORE\_STARTED\_DBG\_EID 14  
*CS cFE Core Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_CFECORE\_CREATE\_CHDTASK\_ERR\_EID 15  
*CS cFE Core Checksum Failed Child Task Create Event ID.*
- #define CS\_RECOMPUTE\_CFECORE\_CHDTASK\_ERR\_EID 16  
*CS cFE Core Checksum Failed Child Task In Use Event ID.*
- #define CS\_RECOMPUTE\_OS\_STARTED\_DBG\_EID 17  
*CS OS Code Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_OS\_CREATE\_CHDTASK\_ERR\_EID 18  
*CS OS Code Checksum Recompute Failed Child Task Create Event ID.*
- #define CS\_RECOMPUTE\_OS\_CHDTASK\_ERR\_EID 19  
*CS OS Code Checksum Recompute Failed Child Task In Use Event ID.*
- #define CS\_ONESHOT\_STARTED\_DBG\_EID 20  
*CS Oneshot Checksum Started Event ID.*
- #define CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID 21  
*CS Oneshot Checksum Failed Child Task Create Event ID.*
- #define CS\_ONESHOT\_CHDTASK\_ERR\_EID 22  
*CS Oneshot Checksum Failed Child Task In Use Event ID.*
- #define CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID 23  
*CS Oneshot Checksum Failed Invalid Memory Range Event ID.*
- #define CS\_ONESHOT\_CANCELLED\_INF\_EID 24  
*CS Oneshot Checksum Cancelled Event ID.*
- #define CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID 25  
*CS Oneshot Checksum Failed Child Task Delete Event ID.*
- #define CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID 26  
*CS Oneshot Checksum Cancel Failed No Oneshot Active Event ID.*
- #define CS\_EEPROM\_MISCOMPARE\_ERR\_EID 27  
*CS EEPROM Checksum Miscompare Event ID.*
- #define CS\_MEMORY\_MISCOMPARE\_ERR\_EID 28  
*CS Memory Checksum Miscompare Event ID.*
- #define CS\_TABLES\_MISCOMPARE\_ERR\_EID 29  
*CS Table Checksum Miscompare Event ID.*
- #define CS\_APP\_MISCOMPARE\_ERR\_EID 30  
*CS App Checksum Miscompare Event ID.*
- #define CS\_CFECORE\_MISCOMPARE\_ERR\_EID 31  
*CS cFE Core Checksum Miscompare Event ID.*
- #define CS\_OS\_MISCOMPARE\_ERR\_EID 32  
*CS OS Code Checksum Miscompare Event ID.*
- #define CS\_MID\_ERR\_EID 33  
*CS Invalid Command Pipe Message ID Event ID.*
- #define CS\_CC1\_ERR\_EID 34

- #define CS\_INVALID\_GROUND\_COMMAND\_CODE\_EVENT\_ID
- #define CS\_EXIT\_ERR\_EID 35
  - CS App Termination Event ID.
- #define CS\_LEN\_ERR\_EID 36
  - CS Invalid Message Length Event ID.
- #define CS\_DISABLE\_EEPROM\_INF\_EID 37
  - CS Disable EEPROM Checksumming Command Event ID.
- #define CS\_ENABLE\_EEPROM\_INF\_EID 38
  - CS Enable EEPROM Checksumming Command Event ID.
- #define CS\_BASELINE\_EEPROM\_INF\_EID 39
  - CS Baseline EEPROM Checksum Event ID.
- #define CS\_NO\_BASELINE\_EEPROM\_INF\_EID 40
  - CS Baseline EEPROM Checksum Pending Event ID.
- #define CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID 41
  - CS Baseline EEPROM Checksum Failed Invalid Entry ID Event ID.
- #define CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID 42
  - CS EEPROM Checksum Recompute Started Event ID.
- #define CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID 43
  - CS EEPROM Checksum Recompute Failed Child Task Create Event ID.
- #define CS\_RECOMPUTE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID 44
  - CS EEPROM Checksum Recompute Failed Invalid Entry ID Event ID.
- #define CS\_RECOMPUTE\_EEPROM\_CHDTASK\_ERR\_EID 45
  - CS EEPROM Checksum Recompute Failed Child Task In Use Event ID.
- #define CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID 46
  - CS EEPROM Checksum Entry ID Enabled Event ID.
- #define CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID 47
  - CS EEPROM Checksum Entry ID Enable Failed Invalid Entry ID Event ID.
- #define CS\_DISABLE\_EEPROM\_ENTRY\_INF\_EID 48
  - CS EEPROM Checksum Entry ID Disabled Event ID.
- #define CS\_DISABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID 49
  - CS EEPROM Checksum Entry ID Disable Failed Invalid Entry ID Event ID.
- #define CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID 50
  - CS EEPROM Get Entry ID Event ID.
- #define CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID 51
  - CS EEPROM Get Entry ID Address Not Found Event ID.
- #define CS\_DISABLE\_MEMORY\_INF\_EID 52
  - CS Disable Memory Checksumming Command Event ID.
- #define CS\_ENABLE\_MEMORY\_INF\_EID 53
  - CS Enable Memory Checksumming Command Event ID.
- #define CS\_BASELINE\_MEMORY\_INF\_EID 54
  - CS Baseline Memory Checksum Event ID.
- #define CS\_NO\_BASELINE\_MEMORY\_INF\_EID 55
  - CS Baseline Memory Checksum Pending Event ID.
- #define CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID 56
  - CS Baseline Memory Checksum Failed Invalid Entry ID Event ID.
- #define CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID 57
  - CS Memory Checksum Recompute Started Event ID.

- #define CS\_RECOMPUTE\_MEMORY\_CREATE\_CHDTASK\_ERR\_EID 58  
*CS Memory Checksum Recompute Failed Child Task Create Event ID.*
- #define CS\_RECOMPUTE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID 59  
*CS Memory Checksum Recompute Failed Invalid Entry ID Event ID.*
- #define CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR\_EID 60  
*CS Memory Checksum Recompute Failed Child Task In Use Event ID.*
- #define CS\_ENABLE\_MEMORY\_ENTRY\_INF\_EID 61  
*CS Memory Checksum Entry ID Enabled Event ID.*
- #define CS\_ENABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID 62  
*CS Memory Checksum Entry ID Enable Failed Invalid Entry ID Event ID.*
- #define CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID 63  
*CS Memory Checksum Entry ID Disabled Event ID.*
- #define CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID 64  
*CS Memory Checksum Entry ID Disable Failed Invalid Entry ID Event ID.*
- #define CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID 65  
*CS Memory Get Entry ID Event ID.*
- #define CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID 66  
*CS Memory Get Entry ID Address Not Found Event ID.*
- #define CS\_DISABLE\_TABLES\_INF\_EID 67  
*CS Disable Tables Checksumming Command Event ID.*
- #define CS\_ENABLE\_TABLES\_INF\_EID 68  
*CS Enable Tables Checksumming Command Event ID.*
- #define CS\_BASELINE\_TABLES\_INF\_EID 69  
*CS Baseline Table Checksum Event ID.*
- #define CS\_NO\_BASELINE\_TABLES\_INF\_EID 70  
*CS Baseline Table Checksum Pending Event ID.*
- #define CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID 71  
*CS Baseline Table Checksum Failed Table Not Found Event ID.*
- #define CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID 72  
*CS Table Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID 73  
*CS Table Checksum Recompute Failed Child Task Create Event ID.*
- #define CS\_RECOMPUTE\_UNKNOWN\_NAME\_TABLES\_ERR\_EID 74  
*CS Table Checksum Recompute Failed Table Not Found Event ID.*
- #define CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID 75  
*CS Table Checksum Recompute Failed Child Task In Use Event ID.*
- #define CS\_ENABLE\_TABLES\_NAME\_INF\_EID 76  
*CS Table Checksum Enabled Event ID.*
- #define CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID 77  
*CS Table Checksum Failed Table Not Found Event ID.*
- #define CS\_DISABLE\_TABLES\_NAME\_INF\_EID 78  
*CS Table Checksum Disabled Event ID.*
- #define CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID 79  
*CS Table Checksum Disable Failed Table Not Found Event ID.*
- #define CS\_DISABLE\_APP\_INF\_EID 80  
*CS Disable Apps Checksumming Command Event ID.*
- #define CS\_ENABLE\_APP\_INF\_EID 81

- #define CS\_BASELINE\_APP\_INF\_EID 82  
*CS Baseline App Checksum Event ID.*
- #define CS\_NO\_BASELINE\_APP\_INF\_EID 83  
*CS Baseline App Checksum Pending Event ID.*
- #define CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID 84  
*CS Baseline App Checksum Failed App Not Found Event ID.*
- #define CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID 85  
*CS App Checksum Recompute Started Event ID.*
- #define CS\_RECOMPUTE\_APP\_CREATE\_CHDTASK\_ERR\_EID 86  
*CS App Checksum Recompute Failed Create Child Task Event ID.*
- #define CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID 87  
*CS App Checksum Recompute Failed App Not Found Event ID.*
- #define CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID 88  
*CS App Checksum Recompute Failed Child Task In Use Event ID.*
- #define CS\_ENABLE\_APP\_NAME\_INF\_EID 89  
*CS App Checksum Enabled Event ID.*
- #define CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID 90  
*CS App Checksum Enable Failed App Not Found Event ID.*
- #define CS\_DISABLE\_APP\_NAME\_INF\_EID 91  
*CS App Checksum Disabled Event ID.*
- #define CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID 92  
*CS App Checksum Disable Failed App Not Found Event ID.*
- #define CS\_COMPUTE\_APP\_NOT\_FOUND\_ERR\_EID 93  
*CS Apps Table Checksumming Failed App Not Found Event ID.*
- #define CS\_COMPUTE\_TABLES\_NOT\_FOUND\_ERR\_EID 94  
*CS Tables Table Checksumming Failed Table Not Found Event ID.*
- #define CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID 95  
*CS EEPROM or Memory Checksum Recompute Finished Event ID.*
- #define CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID 96  
*CS Table Checksum Recompute Failed Could Not Get Address Event ID.*
- #define CS\_RECOMPUTE\_ERROR\_APP\_ERR\_EID 97  
*CS App Checksum Recompute Failed Could Not Get Address Event ID.*
- #define CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID 98  
*CS Table Recompute Finished Event ID.*
- #define CS\_RECOMPUTE\_FINISH\_APP\_INF\_EID 99  
*CS App Recompute Finished Event ID.*
- #define CS\_ONESHOT\_FINISHED\_INF\_EID 100  
*CS Oneshot Checksum Finished Event ID.*
- #define CS\_VAL\_EEPROM\_STATE\_ERR\_EID 101  
*CS EEPROM Table Validate Failed Illegal State Field Event ID.*
- #define CS\_VAL\_EEPROM\_RANGE\_ERR\_EID 102  
*CS EEPROM Table Validate Failed Illegal Checksum Range Event ID.*
- #define CS\_VAL\_MEMORY\_STATE\_ERR\_EID 103  
*CS Memory Table Valide Failed Illegal State Field Event ID.*
- #define CS\_VAL\_MEMORY\_RANGE\_ERR\_EID 104  
*CS Memory Table Validate Failed Illegal Checksum Range Event ID.*

- #define CS\_VAL\_TABLES\_STATE\_ERR\_EID 105  
*CS Tables Table Validate Failed Illegal State Field Event ID.*
- #define CS\_VAL\_APP\_STATE\_ERR\_EID 106  
*CS Apps Table Validate Failed Illegal State Field Event ID.*
- #define CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID 107  
*CS EEPROM or Memory Table No Valid Entries Event ID.*
- #define CS\_PROCESS\_APP\_NO\_ENTRIES\_INF\_EID 108  
*CS Apps Table No Valid Entries Event ID.*
- #define CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID 109  
*CS Tables Table No Valid Entries Event ID.*
- #define CS\_TBL\_INIT\_ERR\_EID 110  
*CS Table Initialization Failed Event ID.*
- #define CS\_TBL\_UPDATE\_ERR\_EID 111  
*CS Table Update Failed Event ID.*
- #define CS\_INIT\_SB\_CREATE\_ERR\_EID 112  
*CS Software Bus Create Pipe Failed Event ID.*
- #define CS\_INIT\_SB\_SUBSCRIBE\_HK\_ERR\_EID 113  
*CS Software Bus Subscribe To Housekeeping Failed Event ID.*
- #define CS\_INIT\_SB\_SUBSCRIBE\_BACK\_ERR\_EID 114  
*CS Software Bus Subscribe To Background Cycle Failed Event ID.*
- #define CS\_INIT\_SB\_SUBSCRIBE\_CMD\_ERR\_EID 115  
*CS Software Bus Subscribe to Command Failed Event ID.*
- #define CS\_INIT\_EEPROM\_ERR\_EID 116  
*CS EEPROM Table Initialization Failed Event ID.*
- #define CS\_INIT\_MEMORY\_ERR\_EID 117  
*CS Memory Table Initialization Failed Event ID.*
- #define CS\_INIT\_TABLES\_ERR\_EID 118  
*CS Tables Table Initialization Failed Event ID.*
- #define CS\_INIT\_APP\_ERR\_EID 119  
*CS Apps Table Initialization Failed Event ID.*
- #define CS\_COMPUTE\_TABLES\_RELEASE\_ERR\_EID 120  
*CS Table Release Address Failed Event ID.*
- #define CS\_COMPUTE\_TABLES\_ERR\_EID 121  
*CS Get Table Info Failed Event ID.*
- #define CS\_COMPUTE\_APP\_ERR\_EID 122  
*CS Get App Info Failed Event ID.*
- #define CS\_UPDATE\_EEPROM\_ERR\_EID 123  
*CS EEPROM Table Update Failed Event ID.*
- #define CS\_UPDATE\_MEMORY\_ERR\_EID 124  
*CS Memory Table Update Failed Event ID.*
- #define CS\_UPDATE\_TABLES\_ERR\_EID 125  
*CS Tables Table Update Failed Event ID.*
- #define CS\_UPDATE\_APP\_ERR\_EID 126  
*CS Apps Table Update Failed Event ID.*
- #define CS\_OS\_TEXT\_SEG\_INF\_EID 127  
*CS OS Code Checksum Disabled Platform Not Supported Event ID.*
- #define CS\_COMPUTE\_APP\_PLATFORM\_DBG\_EID 128

- #define CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID 129  
*CS Table Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID 130  
*CS Table Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID 131  
*CS App Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID 132  
*CS App Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID 133  
*CS Memory Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_ENABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID 134  
*CS Memory Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_DISABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID 135  
*CS EEPROM Disable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID 136  
*CS EEPROM Enable Checksum Unable To Find Entry In Definition Table Event ID.*
- #define CS\_VAL\_TABLES\_DEF\_TBL\_DUPL\_ERR\_EID 137  
*CS Tables Table Validate Failed Duplicate Name Event ID.*
- #define CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID 138  
*CS Tables Table Validate Failed Illegal State With Empty Name Event ID.*
- #define CS\_VAL\_TABLES\_INF\_EID 139  
*CS Tables Table Verification Results Event ID.*
- #define CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID 140  
*CS Apps Table Validate Failed Duplicate Name Event ID.*
- #define CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID 141  
*CS Apps Table Validate Failed Illegal State With Empty Name Event ID.*
- #define CS\_VAL\_APP\_INF\_EID 142  
*CS Apps Table Verification Results Event ID.*
- #define CS\_VAL\_MEMORY\_INF\_EID 143  
*CS Memory Table Verification Results Event ID.*
- #define CS\_VAL\_EEPROM\_INF\_EID 144  
*CS EEPROM Table Verification Results Event ID.*
- #define CS\_CR\_CDS\_REG\_ERR\_EID 145  
*CS Critical Data Store Registration Failed Event ID.*
- #define CS\_CR\_CDS\_CPY\_ERR\_EID 146  
*CS Critical Data Store Copy Failed Event ID.*
- #define CS\_CR\_CDS\_RES\_ERR\_EID 147  
*CS Critical Data Store Restore Failed Event ID.*
- #define CS\_UPDATE\_CDS\_ERR\_EID 148  
*CS Critical Data Store Update Failed Event ID.*
- #define CS\_EXIT\_INF\_EID 149  
*CS App Termination Event ID.*
- #define CS\_CFE\_TEXT\_SEG\_INF\_EID 150  
*CS cFE Core Checksum Disabled Address Not Available Event ID.*
- #define CS\_CMD\_COMPUTE\_PROG\_ERR\_EID 152  
*CS Command Failed Due To Recompute Or Oneshot In Progress Event ID.*

- #define CS\_BKGND\_COMPUTE\_PROG\_INF\_EID 153  
*CS Skipping Background Cycle Due To Recompute Or Oneshote In Progress Event ID.*
- #define CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID 154  
*CS Apps Table Validate Failed Illegal State With Long Name Event ID.*

### 12.2.1 Detailed Description

Specification for the CFS Checksum event identifiers.

## 12.3 apps/cs/fsw/inc/cs\_mission\_cfg.h File Reference

```
#include <cfef_mission_cfg.h>
```

### Macros

- #define CS\_DEFAULT\_ALGORITHM CFE\_MISSION\_ES\_DEFAULT\_CRC  
*default CRC algorithm*

### 12.3.1 Detailed Description

Specification for the CFS Checksum macro constants that can be configured from one mission to another

## 12.4 apps/cs/fsw/inc/cs\_msg.h File Reference

```
#include <cfef.h>
```

### Data Structures

- struct **CS\_HkPacket\_Payload\_t**  
*Housekeeping Payload Structure.*
- struct **CS\_HkPacket\_t**  
*Housekeeping Packet Structure.*
- struct **CS\_GetEntryIDCmd\_Payload\_t**  
*Get entry ID command payload.*
- struct **CS\_EntryCmd\_Payload\_t**  
*Payload for commands using Memory or EEPROM tables.*
- struct **CS\_TableNameCmd\_Payload\_t**  
*Payload for commanding by table name.*
- struct **CS\_AppNameCmd\_Payload\_t**  
*Payload for commanding by app name.*
- struct **CS\_OneShotCmd\_Payload\_t**  
*Payload for sending one shot calculation.*
- struct **CS\_NoArgsCmd\_t**  
*No arguments command data type.*
- struct **CS\_GetEntryIDCmd\_t**  
*Get entry ID command.*
- struct **CS\_EntryCmd\_t**  
*Command type for commands using Memory or EEPROM tables.*
- struct **CS\_TableNameCmd\_t**

- struct `CS_AppNameCmd_t`  
*Command type for commanding by table name.*
- struct `CS_OneShotCmd_t`  
*Command type for commanding by app name.*
- struct `CS_OneShotCmd_t`  
*Command type for sending one shot calculation.*

#### 12.4.1 Detailed Description

Specification for the CFS Checksum command and telemetry messages.

### 12.5 apps/cs/fsw/inc/cs\_msgdefs.h File Reference

```
#include <cfi.h>
```

#### Macros

- #define `CS_NOOP_CC` 0  
*Noop.*
- #define `CS_RESET_CC` 1  
*Reset Counters.*
- #define `CS_ONE_SHOT_CC` 2  
*Start One shot calculation.*
- #define `CS_CANCEL_ONE_SHOT_CC` 3  
*Cancel one shot.*
- #define `CS_ENABLE_ALL_CS_CC` 4  
*Enable Global Checksumming.*
- #define `CS_DISABLE_ALL_CS_CC` 5  
*Disable Global Checksumming.*
- #define `CS_ENABLE_CFE_CORE_CC` 6  
*Enable checksumming of cFE core.*
- #define `CS_DISABLE_CFE_CORE_CC` 7  
*Disable checksumming of cFE core.*
- #define `CS_REPORT_BASELINE_CFE_CORE_CC` 8  
*Report Baseline checksum of cFE core.*
- #define `CS_RECOMPUTE_BASELINE_CFE_CORE_CC` 9  
*Recompute Baseline checksum of cFE core.*
- #define `CS_ENABLE_OS_CC` 10  
*Enable checksumming of OS code segment.*
- #define `CS_DISABLE_OS_CC` 11  
*Disable checksumming of OS code segment.*
- #define `CS_REPORT_BASELINE_OS_CC` 12  
*Report Baseline checksum of OS code segment.*
- #define `CS_RECOMPUTE_BASELINE_OS_CC` 13  
*Recompute Baseline checksum of OS code segment.*
- #define `CS_ENABLE_EEPROM_CC` 14  
*Enable checksumming for EEPROM table.*
- #define `CS_DISABLE_EEPROM_CC` 15  
*Disable checksumming for EEPROM table.*

- #define CS\_REPORT\_BASELINE\_EEPROM\_CC 16  
*Report Baseline checksum of EEPROM Entry.*
- #define CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC 17  
*Recompute Baseline checksum of EEPROM Entry.*
- #define CS\_ENABLE\_ENTRY\_EEPROM\_CC 18  
*Enable checksumming for an EEPROM entry.*
- #define CS\_DISABLE\_ENTRY\_EEPROM\_CC 19  
*Disable checksumming for an EEPROM entry.*
- #define CS\_GET\_ENTRY\_ID\_EEPROM\_CC 20  
*Get the Entry ID for a given EEPROM address.*
- #define CS\_ENABLE\_MEMORY\_CC 21  
*Enable checksumming for Memory table.*
- #define CS\_DISABLE\_MEMORY\_CC 22  
*Disable checksumming for Memory table.*
- #define CS\_REPORT\_BASELINE\_MEMORY\_CC 23  
*Report Baseline checksum of Memory Entry.*
- #define CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC 24  
*Recompute Baseline checksum of Memory Entry.*
- #define CS\_ENABLE\_ENTRY\_MEMORY\_CC 25  
*Enable checksumming for a Memory entry.*
- #define CS\_DISABLE\_ENTRY\_MEMORY\_CC 26  
*Disable checksumming for a Memory entry.*
- #define CS\_GET\_ENTRY\_ID\_MEMORY\_CC 27  
*Get the Entry ID for a given Memory address.*
- #define CS\_ENABLE\_TABLES\_CC 28  
*Enable checksumming for Tables table.*
- #define CS\_DISABLE\_TABLES\_CC 29  
*Disable checksumming for Tables table.*
- #define CS\_REPORT\_BASELINE\_TABLE\_CC 30  
*Report Baseline checksum of a table.*
- #define CS\_RECOMPUTE\_BASELINE\_TABLE\_CC 31  
*Recompute Baseline checksum of a table.*
- #define CS\_ENABLE\_NAME\_TABLE\_CC 32  
*Enable checksumming for a table.*
- #define CS\_DISABLE\_NAME\_TABLE\_CC 33  
*Disable checksumming for a table.*
- #define CS\_ENABLE\_APPS\_CC 34  
*Enable checksumming for App table.*
- #define CS\_DISABLE\_APPS\_CC 35  
*Disable checksumming for App table.*
- #define CS\_REPORT\_BASELINE\_APP\_CC 36  
*Report Baseline checksum of an app.*
- #define CS\_RECOMPUTE\_BASELINE\_APP\_CC 37  
*Recompute Baseline checksum of an app.*
- #define CS\_ENABLE\_NAME\_APP\_CC 38  
*Enable checksumming for an app.*
- #define CS\_DISABLE\_NAME\_APP\_CC 39  
*Disable checksumming for an app.*

*Disable checksumming for an app.*

- `#define CS_ONESHOT_CC CS_CS_ONE_SHOT_CC`
- `#define CS_CANCEL_ONESHOT_CC CS_CS_CANCEL_ONE_SHOT_CC`
- `#define CS_ENABLE_CFECore_CC CS_CS_ENABLE_CFE_CORE_CC`
- `#define CS_DISABLE_CFECore_CC CS_CS_DISABLE_CFE_CORE_CC`
- `#define CS_REPORT_BASELINE_CFECore_CC CS_CS_REPORT_BASELINE_CFE_CORE_CC`
- `#define CS_RECOMPUTE_BASELINE_CFECore_CC CS_CS_RECOMPUTE_BASELINE_CFE_CORE_CC`

### CS Checksum Type Number Definitions

- `#define CS_CFECore 0`  
*cFE Core checksum.*
- `#define CS_OSSCORE 1`  
*OS Core checksum.*
- `#define CS_EEPROM_TABLE 2`  
*EEPROM checksum.*
- `#define CS_MEMORY_TABLE 3`  
*Memory checksum.*
- `#define CS_TABLES_TABLE 4`  
*Tables checksum.*
- `#define CS_APP_TABLE 5`  
*App checksum.*
- `#define CS_NUM_TABLES 6`  
*Number of checksum types.*

### CS Checkum States

- `#define CS_STATE_EMPTY 0x00`  
*Entry unused.*
- `#define CS_STATE_ENABLED 0x01`  
*Entry or table enabled.*
- `#define CS_STATE_DISABLED 0x02`  
*Entry or table disabled.*
- `#define CS_STATE_UNDEFINED 0x03`  
*Entry not found state undefined.*

#### 12.5.1 Detailed Description

Specification for the CFS Checksum command and telemetry messages.

#### 12.5.2 Macro Definition Documentation

##### 12.5.2.1 CS\_APP\_TABLE `#define CS_APP_TABLE 5`

App checksum.

Definition at line 1457 of file cs\_msgdefs.h.

##### 12.5.2.2 CS\_CANCEL\_ONESHOT\_CC `#define CS_CANCEL_ONESHOT_CC CS_CS_CANCEL_ONE_SHOT_CC`

Definition at line 1473 of file cs\_msgdefs.h.

**12.5.2.3 CS\_CFECORE** `#define CS_CFECORE 0`  
cFE Core checksum  
Definition at line 1452 of file cs\_msgdefs.h.

**12.5.2.4 CS\_DISABLE\_CFECORE\_CC** `#define CS_DISABLE_CFECORE_CC CS_DISABLE_CFE_CORE_CC`  
Definition at line 1475 of file cs\_msgdefs.h.

**12.5.2.5 CS\_EEPROM\_TABLE** `#define CS_EEPROM_TABLE 2`  
EEPROM checksum.  
Definition at line 1454 of file cs\_msgdefs.h.

**12.5.2.6 CS\_ENABLE\_CFECORE\_CC** `#define CS_ENABLE_CFECORE_CC CS_ENABLE_CFE_CORE_CC`  
Definition at line 1474 of file cs\_msgdefs.h.

**12.5.2.7 CS\_MEMORY\_TABLE** `#define CS_MEMORY_TABLE 3`  
Memory checksum.  
Definition at line 1455 of file cs\_msgdefs.h.

**12.5.2.8 CS\_NUM\_TABLES** `#define CS_NUM_TABLES 6`  
Number of checksum types.  
Definition at line 1458 of file cs\_msgdefs.h.

**12.5.2.9 CS\_ONESHOT\_CC** `#define CS_ONESHOT_CC CS_ONE_SHOT_CC`  
Definition at line 1472 of file cs\_msgdefs.h.

**12.5.2.10 CS\_OSCORE** `#define CS_OSCORE 1`  
OS Core checksum.  
Definition at line 1453 of file cs\_msgdefs.h.

**12.5.2.11 CS\_RECOMPUTE\_BASELINE\_CFECORE\_CC** `#define CS_RECOMPUTE_BASELINE_CFECORE_CC CS_RECOMPUTE_BASELINE_CFE_CORE_CC`  
Definition at line 1477 of file cs\_msgdefs.h.

**12.5.2.12 CS\_REPORT\_BASELINE\_CFECORE\_CC** `#define CS_REPORT_BASELINE_CFECORE_CC CS_REPORT_BASELINE_CFE_CORE_CC`  
Definition at line 1476 of file cs\_msgdefs.h.

**12.5.2.13 CS\_STATE\_DISABLED** `#define CS_STATE_DISABLED 0x02`  
Entry or table disabled.  
Definition at line 1467 of file cs\_msgdefs.h.

**12.5.2.14 CS\_STATE\_EMPTY** #define CS\_STATE\_EMPTY 0x00  
Entry unused.  
Definition at line 1465 of file cs\_msgdefs.h.

**12.5.2.15 CS\_STATE\_ENABLED** #define CS\_STATE\_ENABLED 0x01  
Entry or table enabled.  
Definition at line 1466 of file cs\_msgdefs.h.

**12.5.2.16 CS\_STATE\_UNDEFINED** #define CS\_STATE\_UNDEFINED 0x03  
Entry not found state undefined.  
Definition at line 1468 of file cs\_msgdefs.h.

**12.5.2.17 CS\_TABLES\_TABLE** #define CS\_TABLES\_TABLE 4  
Tables checksum.  
Definition at line 1456 of file cs\_msgdefs.h.

## 12.6 apps/cs/fsw/inc/cs\_msgids.h File Reference

### Macros

- #define **CS\_CMD\_MID** (0x189F)  
*CS Command Message ID.*
- #define **CS\_SEND\_HK\_MID** (0x18A0)  
*CS Housekeeping Request Message ID.*
- #define **CS\_BACKGROUND\_CYCLE\_MID** (0x18A1)  
*CS Background Cycle Message ID.*
- #define **CS\_HK\_TLM\_MID** (0x08A4)  
*CS Housekeeping Telemetry Message ID.*

### 12.6.1 Detailed Description

Specification for the CFS Checksum constants for message IDs

## 12.7 apps/cs/fsw/inc/cs\_perfids.h File Reference

### Macros

- #define **CS\_APPMAIN\_PERF\_ID** 29  
*Main application performance ID.*

### 12.7.1 Detailed Description

Specification for the CFS Checksum performance ids.

## 12.8 apps/cs/fsw/inc/cs\_platform\_cfg.h File Reference

```
#include <cfe_platform_cfg.h>
```

## Macros

- `#define CS_DEF_EEPROM_TABLE_FILENAME "/cf/cs_eepromtbl.tbl"`  
*EEPROM File Table – default table filename.*
- `#define CS_DEF_MEMORY_TABLE_FILENAME "/cf/cs_memorytbl.tbl"`  
*Memory Address File Table – default table filename.*
- `#define CS_DEF_TABLES_TABLE_FILENAME "/cf/cs_tablestbl.tbl"`  
*Tables File Table – default table filename.*
- `#define CS_DEF_APP_TABLE_FILENAME "/cf/cs_apptbl.tbl"`  
*Application File Table – default table filename.*
- `#define CS_PIPE_DEPTH (3 * CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT)`  
*Application Pipe Depth.*
- `#define CS_MAX_NUM EEPROM_TABLE_ENTRIES 16`  
*Maximum number of entries in the EEPROM table to checksum.*
- `#define CS_MAX_NUM_MEMORY_TABLE_ENTRIES 16`  
*Maximum number of entries in the Memory table to checksum.*
- `#define CS_MAX_NUM_TABLES_TABLE_ENTRIES 24`  
*Maximum number of tables to checksum.*
- `#define CS_MAX_NUM_APP_TABLE_ENTRIES 24`  
*Maximum number of applications to checksum.*
- `#define CS_DEFAULT_BYTES_PER_CYCLE (1024 * 16)`  
*Default number of bytes to checksum per cycle.*
- `#define CS_CHILD_TASK_PRIORITY 200`  
*CS Child Task Priority.*
- `#define CS_CHILD_TASK_DELAY 1000`  
*Delay between checksumming cycles for child task.*
- `#define CS_STARTUP_TIMEOUT 60000`  
*Timeout for waiting for other apps to start.*
- `#define CS_OSCS_CHECKSUM_STATE CS_STATE_ENABLED`  
*Desired state of the checksumming of OS code segment at power on.*
- `#define CS_CFECORE_CHECKSUM_STATE CS_STATE_ENABLED`  
*Desired state of the checksumming of CFE core checksum at power on.*
- `#define CS_EEPROM_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the EEPROM table at power on.*
- `#define CS_MEMORY_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the Memory table at power on.*
- `#define CS_APPS_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the Applications table at power on.*
- `#define CS_TABLES_TBL_POWERON_STATE CS_STATE_ENABLED`  
*Desired state of the Tables table at power on.*
- `#define CS_PRESERVE_STATES_ON_PROCESSOR_RESET true`  
*Whether to preserve checksum states on processor reset.*
- `#define CS_CDS_NAME "CS_CDS"`  
*Name of the Critical Data Store Used for CS.*
- `#define CS_MISSION_REV 0`  
*Mission specific version number for CS application.*

### 12.8.1 Detailed Description

Specification for the CFS Checksum constants that can be configured from one platform to another

## 12.9 apps/cs/fsw/inc/cs\_tbldefs.h File Reference

```
#include <cfef.h>
```

### Data Structures

- struct [CS\\_Def\\_EepromMemory\\_Table\\_Entry\\_t](#)  
*Data structure for the EEPROM or Memory definition table.*
- struct [CS\\_Res\\_EepromMemory\\_Table\\_Entry\\_t](#)  
*Data structure for the Eeprom or Memory results table.*
- struct [CS\\_Def\\_Tables\\_Table\\_Entry\\_t](#)  
*Data structure for the Tables definition table.*
- struct [CS\\_Def\\_App\\_Table\\_Entry\\_t](#)  
*Data structure for the App definition table.*
- struct [CS\\_Res\\_Tables\\_Table\\_Entry\\_t](#)  
*Data structure for the Tables result table.*
- struct [CS\\_Res\\_App\\_Table\\_Entry\\_t](#)  
*Data structure for the app result table.*

### Macros

- `#define CS_DEF_EEPROM_TABLE_NAME "DefEepromTbl"`  
*table names for definition tables*
- `#define CS_DEF_MEMORY_TABLE_NAME "DefMemoryTbl"`
- `#define CS_DEF_TABLES_TABLE_NAME "DefTablesTbl"`
- `#define CS_DEF_APP_TABLE_NAME "DefAppTbl"`
  
- `#define CS_RESULTS_EEPROM_TABLE_NAME "ResEepromTbl"`  
*names for the results tables*
- `#define CS_RESULTS_MEMORY_TABLE_NAME "ResMemoryTbl"`
- `#define CS_RESULTS_TABLES_TABLE_NAME "ResTablesTbl"`
- `#define CS_RESULTS_APP_TABLE_NAME "ResAppTbl"`

### Functions

- `CFE_Status_t CS_ValidateEepromChecksumDefinitionTable (void *TblPtr)`  
*Validate EEPROM definition table.*
- `CFE_Status_t CS_ValidateMemoryChecksumDefinitionTable (void *TblPtr)`  
*Validate Memory definition table.*
- `CFE_Status_t CS_ValidateTablesChecksumDefinitionTable (void *TblPtr)`  
*Validate Tables definition table.*
- `CFE_Status_t CS_ValidateAppChecksumDefinitionTable (void *TblPtr)`  
*Validate App definition table.*

- void `CS_ProcessNewEepromMemoryDefinitionTable` (const `CS_Def_EepromMemory_Table_Entry_t` \*DefinitionTblPtr, const `CS_Res_EepromMemory_Table_Entry_t` \*ResultsTblPtr, const `uint16` NumEntries, const `uint16` Table)
 

*Processes a new definition table for EEPROM or Memory tables.*
- void `CS_ProcessNewTablesDefinitionTable` (const `CS_Def_Tables_Table_Entry_t` \*DefinitionTblPtr, const `CS_Res_Tables_Table_Entry_t` \*ResultsTblPtr)
 

*Processes a new definition table for the Tables table.*
- void `CS_ProcessNewAppDefinitionTable` (const `CS_Def_App_Table_Entry_t` \*DefinitionTblPtr, const `CS_Res_App_Table_Entry_t` \*ResultsTblPtr)
 

*Processes a new definition table for the App table.*
- `CFE_Status_t CS_TableInit` (`CFE_TBL_Handle_t` \*DefinitionTableHandle, `CFE_TBL_Handle_t` \*ResultsTableHandle, void \*DefinitionTblPtr, void \*ResultsTblPtr, const `uint16` Table, const char \*DefinitionTableName, const char \*ResultsTableName, const `uint16` NumEntries, const char \*DefinitionTableFileName, const void \*DefaultDefTableAddress, const `uint16` SizeofDefinitionTableEntry, const `uint16` SizeofResultsTableEntry, const `CFE_TBL_CallbackFuncPtr_t` CallBackFunction)
 

*Initializes the results and definition table on startup.*
- `CFE_Status_t CS_HandleTableUpdate` (void \*DefinitionTblPtr, void \*ResultsTblPtr, const `CFE_TBL_Handle_t` DefinitionTableHandle, const `CFE_TBL_Handle_t` ResultsTableHandle, const `uint16` Table, const `uint16` NumEntries)
 

*Handles table updates for all CS tables.*

### 12.9.1 Detailed Description

Specification for the CFS Checksum tables.

### 12.9.2 Macro Definition Documentation

**12.9.2.1 CS\_DEF\_APP\_TABLE\_NAME** `#define CS_DEF_APP_TABLE_NAME "DefAppTbl"`  
 Definition at line 47 of file cs\_tbldefs.h.

**12.9.2.2 CS\_DEF\_EEPROM\_TABLE\_NAME** `#define CS_DEF_EEPROM_TABLE_NAME "DefEepromTbl"`  
 table names for definition tables  
 Definition at line 44 of file cs\_tbldefs.h.

**12.9.2.3 CS\_DEF\_MEMORY\_TABLE\_NAME** `#define CS_DEF_MEMORY_TABLE_NAME "DefMemoryTbl"`  
 Definition at line 45 of file cs\_tbldefs.h.

**12.9.2.4 CS\_DEF\_TABLES\_TABLE\_NAME** `#define CS_DEF_TABLES_TABLE_NAME "DefTablesTbl"`  
 Definition at line 46 of file cs\_tbldefs.h.

**12.9.2.5 CS\_RESULTS\_APP\_TABLE\_NAME** `#define CS_RESULTS_APP_TABLE_NAME "ResAppTbl"`  
 Definition at line 57 of file cs\_tbldefs.h.

**12.9.2.6 CS\_RESULTS\_EEPROM\_TABLE\_NAME** #define CS\_RESULTS\_EEPROM\_TABLE\_NAME "ResEepromTbl"  
names for the results tables  
Definition at line 54 of file cs\_tbldefs.h.

**12.9.2.7 CS\_RESULTS\_MEMORY\_TABLE\_NAME** #define CS\_RESULTS\_MEMORY\_TABLE\_NAME "ResMemoryTbl"  
Definition at line 55 of file cs\_tbldefs.h.

**12.9.2.8 CS\_RESULTS\_TABLES\_TABLE\_NAME** #define CS\_RESULTS\_TABLES\_TABLE\_NAME "ResTablesTbl"  
Definition at line 56 of file cs\_tbldefs.h.

### 12.9.3 Function Documentation

**12.9.3.1 CS\_HandleTableUpdate()** [CFE\\_Status\\_t](#) CS\_HandleTableUpdate (

```
void * DefinitionTblPtr,
void * ResultsTblPtr,
const CFE_TBL_Handle_t DefinitionTableHandle,
const CFE_TBL_Handle_t ResultsTableHandle,
const uint16 Table,
const uint16 NumEntries )
```

Handles table updates for all CS tables.

#### Description

Completes the handshake with Table Services that releases Addresses for the tables and checks for updates

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>DefinitionTblPtr</i>	A pointer to the definition table that we are operating on
in	<i>ResultsTblPtr</i>	A pointer to the result table to operate on
in	<i>DefinitionTableHandle</i>	A table handle to the definition table
in	<i>ResultsTableHandle</i>	A table handle to the results table
in	<i>NumEntries</i>	The number of entries in the table
in	<i>Table</i>	The specific table we are operating on

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

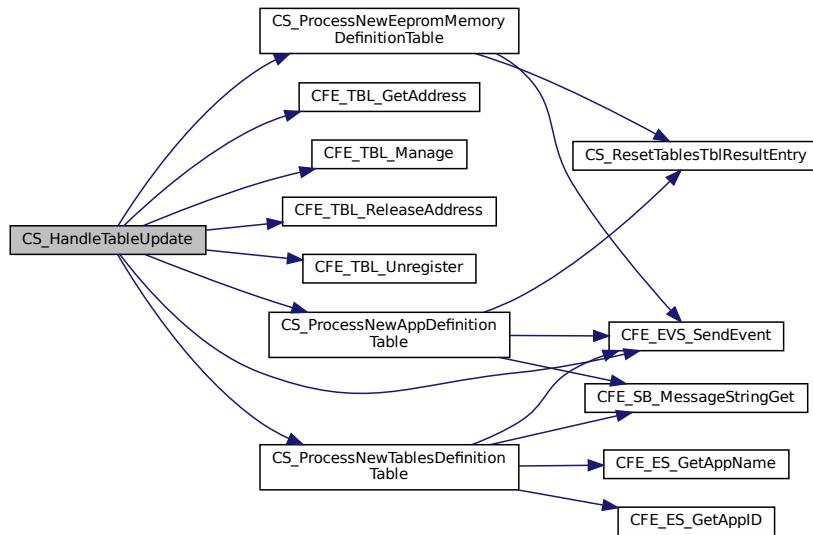
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 957 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_GetAddress(), CFE\_TBL\_INFO\_UPDATED, CFE\_TBL\_Manage(), CFE\_TBL\_ReleaseAddress(), CFE\_TBL\_Unregister(), CS\_APP\_TABLE, CS\_AppData, CS\_EEPROM\_TABLE, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), CS\_TABLES\_TABLE, CS\_TABLETYPE\_NAME\_SIZE, CS\_TBL\_UPDATE\_ERR\_EID, CS\_Res\_Tables\_Table\_Entry\_t::IsCSOwner, CS\_AppData\_t::ResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TblHandle.

Referenced by CS\_HandleRoutineTableUpdates().

Here is the call graph for this function:



**12.9.3.2 CS\_ProcessNewAppDefinitionTable()** void CS\_ProcessNewAppDefinitionTable ( const `CS_Def_App_Table_Entry_t` \* DefinitionTblPtr, const `CS_Res_App_Table_Entry_t` \* ResultsTblPtr )

Processes a new definition table for the App table.

#### Description

Copies data from the definition table to the results table because the results table is where CS keeps all of its checksum data

#### Assumptions, External Events, and Notes:

None

#### Parameters

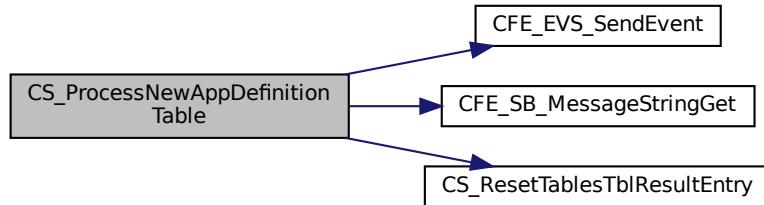
in	<code>DefinitionTblPtr</code>	A pointer to the definition table ( <code>CS_Def_App_Table_Entry_t</code> ) that we are operating on
in	<code>ResultsTblPtr</code>	A pointer to the result table ( <code>CS_Res_App_Table_Entry_t</code> ) to operate on

Definition at line 741 of file cs\_table\_processing.c.

References CS\_HkPacket\_Payload\_t::AppCSSState, CS\_AppData\_t::AppResTablesTblPtr, CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SB\_MessageStringGet(), CS\_Res\_App\_Table\_Entry\_t::ComparisonValue, CS\_Res\_App\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_PROCESS\_APP\_NO\_ENTRIES\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_AppData\_t::HkPacket, CS\_Def\_App\_Table\_Entry\_t::Name, CS\_Res\_App\_Table\_Entry\_t::Name, CS\_Res\_App\_Table\_Entry\_t::NumBytesToChecksum, CS\_HkPacket\_t::Payload, CS\_Res\_App\_Table\_Entry\_t::StartAddress, CS\_Def\_App\_Table\_Entry\_t::State, CS\_Res\_App\_Table\_Entry\_t::State, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_HandleTableUpdate(), and CS\_TableInit().

Here is the call graph for this function:



**12.9.3.3 CS\_ProcessNewEepromMemoryDefinitionTable()** void CS\_ProcessNewEepromMemoryDefinitionTable (

```

        const CS_Def_EepromMemory_Table_Entry_t * DefinitionTblPtr,
        const CS_Res_EepromMemory_Table_Entry_t * ResultsTblPtr,
        const uint16 NumEntries,
        const uint16 Table )
  
```

Processes a new definition table for EEPROM or Memory tables.

#### Description

Copies data from the definition table to the results table because the results table is where CS keeps all of its checksum data

#### Assumptions, External Events, and Notes:

None

#### Parameters

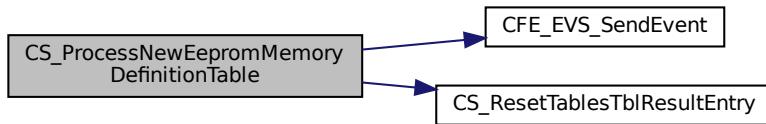
in	<i>DefinitionTblPtr</i>	A pointer to the definition table ( <a href="#">CS_Def_EepromMemory_Table_Entry_t</a> ) that we are operating on
in	<i>ResultsTblPtr</i>	A pointer to the result table ( <a href="#">CS_Res_EepromMemory_Table_Entry_t</a> ) to operate on
in	<i>NumEntries</i>	The number of entries in the table
in	<i>Table</i>	The specific table we are operating on

Definition at line 453 of file cs\_table\_processing.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_INFORMATION, CFE\_ES\_SendEvent(), CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_EEPROM\_TABLE, CS\_MEMORY\_TABLE, CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_TABLETYPE\_NAME\_SIZE, CS\_AppData\_t::EepResTablesTblPtr, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_AppData\_t::MemResTablesTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_HkPacket\_t::Payload, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_HandleTableUpdate(), and CS\_TableInit().

Here is the call graph for this function:



**12.9.3.4 CS\_ProcessNewTablesDefinitionTable()** void CS\_ProcessNewTablesDefinitionTable ( const CS\_Def\_Tables\_Table\_Entry\_t \* DefinitionTblPtr, const CS\_Res\_Tables\_Table\_Entry\_t \* ResultsTblPtr )

Processes a new definition table for the Tables table.

#### Description

Copies data from the definition table to the results table because the results table is where CS keeps all of its checksum data

#### Assumptions, External Events, and Notes:

None

#### Parameters

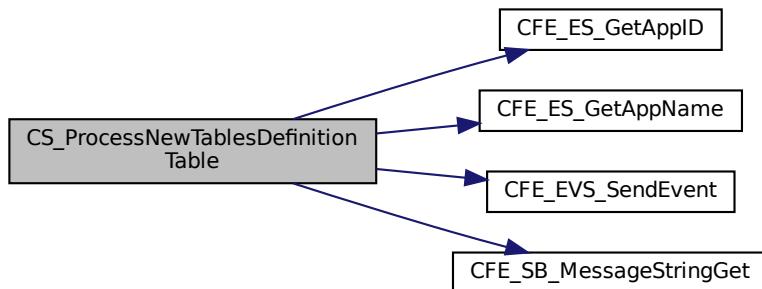
in	<i>DefinitionTblPtr</i>	A pointer to the definiton table ( <a href="#">CS_Def_Tables_Table_Entry_t</a> ) that we are operating on
in	<i>ResultsTblPtr</i>	A pointer to the result table ( <a href="#">CS_Res_Tables_Table_Entry_t</a> ) to operate on

Definition at line 548 of file cs\_table\_processing.c.

References CS\_AppData\_t::AppResTablesTblPtr, CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CFE\_ES\_APPID\_UNDEFINED, CFE\_ES\_GetAppID(), CFE\_ES\_GetAppName(), CFE\_EVS\_EventType\_INFORMATION, CFE\_ES\_SendEvent(), CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, CFE\_SB\_MessageStringGet(), CFE\_TBL\_BAD\_HANDLE, CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_Res\_Tables\_Table\_Entry\_t::ComparisonValue, CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_DEF\_APP\_TABLE\_NAME, CS\_DEF\_EEPROM\_TABLE\_NAME, CS\_DEF\_MEMORY\_TABLE\_NAME, CS\_DEF\_TABLES\_TABLE\_NAME, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID, CS\_RESULTS\_APP\_TABLE\_NAME, CS\_RESULTS\_EEPROM\_TABLE\_NAME, CS\_RESULTS\_MEMORY\_TABLE\_NAME, CS\_RESULTS\_TABLES\_TABLE\_NAME,

CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::EepResTablesTblPtr, CS\_AppData\_t::HkPacket, CS\_Res\_Tables\_Table\_Entry\_t::IsCSCOwner, CS\_AppData\_t::MemResTablesTblPtr, CS\_Def\_Tables\_Table\_Entry\_t::Name, CS\_Res\_Tables\_Table\_Entry\_t::Name, CS\_Res\_Tables\_Table\_Entry\_t::NumBytesToChecksum, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResAppTableHandle, CS\_AppData\_t::ResEepromTableHandle, CS\_AppData\_t::ResMemoryTableHandle, CS\_AppData\_t::ResTablesTableHandle, CS\_Res\_Tables\_Table\_Entry\_t::StartAddress, CS\_Def\_Tables\_Table\_Entry\_t::State, CS\_Res\_Tables\_Table\_Entry\_t::State, CS\_HkPacket\_Payload\_t::TablesCSSState, CS\_Res\_Tables\_Table\_Entry\_t::TblHandle, CS\_AppData\_t::TblResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.  
Referenced by CS\_HandleTableUpdate(), and CS\_TableInit().

Here is the call graph for this function:



**12.9.3.5 CS\_TableInit()** `CFE_Status_t CS_TableInit (`  
 `CFE_TBL_Handle_t * DefinitionTableHandle,`  
 `CFE_TBL_Handle_t * ResultsTableHandle,`  
 `void * DefinitionTblPtr,`  
 `void * ResultsTblPtr,`  
 `const uint16 Table,`  
 `const char * DefinitionTableName,`  
 `const char * ResultsTableName,`  
 `const uint16 NumEntries,`  
 `const char * DefinitionTableFileName,`  
 `const void * DefaultDefTableAddress,`  
 `const uint16 sizeofDefinitionTableEntry,`  
 `const uint16 sizeofResultsTableEntry,`  
 `const CFE_TBL_CallbackFuncPtr_t CallBackFunction )`

Initializes the results and definition table on startup.

#### Description

Completes the Table Services registration and table load for the definition table and the registration for the results table

#### Assumptions, External Events, and Notes:

This function is used on all four type of tables individually. Also, if the default table file is not found for the definition table, this function loads a 'blank' table from memory

### Parameters

in	<i>DefinitionTableHandle</i>	A <a href="#">CFE_TBL_Handle_t</a> pointer that will get filled in with the table handle to the definition table
in	<i>ResultsTableHandle</i>	A <a href="#">CFE_TBL_Handle_t</a> pointer that will get filled in with the table handle to the results table
in	<i>DefinitionTblPtr</i>	A pointer to the definiton table that we are operating on, it will get assigned during this call
in	<i>ResultsTblPtr</i>	A pointer to the result table to operate on , it will get assigned during this call
in	<i>Table</i>	The specific table we are operating on
in	<i>DefinitionTableName</i>	The name of the definition table
in	<i>ResultsTableName</i>	The name of the results table
in	<i>NumEntries</i>	The number of entries in the table
in	<i>DefinitionTableFileName</i>	The name of the file to load the definition table from
in	<i>DefaultDefTableAddress</i>	The address of the default definition table that we may have to load from memory if the file is absent
in	<i>SizeofDefinitionTableEntry</i>	The sizeof an entry in the definition table
in	<i>SizeofResultsTableEntry</i>	The size of an enrty in the results table
in	<i>CallBackFunction</i>	A pointer to a function used to validate the definition table
out	*	DefinitionTableHandle A <a href="#">CFE_TBL_Handle_t</a> pointer that will get filled in with the tbl handle to the definition table
out	*	ResultsTableHandle A <a href="#">CFE_TBL_Handle_t</a> pointer that will get filled in with the tbl handle to the results table
out	*	DefinitionTblPtr A pointer to the definiton table that we are operating on
in	*	ResultsTblPtr A pointer to the result table to operate on , it will get be used to access the table

### Returns

Execution status, see [cFE Return Code Defines](#)

### Return values

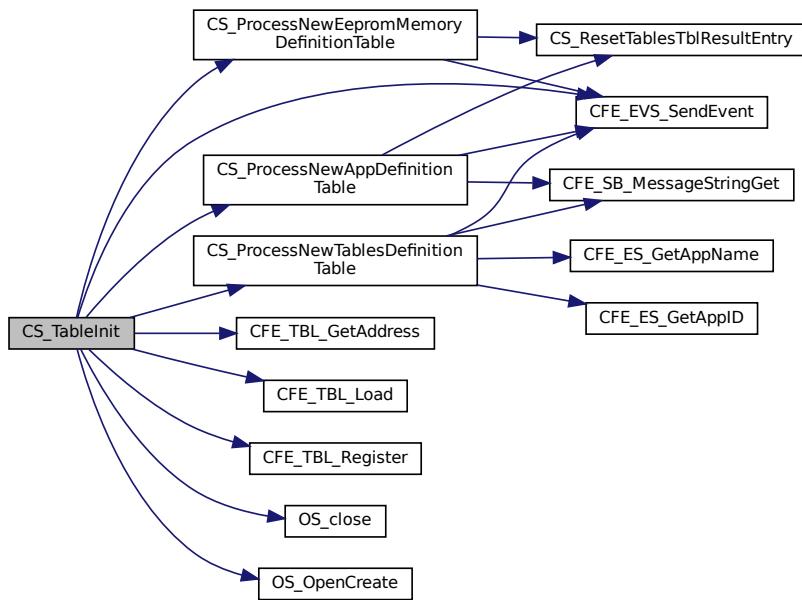
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 813 of file cs\_table\_processing.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_GetAddress(), CFE\_TBL\_INFO\_UPDATED, CFE\_TBL\_Load(), CFE\_TBL\_OPT\_DUMP\_ONLY, CFE\_TBL\_OPT\_LOAD\_DUMP, CFE\_TBL\_OPT\_SNGL\_BUFFER, CFE\_TBL\_Register(), CFE\_TBL\_SRC\_ADDRESS, CFE\_TBL\_SRC\_FILE, CS\_APP\_TABLE, CS\_AppData, CS\_EEPROM\_TABLE, CS\_MEMORY\_TABLE, CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), CS\_STATE\_DISABLED, CS\_TABLES\_TABLE, CS\_TABLETYPE\_NAME\_SIZE, CS\_TBL\_INIT\_ERR\_EID, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, OS\_close(), OS\_ERROR, OS\_FILE\_FLAG\_NONE, OS\_OBJECT\_ID\_UNDEFINED, OS\_OpenCreate(), OS\_READ\_ONLY, OS\_SUCCESS, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_InitAllTables().

Here is the call graph for this function:



**12.9.3.6 CS\_ValidateAppChecksumDefinitionTable()** `CFE_Status_t CS_ValidateAppChecksumDefinitionTable( void * TblPtr )`

Validate App definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>TblPtr</code>	A pointer to the table to be validated
----	---------------------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

#### Return values

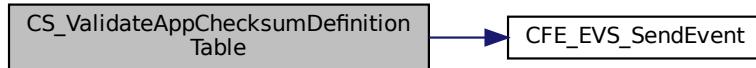
<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CS_TABLE_ERROR</code>	Error code returned on table validation error.

Definition at line 328 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID, CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID, CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID, CS\_VAL\_APP\_INF\_EID, CS\_VAL\_APP\_STAT\_E\_ERR\_EID, CS\_Def\_App\_Table\_Entry\_t::Name, and CS\_Def\_App\_Table\_Entry\_t::State.

Referenced by CS\_InitAllTables().

Here is the call graph for this function:



### 12.9.3.7 CS\_ValidateEepromChecksumDefinitionTable()

`CFE_Status_t CS_ValidateEepromChecksumDefinitionTable(`

`void * TblPtr )`

Validate EEPROM definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>TblPtr</code>	A pointer to the table to be validated
----	---------------------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

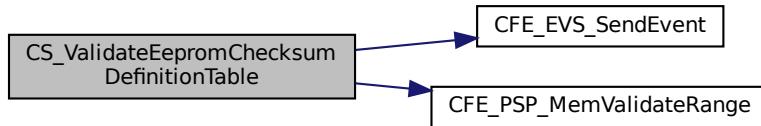
#### Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CS_TABLE_ERROR</code>	Error code returned on table validation error.

Definition at line 48 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_PSP\_MEM\_EEPROM, CFE\_PSP\_MemValidateRange(), CFE\_SUCCESS, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_EEPROM\_INF\_EID, CS\_VAL\_EEPROM\_RANGE\_ERR\_EID, CS\_VAL\_EEPROM\_STATE\_ERR\_EID, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, OS\_ERROR, OS\_SUCCESS, CS\_Def\_EepromMemory\_Table\_

Entry\_t::StartAddress, and CS\_Def\_EepromMemory\_Table\_Entry\_t::State.  
Referenced by CS\_InitAllTables().  
Here is the call graph for this function:



**12.9.3.8 CS\_ValidateMemoryChecksumDefinitionTable()** [CFE\\_Status\\_t](#) CS\_ValidateMemoryChecksumDefinition<→  
Table ( void \* TblPtr )  
Validate Memory definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	TblPtr	A pointer to the table to be validated
----	--------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

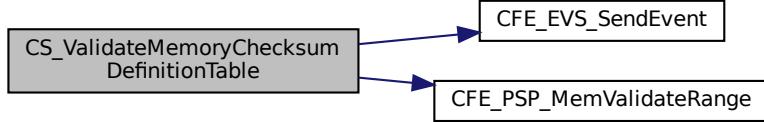
#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
<a href="#">CS_TABLE_ERROR</a>	Error code returned on table validation error.

Definition at line 128 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_PSP\_MEM\_ANY, CFE\_PSP\_MemValidateRange(), CFE\_SUCCESS, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_MEMORY\_INF\_EID, CS\_VAL\_MEMORY\_RANGE\_ERR\_EID, CS\_VAL\_MEMORY\_STATE\_ERR\_EID, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, OS\_ERROR, OS\_SUCCESS, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Def\_EepromMemory\_Table\_Entry\_t::State.  
Referenced by CS\_InitAllTables().

Here is the call graph for this function:



**12.9.3.9 CS\_ValidateTablesChecksumDefinitionTable()** [CFE\\_Status\\_t](#) CS\_ValidateTablesChecksumDefinition<Table (void \* TblPtr ) Validate Tables definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	TblPtr	A pointer to the table to be validated
----	--------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

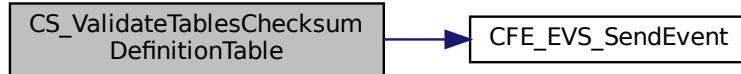
#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
<a href="#">CS_TABLE_ERROR</a>	Error code returned on table validation error.

Definition at line 212 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_TABLES\_DEF\_TBL\_DUPL\_ER\_EID, CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID, CS\_VAL\_TABLES\_INF\_EID, CS\_VAL\_TABLES\_STATE\_ERR\_EID, CS\_DefTables\_Table\_Entry\_t::Name, and CS\_DefTables\_Table\_Entry\_t::State. Referenced by CS\_InitAllTables().

Here is the call graph for this function:



## 12.10 apps/cs/fsw/src/cs\_app.c File Reference

```
#include <string.h>
#include "cfe.h"
#include "cs_app.h"
#include "cs_platform_cfg.h"
#include "cs_events.h"
#include "cs_utils.h"
#include "cs_compute.h"
#include "cs_eeprom_cmds.h"
#include "cs_table_cmds.h"
#include "cs_memory_cmds.h"
#include "cs_app_cmds.h"
#include "cs_cmds.h"
#include "cs_init.h"
```

### Macros

- #define **CS\_NUM\_DATA\_STORE\_STATES** 6 /\* 4 tables + OS CS + cFE core number of checksum states for CDS \*/

### Functions

- void **CS\_AppMain** (void)  
*CFS Checksum (CS) application entry point.*
- **CFE\_Status\_t CS\_AppInit** (void)  
*Initialize the Checksum CFS application.*
- **CFE\_Status\_t CS\_AppPipe** (const **CFE\_SB\_Buffer\_t** \*BufPtr)  
*Process a command pipe message.*
- void **CS\_ProcessCmd** (const **CFE\_SB\_Buffer\_t** \*BufPtr)  
*Command packet processor.*
- void **CS\_HousekeepingCmd** (const **CS\_NoArgsCmd\_t** \*CmdPtr)  
*Process housekeeping request.*
- **CFE\_Status\_t CS\_CreateRestoreStatesFromCDS** (void)  
*Restore tables states from CDS if enabled.*
- void **CS\_UpdateCDS** (void)  
*CFS Checksum (CS) Critical Data Store Update.*

## Variables

- [CS\\_AppData\\_t CS\\_AppData](#)

*Extern the CS\_AppData so all CS files can use it.*

### 12.10.1 Detailed Description

CFS Checksum (CS) Application provides the service of background checksumming user-defined objects in the CFS

### 12.10.2 Macro Definition Documentation

**12.10.2.1 CS\_NUM\_DATA\_STORE\_STATES** `#define CS_NUM_DATA_STORE_STATES 6 /* 4 tables + OS CS + cFE core number of checksum states for CDS */`  
Definition at line 45 of file cs\_app.c.

### 12.10.3 Function Documentation

**12.10.3.1 CS\_AppInit()** `CFE_Status_t CS_AppInit ( void )`

Initialize the Checksum CFS application.

#### Description

Checksum application initialization routine. This function performs all the required startup steps to get the application registered with the cFE services so it can begin to receive command messages and begin background checksumming.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

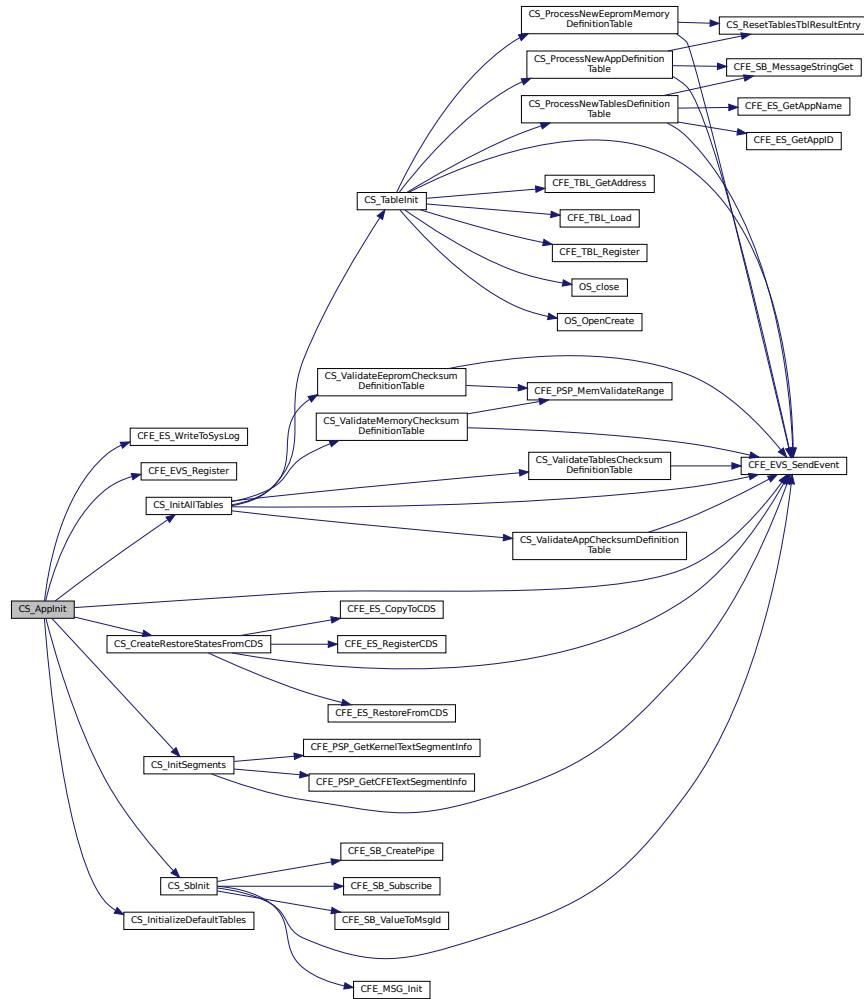
Definition at line 148 of file cs\_app.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_ES\_RunStatus\_APP\_RUN, CFE\_ES\_WriteToSysLog(), CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_Register(), CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_HkPacket\_Payload\_t::CfeCoreCSState, CS\_HkPacket\_Payload\_t::ChecksumState, CS\_AppData, CS\_APPS\_TBL\_POWERON\_STATE, CS\_CFECORE\_CHECKSUM\_STATE, CS\_CreateRestoreStatesFromCDS(), CS\_DEFAULT\_BYTES\_PER\_CYCLE, CS\_EEPROM\_TBL\_POWERON\_STATE, CS\_INIT\_INF\_EID, CS\_InitAllTables(), CS\_InitializeDefaultTables(), CS\_InitSegments(), CS\_MAJOR\_VERSION, CS\_MEMORY\_TBL\_POWERON\_STATE, CS\_MINOR\_VERSION, CS\_MISSION\_REV, CS\_OSCS\_CHECKSUM\_STATE, CS\_REVISION, CS\_SblInit(), CS\_STATUS\_ENABLED, CS\_TABLES\_TBL\_POWERON\_STATE, CS\_HkPacket\_Payload\_t::CurrentCSTable, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_AppData\_t::MaxBytesPerCycle, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OneShotInProgress,

CS\_HkPacket\_Payload\_t::OSCSSState, CS\_HkPacket\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::RunStatus, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_AppMain().

Here is the call graph for this function:



**12.10.3.2 CS\_AppMain()** void CS\_AppMain ( void )

CFS Checksum (CS) application entry point.

## Description

Checksum application entry point and main process loop.

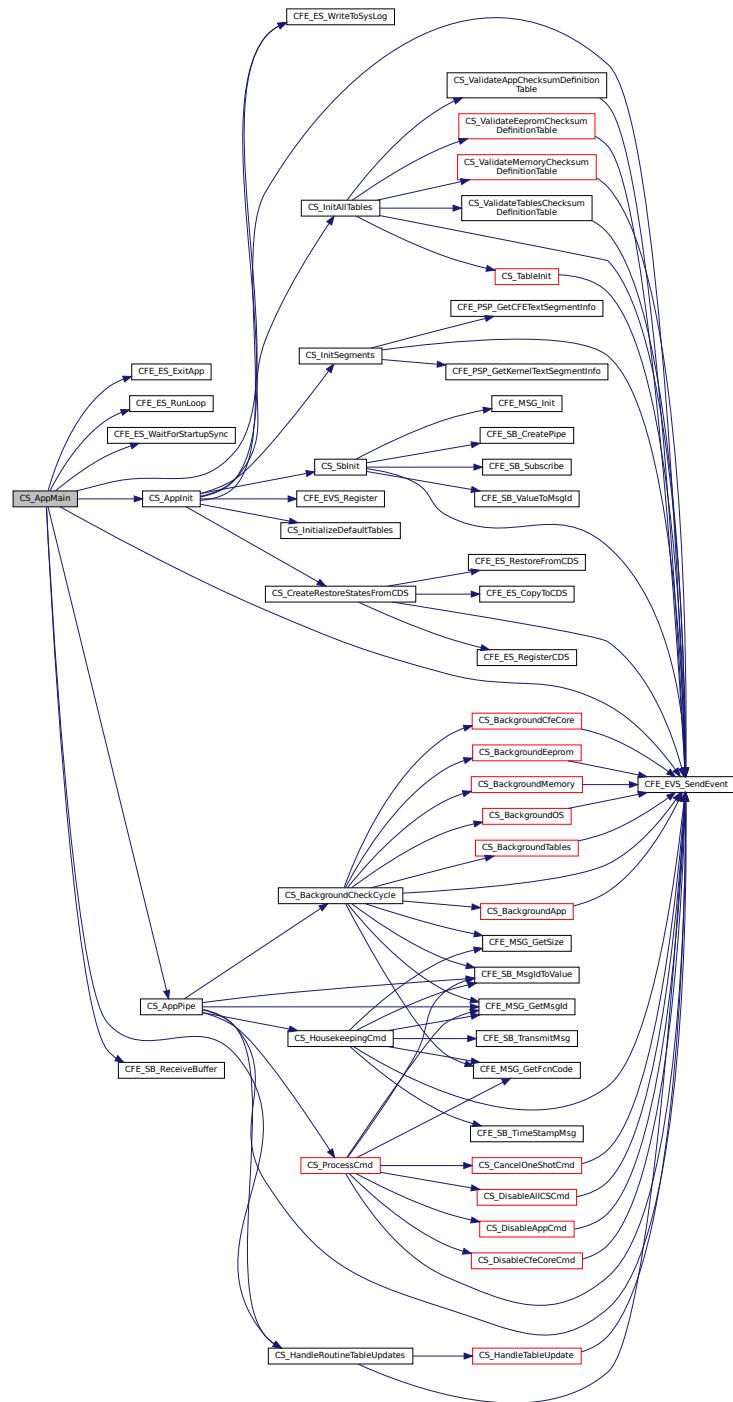
**Assumptions, External Events, and Notes:**

None

Definition at line 59 of file cs\_app.c.

References CFE\_ES\_ExitApp(), CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_ES\_RunLoop(), CFE\_ES\_RunStatus\_APP\_ERROR, CFE\_ES\_RunStatus\_SYS\_EXCEPTION, CFE\_ES\_WaitForStartupSync(), CFE\_ES\_WriteToSysLog(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SB\_NO\_MESSAGE, CFE\_SB\_ReceiveBuffer(), CFE\_SB\_TIME\_OUT, CFE\_SUCCESS, CS\_AppData\_t::CmdPipe, CS\_AppData, CS\_AppInit(), CS\_APPMAIN\_PERF\_ID, CS\_AppPipe(), CS\_EXIT\_ERR\_EID, CS\_EXIT\_INF\_EID, CS\_HandleRoutineTableUpdates(), CS\_STARTUP\_TIMEOUT, CS\_WAKEUP\_TIMEOUT, and CS\_AppData\_t::RunStatus.

Here is the call graph for this function:



```
12.10.3.3 CS_AppPipe() CFE_Status_t CS_AppPipe (
```

Process a command pipe message.

#### Description

Processes a single software bus command pipe message. Checks the message and command IDs and calls the appropriate routine to handle the command.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>BufPtr</i>	A <code>CFE_SB_Buffer_t*</code> pointer that references the software bus message. The calling function verifies that <i>BufPtr</i> is non-null.
----	---------------	---

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

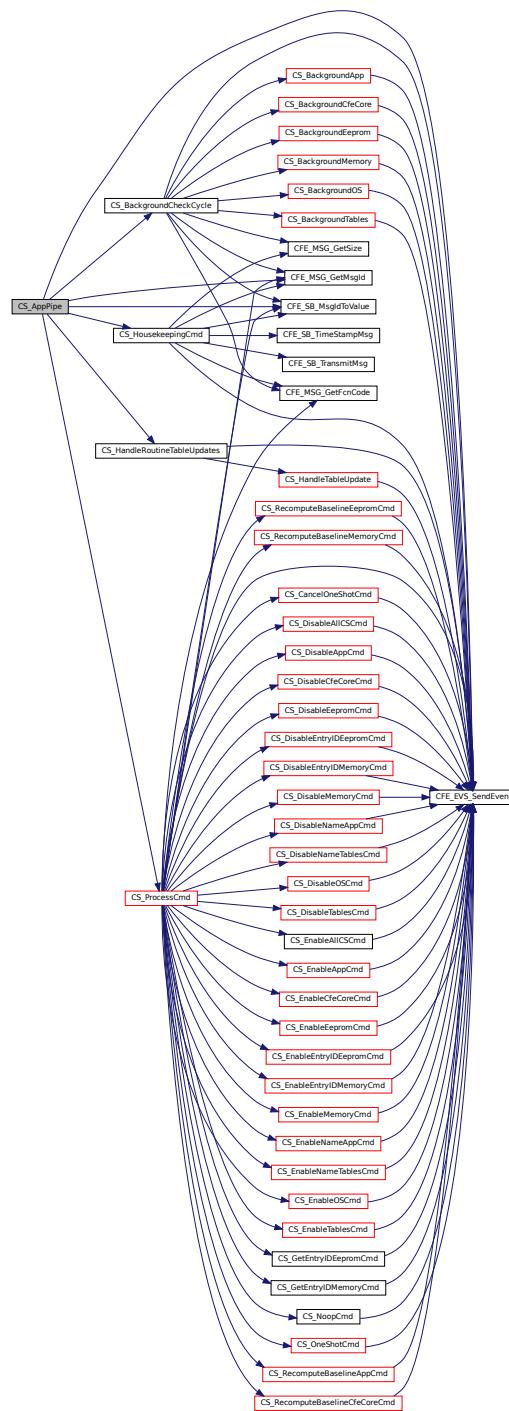
<code>CFE_SUCCESS</code>	Successful execution.
--------------------------	-----------------------

Definition at line 222 of file cs\_app.c.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_MSG_GetMsgId()`, `CFE_SB_INVALID_` ↔ `MSG_ID`, `CFE_SB_MsgIdToValue()`, `CFE_SUCCESS`, `CS_HkPacket_Payload_t::CmdErrCounter`, `CS_AppData`, `CS_` ↔ `BACKGROUND_CYCLE_MID`, `CS_BackgroundCheckCycle()`, `CS_CMD_MID`, `CS_HandleRoutineTableUpdates()`, `C_` ↔ `S_HousekeepingCmd()`, `CS_MID_ERR_EID`, `CS_ProcessCmd()`, `CS_SEND_HK_MID`, `CS_AppData_t::HkPacket`, `C_` ↔ `FE_SB_Msg::Msg`, and `CS_HkPacket_t::Payload`.

Referenced by `CS_AppMain()`.

Here is the call graph for this function:



#### 12.10.3.4 `CS_CreateRestoreStatesFromCDS()`

```
CFE_Status_t CS_CreateRestoreStatesFromCDS (
    void )
```

Restore tables states from CDS if enabled.

#### Description

Restore CS state of tables from CDS

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

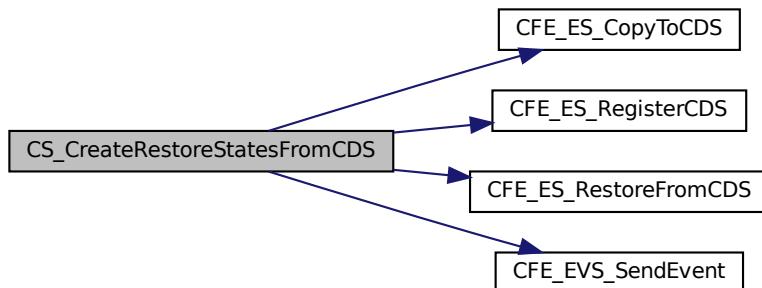
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 615 of file cs\_app.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_ES\_CDS\_ALREADY\_EXISTS, CFE\_ES\_CDS\_BAD\_HAN←DLE, CFE\_ES\_CopyToCDS(), CFE\_ES\_RegisterCDS(), CFE\_ES\_RestoreFromCDS(), CFE\_EVS\_EventType\_ERR←OR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_HkPacket\_Payload\_t::CfeCoreCSState, CS\_AppData, CS\_AP←PS\_TBL\_POWERON\_STATE, CS\_CDS\_NAME, CS\_CFECORE\_CHECKSUM\_STATE, CS\_CR\_CDS\_CPY\_ERR←EID, CS\_CR\_CDS\_REG\_ERR\_EID, CS\_CR\_CDS\_RES\_ERR\_EID, CS\_EEPROM\_TBL\_POWERON\_STATE, CS\_MEMORY\_TBL\_POWERON\_STATE, CS\_NUM\_DATA\_STORE\_STATES, CS\_OSCS\_CHECKSUM\_STATE, CS\_TABLES\_TBL\_POWERON\_STATE, CS\_AppData\_t::DataStoreHandle, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OSCSState, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_AppInit().

Here is the call graph for this function:



#### 12.10.3.5 CS\_HousekeepingCmd()

```
void CS_HousekeepingCmd (
    const CS_NoArgsCmd_t * CmdPtr )
```

Process housekeeping request.

## Description

Processes an on-board housekeeping request message.

### Assumptions, External Events, and Notes:

This command does not affect the command execution counter

## Parameters

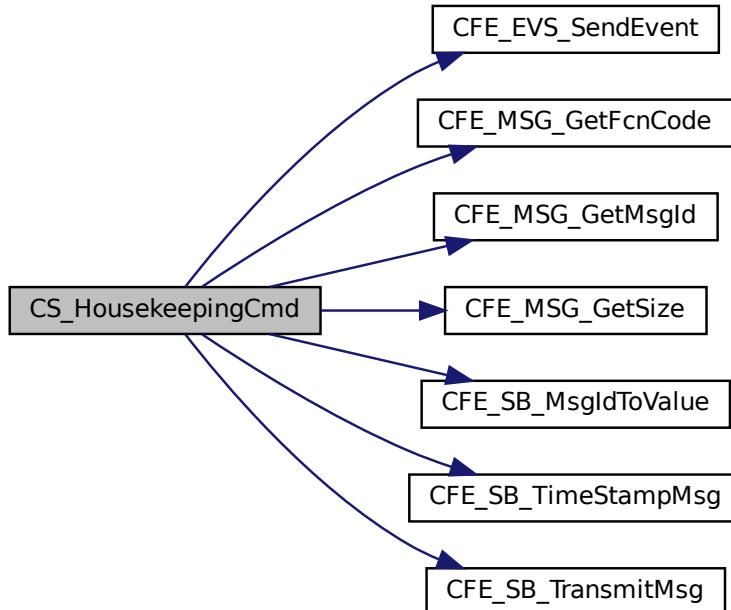
in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

Definition at line 578 of file cs\_app.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_MSG\_GetSize(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_MsgIdToValue(), CFE\_SB\_TimeStampMsg(), CFE\_SB\_TransmitMsg(), CS\_NoArgsCmd\_t::CmdHeader, CS\_AppData, CS\_LEN\_ERR\_EID, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::TlmHeader.

Referenced by CS\_AppPipe().

Here is the call graph for this function:



**12.10.3.6 CS\_ProcessCmd()** void CS\_ProcessCmd ( const CFE\_SB\_Buffer\_t \* BufPtr )

Command packet processor.

**Description**

Processes all CS commands

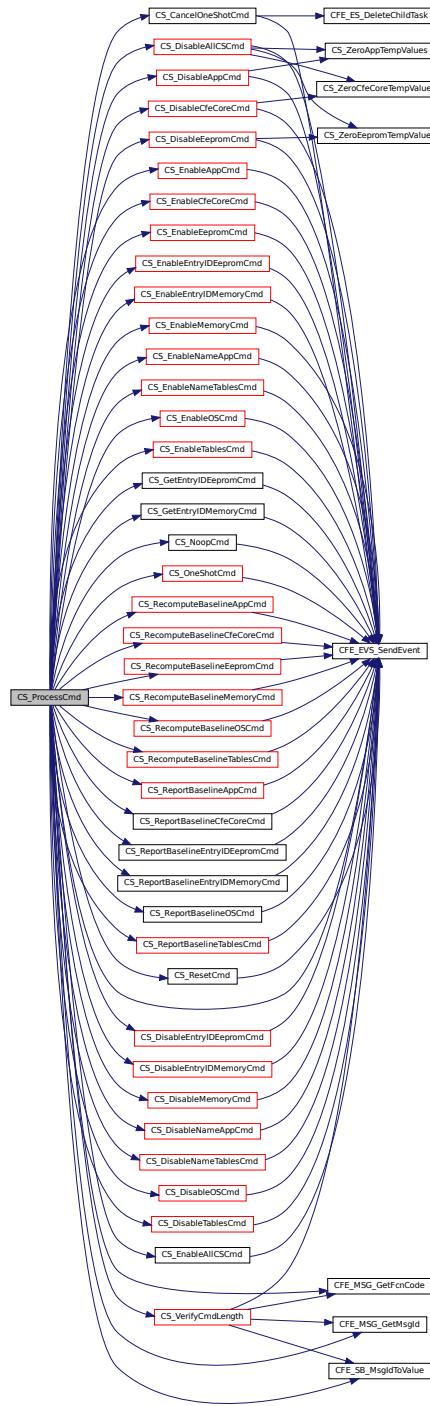
**Parameters**

in	<i>BufPtr</i>	A CFE_SB_Buffer_t* pointer that references the software bus pointer. The BufPtr is verified non-null in CS_AppMain.
----	---------------	---

Definition at line 265 of file cs\_app.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_MsgIdToValue(), CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CANCEL\_ONE\_SHOT\_CC, CS\_CancelOneShotCmd(), CS\_CC1\_ERR\_EID, CS\_DISABLE\_ALL\_CS\_CC, CS\_DISABLE\_APPS\_CC, CS\_DISABLE\_CFE\_CORE\_CC, CS\_DISABLE\_EEPROM\_CC, CS\_DISABLE\_ENTRY\_EEPROM\_CC, CS\_DISABLE\_ENTRY\_MEMORY\_CC, CS\_DISABLE\_MEMORY\_CC, CS\_DISABLE\_NAME\_APP\_CC, CS\_DISABLE\_NAME\_TABLE\_CC, CS\_DISABLE\_OS\_CC, CS\_DISABLE\_TABLES\_CC, CS\_DisableAllCSCmd(), CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEepromCmd(), CS\_DisableEntryIDEepromCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_ENABLE\_ALL\_CS\_CC, CS\_ENABLE\_APPS\_CC, CS\_ENABLE\_CFE\_CORE\_CC, CS\_ENABLE\_EEPROM\_CC, CS\_ENABLE\_ENTRY\_EEPROM\_CC, CS\_ENABLE\_ENTRY\_MEMORY\_CC, CS\_ENABLE\_MEMORY\_CC, CS\_ENABLE\_NAME\_APP\_CC, CS\_ENABLE\_NAME\_TABLE\_CC, CS\_ENABLE\_OS\_CC, CS\_ENABLE\_TABLES\_CC, CS\_EnableAllCSCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEepromCmd(), CS\_EnableEntryIDEepromCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_EnableOSCmd(), CS\_EnableTablesCmd(), CS\_GET\_ENTRY\_ID\_EEPROM\_CC, CS\_GET\_ENTRY\_ID\_MEMORY\_CC, CS\_GetEntryIDEepromCmd(), CS\_GetEntryIDMemoryCmd(), CS\_NOOP\_CC, CS\_NoopCmd(), CS\_ONESHOT\_CC, CS\_OneShotCmd(), CS\_RECOMPUTE\_BASELINE\_APP\_CC, CS\_RECOMPUTE\_BASELINE\_CFE\_CORE\_CC, CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC, CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC, CS\_RecomputeBaselineAppCmd(), CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), CS\_RecomputeBaselineTablesCmd(), CS\_REPORT\_BASELINE\_APP\_CC, CS\_REPORT\_BASELINE\_CFE\_CORE\_CC, CS\_REPORT\_BASELINE\_EEPROM\_CC, CS\_REPORT\_BASELINE\_MEMORY\_CC, CS\_ReportBaselineAppCmd(), CS\_ReportBaselineCfeCoreCmd(), CS\_ReportBaselineEntryIDEepromCmd(), CS\_ReportBaselineEntryIDMemoryCmd(), CS\_ReportBaselineOSCmd(), CS\_ReportBaselineTablesCmd(), CS\_RESET\_CC, CS\_ResetCmd(), CS\_VerifyCmdLength(), CS\_AppData\_t::HkPacket, CFE\_SB\_Msg::Msg, and CS\_HkPacket\_t::Payload.  
Referenced by CS\_AppPipe().

Here is the call graph for this function:



**12.10.3.7 CS\_UpdateCDS()** void CS\_UpdateCDS ( void )

CFS Checksum (CS) Critical Data Store Update.

#### Description

Checksum application entry point and main process loop.

#### Assumptions, External Events, and Notes:

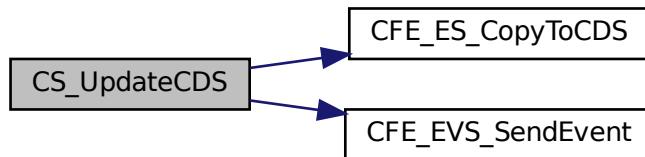
None

Definition at line 709 of file cs\_app.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_ES\_CDS\_BAD\_HANDLE, CFE\_ES\_CopyToCDS(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_RESOURCEID\_TEST\_DEFINED, CFE\_SUCCESS, CS\_HkPacket\_Payload\_t::CfeCoreCSState, CS\_AppData, CS\_NUM\_DATA\_STORE\_STATES, CS\_UPDATE\_CDS\_ER↔R\_EID, CS\_AppData\_t::DataStoreHandle, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OSCSState, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEepromCmd(), CS\_DisableMemoryCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEepromCmd(), CS\_EnableMemoryCmd(), CS\_EnableOSCmd(), and CS\_EnableTablesCmd().

Here is the call graph for this function:



#### 12.10.4 Variable Documentation

##### 12.10.4.1 CS\_AppData CS\_AppData\_t CS\_AppData

Extern the CS\_AppData so all CS files can use it.

Definition at line 52 of file cs\_app.c.

Referenced by CS\_AppInit(), CS\_AppMain(), CS\_AppPipe(), CS\_BackgroundApp(), CS\_BackgroundCfeCore(), CS\_BackgroundCheckCycle(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), CS\_BackgroundTables(), CS\_CancelOneShotCmd(), CS\_CheckRecomputeOneshot(), CS\_ComputeApp(), CS\_ComputeEepromMemory(), CS\_ComputeTables(), CS\_CreateRestoreStatesFromCDS(), CS\_DisableAllCSCmd(), CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEepromCmd(), CS\_DisableEntryIDEepromCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_EnableAllCSCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEepromCmd(), CS\_EnableEntryIDEepromCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_EnableOSCmd(), CS\_EnableTablesCmd(), CS\_FindEnabledAppEntry(), CS\_FindEnabledEepromEntry(), CS\_FindEnabledMemoryEntry(), CS\_FindEnabledTablesEntry(), CS\_GetAppDefTblEntryByName(), CS\_GetAppResTblEntryByName(), CS\_GetEntryIDEepromCmd(), CS\_GetEntryIDMemoryCmd(), CS\_GetTableDefTblEntryByName(), CS\_GetTableResTblEntryByName(), CS\_GoToNextTable(), CS\_HandleRoutineTableUpdates(), CS\_HandleTableUpdate(), CS↔

\_HousekeepingCmd(), CS\_InitAllTables(), CS\_InitializeDefaultTables(), CS\_InitSegments(), CS\_NoopCmd(), CS\_OneShotChildTask(), CS\_OneShotCmd(), CS\_ProcessCmd(), CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), CS\_RecomputeAppChildTask(), CS\_RecomputeBaselineAppCmd(), CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), CS\_RecomputeBaselineTablesCmd(), CS\_RecomputeEepromMemoryChildTask(), CS\_RecomputeTablesChildTask(), CS\_ReportBaselineAppCmd(), CS\_ReportBaselineCfeCoreCmd(), CS\_ReportBaselineEntryIDEepromCmd(), CS\_ReportBaselineEntryIDMemoryCmd(), CS\_ReportBaselineOSCcmd(), CS\_ReportBaselineTablesCmd(), CS\_ResetCmd(), CS\_SbInit(), CS\_TableInit(), CS\_UpdateCDS(), CS\_VerifyCmdLength(), CS\_ZeroAppTempValues(), CS\_ZeroCfeCoreTempValues(), CS\_ZeroEepromTempValues(), CS\_ZeroMemoryTempValues(), CS\_ZeroOSTempValues(), and CS\_ZeroTablesTempValues().

## 12.11 apps/cs/fsw/src/cs\_app.h File Reference

```
#include "cfe.h"
#include "cs_tbldefs.h"
#include "cs_msg.h"
#include "cs_msgdefs.h"
#include "cs_platform_cfg.h"
#include "cs_mission_cfg.h"
#include "cs_msgids.h"
#include "cs_perfids.h"
#include "cs_verify.h"
#include "cs_version.h"
```

### Data Structures

- struct [CS\\_AppData\\_t](#)

*CS global data structure.*

### Macros

- #define [CS\\_WAKEUP\\_TIMEOUT](#) 1000

*Wakeup for CS.*

### CS Error Codes

- #define [CS\\_ERROR](#) (-1)  
*Error code returned when a checksum compare failed.*
- #define [CS\\_ERR\\_NOT\\_FOUND](#) (-2)  
*Error code returned the app or table requested could not be found.*
- #define [CS\\_TABLE\\_ERROR](#) (-3)  
*Error code returned on table validation error.*

### CS Command Pipe Parameters

- #define [CS\\_CMD\\_PIPE\\_NAME](#) "CS\_CMD\_PIPE"
- #define [CS\\_CMD\\_PIPE\\_NAME\\_LEN](#) 16

### CS Name of Table Size

- #define [CS\\_TABLETYPE\\_NAME\\_SIZE](#) 10

### CS Child Task Names

- #define [CS\\_RECOMP\\_OS\\_TASK\\_NAME](#) "CS\_RecompOSTsk"

- #define CS\_RECOMP\_CFECORE\_TASK\_NAME "CS\_RecmpCfeCoreTsk"
- #define CS\_RECOMP\_MEMORY\_TASK\_NAME "CS\_RecmpMemoryTsk"
- #define CS\_RECOMP\_EEPROM\_TASK\_NAME "CS\_RecmpEepromTsk"
- #define CS\_RECOMP\_APP\_TASK\_NAME "CS\_RecmpAppTsk"
- #define CS\_RECOMP\_TABLES\_TASK\_NAME "CS\_RecmpTableTsk"
- #define CS\_ONESHOT\_TASK\_NAME "CS\_OneShotTask"

## Functions

- void **CS\_AppMain** (void)  
*CFS Checksum (CS) application entry point.*
- void **CS\_UpdateCDS** (void)  
*CFS Checksum (CS) Critical Data Store Update.*
- **CFE\_Status\_t CS\_AppInit** (void)  
*Initialize the Checksum CFS application.*
- **CFE\_Status\_t CS\_AppPipe** (const **CFE\_SB\_Buffer\_t** \*BufPtr)  
*Process a command pipe message.*
- void **CS\_HousekeepingCmd** (const **CS\_NoArgsCmd\_t** \*CmdPtr)  
*Process housekeeping request.*
- void **CS\_ProcessCmd** (const **CFE\_SB\_Buffer\_t** \*BufPtr)  
*Command packet processor.*
- **CFE\_Status\_t CS\_CreateRestoreStatesFromCDS** (void)  
*Restore tables states from CDS if enabled.*

## Variables

- **CS\_AppData\_t CS\_AppData**  
*Extern the CS\_AppData so all CS files can use it.*

### 12.11.1 Detailed Description

Unit specification for the Core Flight System (CFS) Checksum (CS) Application.

### 12.11.2 Macro Definition Documentation

**12.11.2.1 CS\_CMD\_PIPE\_NAME** #define CS\_CMD\_PIPE\_NAME "CS\_CMD\_PIPE"  
Definition at line 64 of file cs\_app.h.

**12.11.2.2 CS\_CMD\_PIPE\_NAME\_LEN** #define CS\_CMD\_PIPE\_NAME\_LEN 16  
Definition at line 65 of file cs\_app.h.

**12.11.2.3 CS\_ERR\_NOT\_FOUND** #define CS\_ERR\_NOT\_FOUND (-2)  
Error code returned the app or table requested could not be found.  
Definition at line 56 of file cs\_app.h.

**12.11.2.4 CS\_ERROR** #define CS\_ERROR (-1)  
Error code returned when a checksum compare failed.  
Definition at line 55 of file cs\_app.h.

**12.11.2.5 CS\_ONESHOT\_TASK\_NAME** #define CS\_ONESHOT\_TASK\_NAME "CS\_OneShotTask"  
Definition at line 85 of file cs\_app.h.

**12.11.2.6 CS\_RECOMP\_APP\_TASK\_NAME** #define CS\_RECOMP\_APP\_TASK\_NAME "CS\_RecmpAppTsk"  
Definition at line 83 of file cs\_app.h.

**12.11.2.7 CS\_RECOMP\_CFECORE\_TASK\_NAME** #define CS\_RECOMP\_CFECORE\_TASK\_NAME "CS\_RecmpCfe↔CoreTsk"  
Definition at line 80 of file cs\_app.h.

**12.11.2.8 CS\_RECOMP\_EEPROM\_TASK\_NAME** #define CS\_RECOMP\_EEPROM\_TASK\_NAME "CS\_RecmpEeprom↔Tsk"  
Definition at line 82 of file cs\_app.h.

**12.11.2.9 CS\_RECOMP\_MEMORY\_TASK\_NAME** #define CS\_RECOMP\_MEMORY\_TASK\_NAME "CS\_RecmpMemory↔Tsk"  
Definition at line 81 of file cs\_app.h.

**12.11.2.10 CS\_RECOMP\_OS\_TASK\_NAME** #define CS\_RECOMP\_OS\_TASK\_NAME "CS\_RecmpOSTsk"  
Definition at line 79 of file cs\_app.h.

**12.11.2.11 CS\_RECOMP\_TABLES\_TASK\_NAME** #define CS\_RECOMP\_TABLES\_TASK\_NAME "CS\_RecmpTableTsk"  
Definition at line 84 of file cs\_app.h.

**12.11.2.12 CS\_TABLE\_ERROR** #define CS\_TABLE\_ERROR (-3)  
Error code returned on table validation error.  
Definition at line 57 of file cs\_app.h.

**12.11.2.13 CS\_TABLETYPE\_NAME\_SIZE** #define CS\_TABLETYPE\_NAME\_SIZE 10  
Definition at line 72 of file cs\_app.h.

**12.11.2.14 CS\_WAKEUP\_TIMEOUT** #define CS\_WAKEUP\_TIMEOUT 1000  
Wakeup for CS.

#### Description

Wakes up CS every 1 second for routine maintenance whether a message was received or not.

Definition at line 95 of file cs\_app.h.

### 12.11.3 Function Documentation

**12.11.3.1 CS\_AppInit()** [CFE\\_Status\\_t](#) CS\_AppInit (

```
    void )
```

Initialize the Checksum CFS application.

#### Description

Checksum application initialization routine. This function performs all the required startup steps to get the application registered with the cFE services so it can begin to receive command messages and begin background checksumming.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

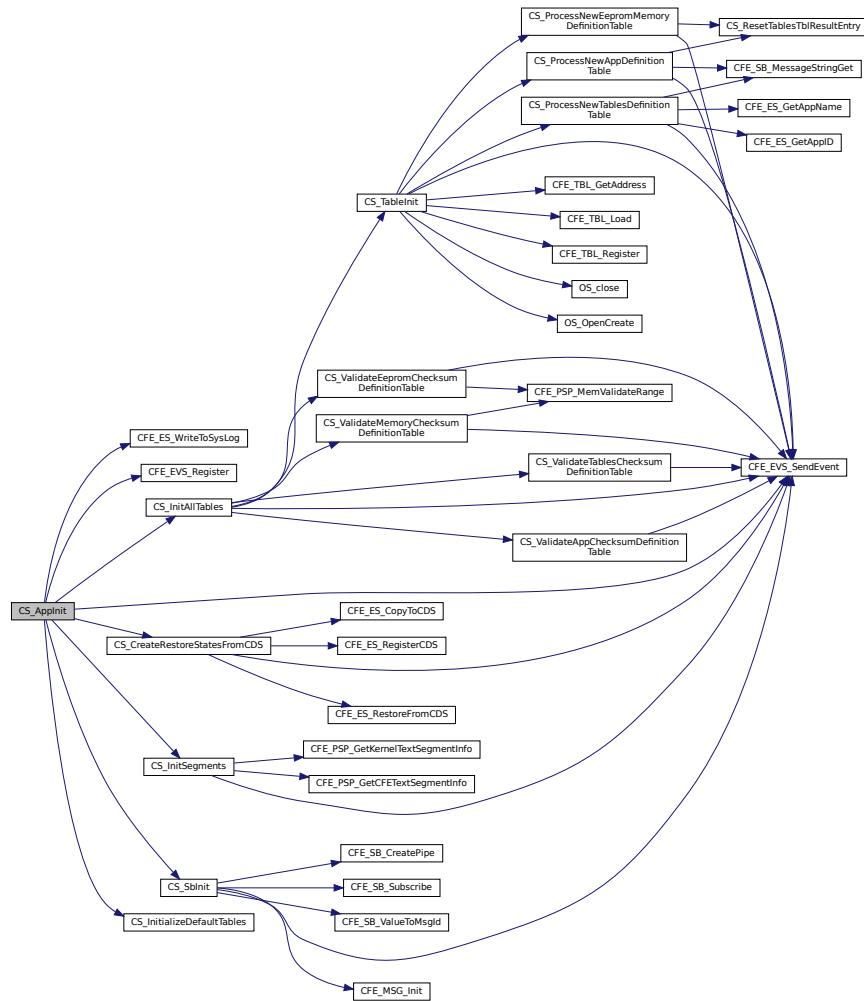
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 148 of file cs\_app.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_ES\_RunStatus\_APP\_RUN, CFE\_ES\_WriteToSysLog(), CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_Register(), CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_HkPacket\_Payload\_t::CfeCoreCSState, CS\_HkPacket\_Payload\_t::ChecksumState, CS\_AppData, CS\_APPS\_TBL\_POWERON\_STATE, CS\_CFECORE\_CHECKSUM\_STATE, CS\_CreateRestoreStatesFromCDS(), CS\_DEFAULT\_BYTES\_PER\_CYCLE, CS\_EEPROM\_TBL\_POWERON\_STATE, CS\_INIT\_INF\_EID, CS\_InitAllTables(), CS\_InitializeDefaultTables(), CS\_InitSegments(), CS\_MAJOR\_VERSION, CS\_MEMORY\_TBL\_POWERON\_STATE, CS\_MINOR\_VERSION, CS\_MISSION\_REV, CS\_OSCS\_CHECKSUM\_STATE, CS\_REVISION, CS\_SbInit(), CS\_STATUS\_ENABLED, CS\_TABLES\_TBL\_POWERON\_STATE, CS\_HkPacket\_Payload\_t::CurrentCSTable, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_AppData\_t::MaxBytesPerCycle, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_Payload\_t::OSCSState, CS\_HkPacket\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::RunStatus, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_AppMain().

Here is the call graph for this function:



### 12.11.3.2 CS\_AppMain()

```
void CS_AppMain (
    void )
```

CFS Checksum (CS) application entry point.

#### Description

Checksum application entry point and main process loop.

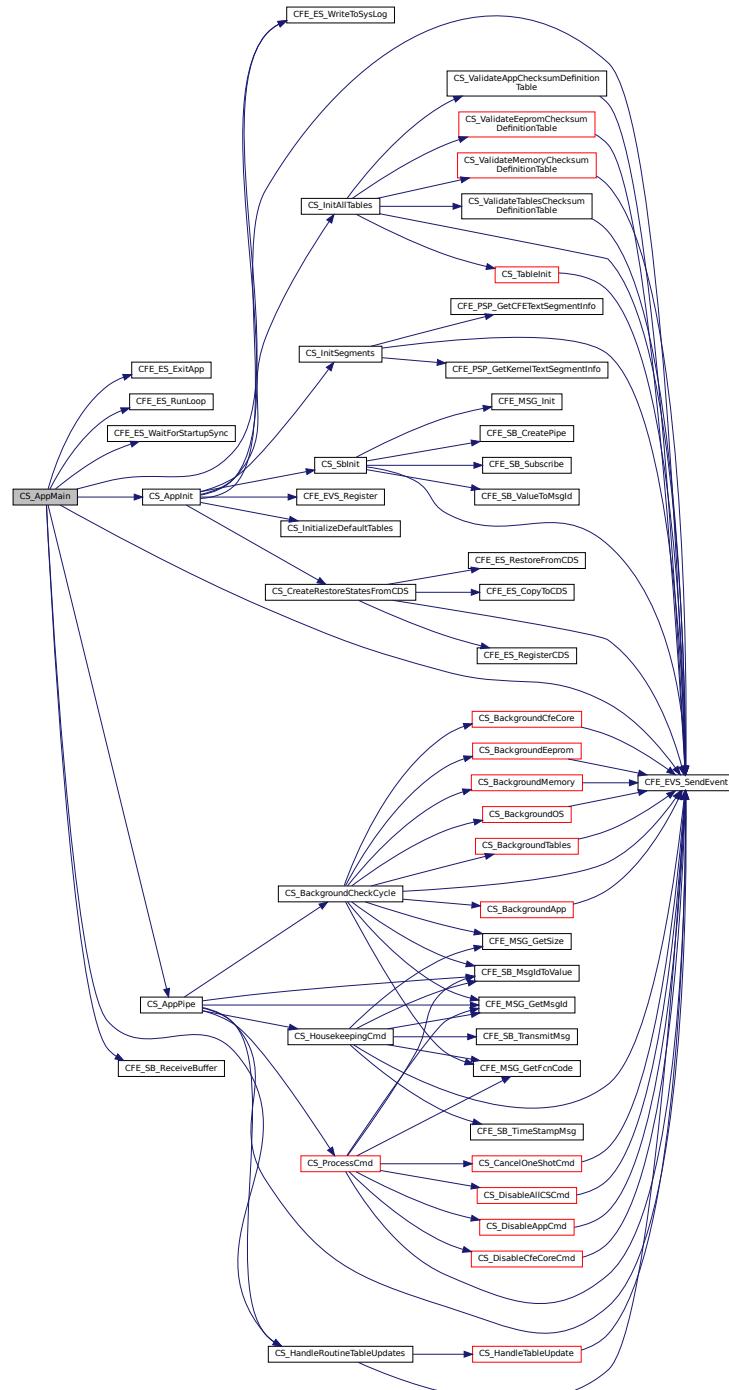
#### Assumptions, External Events, and Notes:

None

Definition at line 59 of file cs\_app.c.

References CFE\_ES\_ExitApp(), CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_ES\_RunLoop(), CFE\_ES\_RunStatus\_APP\_ERROR, CFE\_ES\_RunStatus\_SYS\_EXCEPTION, CFE\_ES\_WaitForStartupSync(), CFE\_ES\_WriteToSysLog(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_S-

B\_NO\_MESSAGE, CFE\_SB\_ReceiveBuffer(), CFE\_SB\_TIME\_OUT, CFE\_SUCCESS, CS\_AppData\_t::CmdPipe, CS\_AppData, CS\_AppInit(), CS\_APPMAIN\_PERF\_ID, CS\_AppPipe(), CS\_EXIT\_ERR\_EID, CS\_EXIT\_INF\_EID, CS\_HANDLEROUTINETABLEUPDATES(), CS\_STARTUP\_TIMEOUT, CS\_WAKEUP\_TIMEOUT, and CS\_AppData\_t::RunStatus. Here is the call graph for this function:



**12.11.3.3 CS\_AppPipe()** `CFE_Status_t CS_AppPipe (`  
    `const CFE_SB_Buffer_t * BufPtr )`

Process a command pipe message.

#### Description

Processes a single software bus command pipe message. Checks the message and command IDs and calls the appropriate routine to handle the command.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>BufPtr</i>	A <code>CFE_SB_Buffer_t*</code> pointer that references the software bus message. The calling function verifies that <i>BufPtr</i> is non-null.
----	---------------	---

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

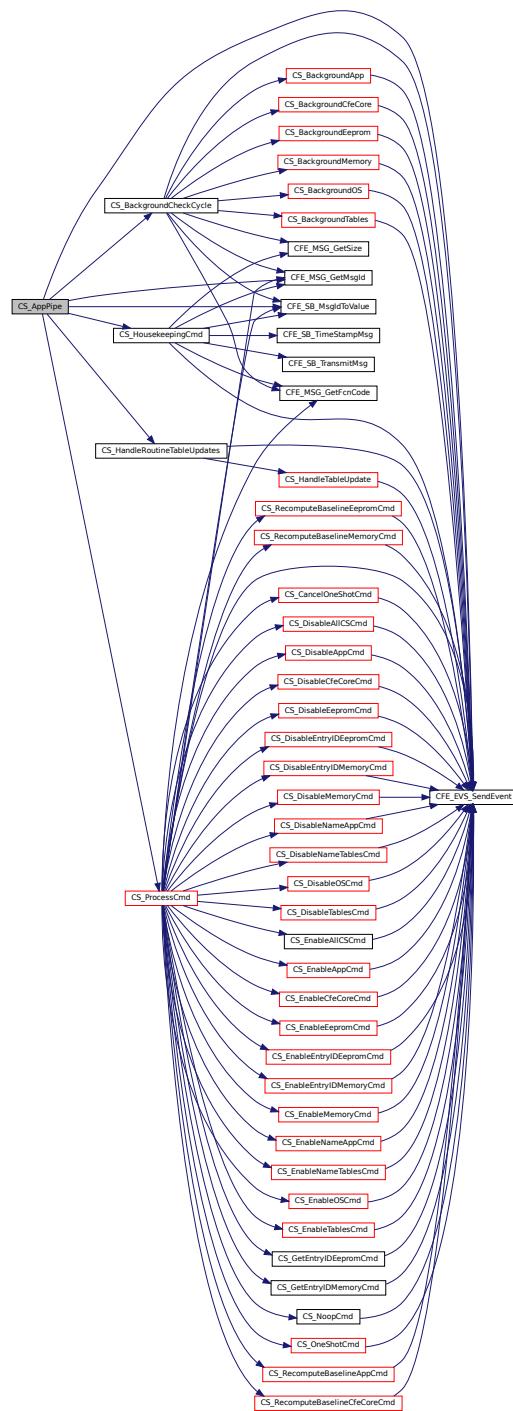
<code>CFE_SUCCESS</code>	Successful execution.
--------------------------	-----------------------

Definition at line 222 of file cs\_app.c.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_MSG_GetMsgId()`, `CFE_SB_INVALID_`←  
`MSG_ID`, `CFE_SB_MsgIdToValue()`, `CFE_SUCCESS`, `CS_HkPacket_Payload_t::CmdErrCounter`, `CS_AppData`, `CS_`←  
`BACKGROUND_CYCLE_MID`, `CS_BackgroundCheckCycle()`, `CS_CMD_MID`, `CS_HandleRoutineTableUpdates()`, `C_`←  
`S_HousekeepingCmd()`, `CS_MID_ERR_EID`, `CS_ProcessCmd()`, `CS_SEND_HK_MID`, `CS_AppData_t::HkPacket`, `C_`←  
`FE_SB_Msg::Msg`, and `CS_HkPacket_t::Payload`.

Referenced by `CS_AppMain()`.

Here is the call graph for this function:



#### 12.11.3.4 CS\_CreateRestoreStatesFromCDS()

```
CFE_Status_t CS_CreateRestoreStatesFromCDS (
    void )
```

Restore tables states from CDS if enabled.

#### Description

Restore CS state of tables from CDS

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

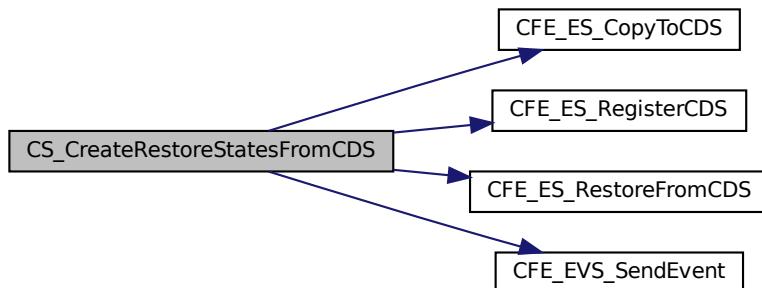
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 615 of file cs\_app.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_ES\_CDS\_ALREADY\_EXISTS, CFE\_ES\_CDS\_BAD\_HAN←DLE, CFE\_ES\_CopyToCDS(), CFE\_ES\_RegisterCDS(), CFE\_ES\_RestoreFromCDS(), CFE\_EVS\_EventType\_ERR←OR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_HkPacket\_Payload\_t::CfeCoreCSState, CS\_AppData, CS\_AP←PS\_TBL\_POWERON\_STATE, CS\_CDS\_NAME, CS\_CFECORE\_CHECKSUM\_STATE, CS\_CR\_CDS\_CPY\_ERR←EID, CS\_CR\_CDS\_REG\_ERR\_EID, CS\_CR\_CDS\_RES\_ERR\_EID, CS\_EEPROM\_TBL\_POWERON\_STATE, CS\_MEMORY\_TBL\_POWERON\_STATE, CS\_NUM\_DATA\_STORE\_STATES, CS\_OSCS\_CHECKSUM\_STATE, CS\_TABLES\_TBL\_POWERON\_STATE, CS\_AppData\_t::DataStoreHandle, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OSCSState, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_AppInit().

Here is the call graph for this function:



**12.11.3.5 CS\_HousekeepingCmd()** void CS\_HousekeepingCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process housekeeping request.

## Description

Processes an on-board housekeeping request message.

## Assumptions, External Events, and Notes:

This command does not affect the command execution counter

## Parameters

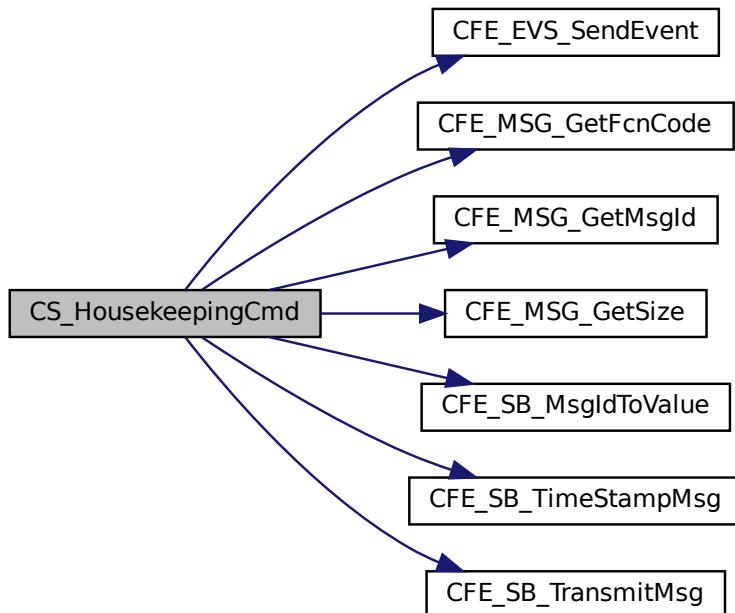
in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

Definition at line 578 of file cs\_app.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_MSG\_GetSize(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_MsgIdToValue(), CFE\_SB\_TimeStampMsg(), CFE\_SB\_TransmitMsg(), CS\_NoArgsCmd\_t::CmdHeader, CS\_AppData, CS\_LEN\_ERR\_EID, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::TlmHeader.

Referenced by CS\_AppPipe().

Here is the call graph for this function:



### 12.11.3.6 CS\_ProcessCmd()

```
void CS_ProcessCmd (
    const CFE_SB_Buffer_t * BufPtr )
```

Command packet processor.

**Description**

Processes all CS commands

**Parameters**

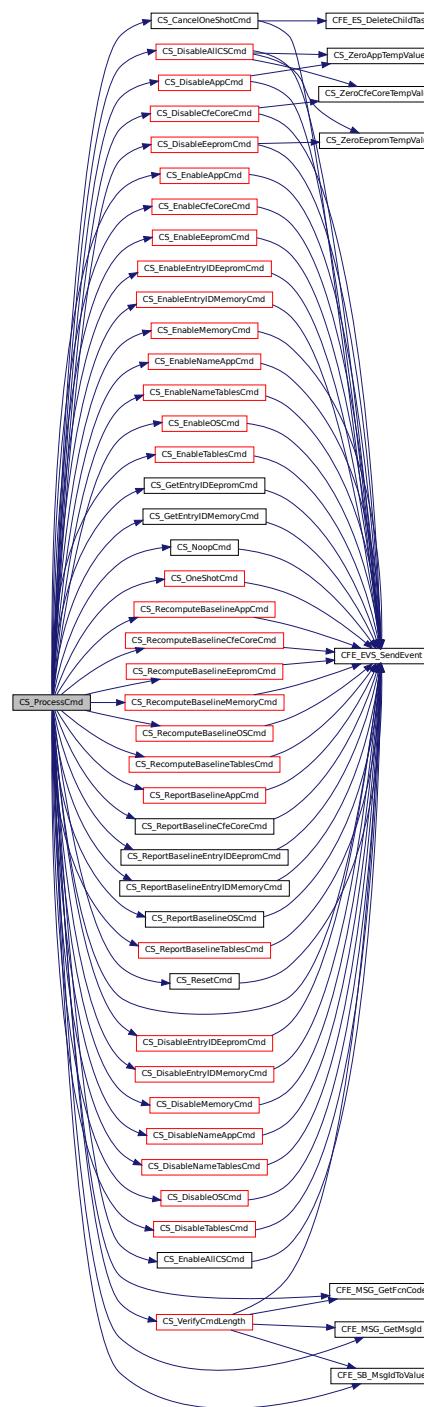
in	<i>BufPtr</i>	A CFE_SB_Buffer_t* pointer that references the software bus pointer. The BufPtr is verified non-null in CS_AppMain.
----	---------------	---

Definition at line 265 of file cs\_app.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_MsgIdToValue(), CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CANCEL\_ONE\_SHOT\_CC, CS\_CancelOneShotCmd(), CS\_CC1\_ERR\_EID, CS\_DISABLE\_ALL\_CS\_CC, CS\_DISABLE\_APPS\_CC, CS\_DISABLE\_CFE\_CORE\_CC, CS\_DISABLE\_EEPROM\_CC, CS\_DISABLE\_ENTRY\_EEPROM\_CC, CS\_DISABLE\_ENTRY\_MEMORY\_CC, CS\_DISABLE\_MEMORY\_CC, CS\_DISABLE\_NAME\_APP\_CC, CS\_DISABLE\_NAME\_TABLE\_CC, CS\_DISABLE\_OS\_CC, CS\_DISABLE\_TABLES\_CC, CS\_DisableAllCSCmd(), CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEepromCmd(), CS\_DisableEntryIDEepromCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_ENABLE\_ALL\_CS\_CC, CS\_ENABLE\_APPS\_CC, CS\_ENABLE\_CFE\_CORE\_CC, CS\_ENABLE\_EEPROM\_CC, CS\_ENABLE\_ENTRY\_EEPROM\_CC, CS\_ENABLE\_ENTRY\_MEMORY\_CC, CS\_ENABLE\_MEMORY\_CC, CS\_ENABLE\_NAME\_APP\_CC, CS\_ENABLE\_NAME\_TABLE\_CC, CS\_ENABLE\_OS\_CC, CS\_ENABLE\_TABLES\_CC, CS\_EnableAllCSCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEepromCmd(), CS\_EnableEntryIDEepromCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_EnableOSCmd(), CS\_EnableTablesCmd(), CS\_GET\_ENTRY\_ID\_EEPROM\_CC, CS\_GET\_ENTRY\_ID\_MEMORY\_CC, CS\_GetEntryIDEepromCmd(), CS\_GetEntryIDMemoryCmd(), CS\_NOOP\_CC, CS\_NoopCmd(), CS\_ONESHOT\_CC, CS\_OneShotCmd(), CS\_RECOMPUTE\_BASELINE\_APP\_CC, CS\_RECOMPUTE\_BASELINE\_CFE\_CORE\_CC, CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC, CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC, CS\_RecomputeBaselineAppCmd(), CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), CS\_RecomputeBaselineTablesCmd(), CS\_REPORT\_BASELINE\_APP\_CC, CS\_REPORT\_BASELINE\_CFE\_CORE\_CC, CS\_REPORT\_BASELINE\_EEPROM\_CC, CS\_REPORT\_BASELINE\_MEMORY\_CC, CS\_ReportBaselineAppCmd(), CS\_ReportBaselineCfeCoreCmd(), CS\_ReportBaselineEntryIDEepromCmd(), CS\_ReportBaselineEntryIDMemoryCmd(), CS\_ReportBaselineOSCmd(), CS\_ReportBaselineTablesCmd(), CS\_RESET\_CC, CS\_ResetCmd(), CS\_VerifyCmdLength(), CS\_AppData\_t::HkPacket, CFE\_SB\_Msg::Msg, and CS\_HkPacket\_t::Payload.

Referenced by CS\_AppPipe().

Here is the call graph for this function:



#### 12.11.3.7 CS\_UpdateCDS()

```
void CS_UpdateCDS (
    void )
```

CFS Checksum (CS) Critical Data Store Update.

#### Description

Checksum application entry point and main process loop.

#### Assumptions, External Events, and Notes:

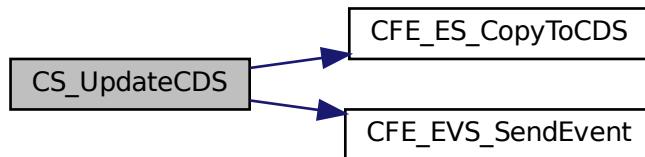
None

Definition at line 709 of file cs\_app.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_ES\_CDS\_BAD\_HANDLE, CFE\_ES\_CopyToCDS(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_RESOURCEID\_TEST\_DEFINED, CFE\_SUCCESS, CS\_HkPacket\_Payload\_t::CfeCoreCSState, CS\_AppData, CS\_NUM\_DATA\_STORE\_STATES, CS\_UPDATE\_CDS\_ER↔R\_EID, CS\_AppData\_t::DataStoreHandle, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OSCSState, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEepromCmd(), CS\_DisableMemoryCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEepromCmd(), CS\_EnableMemoryCmd(), CS\_EnableOSCmd(), and CS\_EnableTablesCmd().

Here is the call graph for this function:



#### 12.11.4 Variable Documentation

##### 12.11.4.1 CS\_AppData CS\_AppData\_t CS\_AppData

Extern the CS\_AppData so all CS files can use it.

Definition at line 52 of file cs\_app.c.

Referenced by CS\_AppInit(), CS\_AppMain(), CS\_AppPipe(), CS\_BackgroundApp(), CS\_BackgroundCfeCore(), CS\_BackgroundCheckCycle(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), CS\_BackgroundTables(), CS\_CancelOneShotCmd(), CS\_CheckRecomputeOneshot(), CS\_ComputeApp(), CS\_ComputeEepromMemory(), CS\_ComputeTables(), CS\_CreateRestoreStatesFromCDS(), CS\_DisableAllCSCmd(), CS\_DisableAppCmd(), CS\_DisableCfeCoreCmd(), CS\_DisableEepromCmd(), CS\_DisableEntryIDEepromCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableOSCmd(), CS\_DisableTablesCmd(), CS\_EnableAllCSCmd(), CS\_EnableAppCmd(), CS\_EnableCfeCoreCmd(), CS\_EnableEepromCmd(), CS\_EnableEntryIDEepromCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_EnableOSCmd(), CS\_EnableTablesCmd(), CS\_FindEnabledAppEntry(), CS\_FindEnabledEepromEntry(), CS\_FindEnabledMemoryEntry(), CS\_FindEnabledTablesEntry(), CS\_GetAppDefTblEntryByName(), CS\_GetAppResTblEntryByName(), CS\_GetEntryIDEepromCmd(), CS\_GetEntryIDMemoryCmd(), CS\_GetTableDefTblEntryByName(), CS\_GetTableResTblEntryByName(), CS\_GoToNextTable(), CS\_HandleRoutineTableUpdates(), CS\_HandleTableUpdate(), CS↔

\_HousekeepingCmd(), CS\_InitAllTables(), CS\_InitializeDefaultTables(), CS\_InitSegments(), CS\_NoopCmd(), CS\_OneShotChildTask(), CS\_OneShotCmd(), CS\_ProcessCmd(), CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), CS\_RecomputeAppChildTask(), CS\_RecomputeBaselineAppCmd(), CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), CS\_RecomputeBaselineOSCmd(), CS\_RecomputeBaselineTablesCmd(), CS\_RecomputeEepromMemoryChildTask(), CS\_RecomputeTablesChildTask(), CS\_ReportBaselineAppCmd(), CS\_ReportBaselineCfeCoreCmd(), CS\_ReportBaselineEntryIDEepromCmd(), CS\_ReportBaselineEntryIDMemoryCmd(), CS\_ReportBaselineOSCmd(), CS\_ReportBaselineTablesCmd(), CS\_ResetCmd(), CS\_SbInit(), CS\_TableInit(), CS\_UpdateCDS(), CS\_VerifyCmdLength(), CS\_ZeroAppTempValues(), CS\_ZeroCfeCoreTempValues(), CS\_ZeroEepromTempValues(), CS\_ZeroMemoryTempValues(), CS\_ZeroOSTempValues(), and CS\_ZeroTablesTempValues().

## 12.12 apps/cs/fsw/src/cs\_app\_cmds.c File Reference

```
#include "cfe.h"
#include "cs_app.h"
#include "cs_events.h"
#include "cs_utils.h"
#include "cs_compute.h"
#include "cs_app_cmds.h"
```

### Functions

- void **CS\_DisableAppCmd** (const **CS\_NoArgsCmd\_t** \*CmdPtr)
   
*Process a disable background checking for the App table command.*
- void **CS\_EnableAppCmd** (const **CS\_NoArgsCmd\_t** \*CmdPtr)
   
*Process an enable background checking for the App table command.*
- void **CS\_ReportBaselineAppCmd** (const **CS\_AppNameCmd\_t** \*CmdPtr)
   
*Process a report baseline of an App command.*
- void **CS\_RecomputeBaselineAppCmd** (const **CS\_AppNameCmd\_t** \*CmdPtr)
   
*Process a recopmpute baseline of an app command.*
- void **CS\_DisableNameAppCmd** (const **CS\_AppNameCmd\_t** \*CmdPtr)
   
*Process a disable background checking for an App entry command.*
- void **CS\_EnableNameAppCmd** (const **CS\_AppNameCmd\_t** \*CmdPtr)
   
*Process an enable background checking for an App entry command.*

### 12.12.1 Detailed Description

The CFS Checksum (CS) Application's commands for checking App

### 12.12.2 Function Documentation

#### 12.12.2.1 **CS\_DisableAppCmd()** void CS\_DisableAppCmd ( const **CS\_NoArgsCmd\_t** \* CmdPtr )

Process a disable background checking for the App table command.

##### Description

Disables background checking for the App table

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Tables, and App) to occur, the table must be enabled and overall checksumming must be enabled.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

**See also**

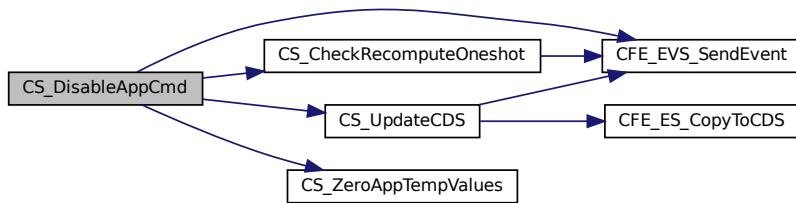
[CS\\_DISABLE\\_APPS\\_CC](#)

Definition at line 47 of file cs\_app\_cmds.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_APP\_INF\_E\_ID, CS\_STATE\_DISABLED, CS\_UpdateCDS(), CS\_ZeroAppTempValues(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.12.2.2 CS\_DisableNameAppCmd()** void CS\_DisableNameAppCmd ( const CS\_AppNameCmd\_t \* CmdPtr )

Process a disable background checking for an App entry command.

**Description**

Disables the specified App entry to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

See also

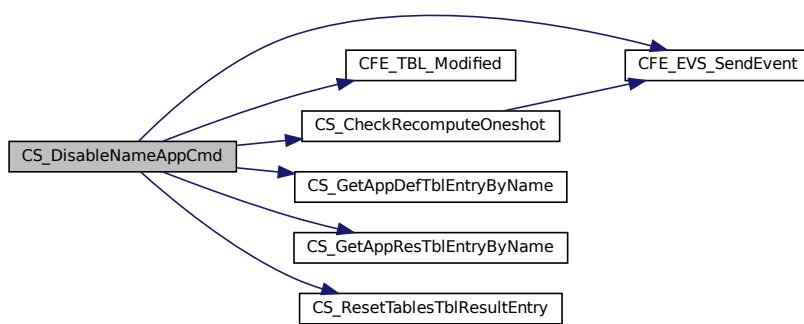
[CS\\_DISABLE\\_NAME\\_APP\\_CC](#)

Definition at line 188 of file cs\_app\_cmds.c.

References CS\_AppData\_t::AppResTablesTblPtr, CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_DISABLE\_APP\_NAME\_INF\_EID, CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetAppDefTblEntryByName(), CS\_GetAppResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::HkPacket, CS\_AppNameCmd\_Payload\_t::Name, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppNameCmd\_t::Payload, CS\_Def\_App\_Table\_Entry\_t::State, CS\_Res\_App\_Table\_Entry\_t::State, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.12.2.3 CS\_EnableAppCmd()** void CS\_EnableAppCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process an enable background checking for the App table command.

#### Description

Allows the App table to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Tables, and App) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

See also

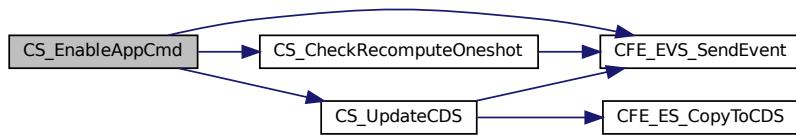
[CS\\_ENABLE\\_APPS\\_CC](#)

Definition at line 68 of file cs\_app\_cmds.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_APP\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.12.2.4 CS\_EnableNameAppCmd()** `void CS_EnableNameAppCmd (const CS_AppNameCmd_t * CmdPtr )`

Process an enable background checking for an App entry command.

Description

Allows the specified table to be background checksummed.

Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

See also

[CS\\_ENABLE\\_NAME\\_APP\\_CC](#)

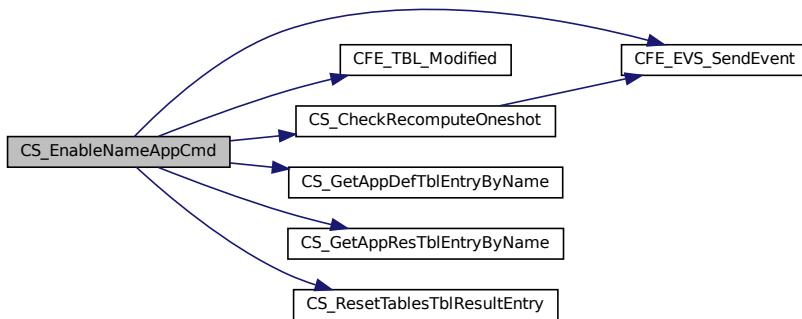
Definition at line 238 of file cs\_app\_cmds.c.

References CS\_AppData\_t::AppResTablesTblPtr, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_ENABLE\_APP\_NAME\_INF\_EID, CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetAppDefTblEntryByName(), CS\_GetAppResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_ENABLED, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::HkPacket, CS\_AppNameCmd

\_Payload\_t::Name, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppNameCmd\_t::Payload, CS\_Def\_App←Table\_Entry\_t::State, and CS\_Res\_App\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.12.2.5 CS\_RecomputeBaselineAppCmd()** `void CS_RecomputeBaselineAppCmd (const CS_AppNameCmd_t * CmdPtr )`

Process a recopmpute baseline of an app command.

#### Description

Recomputes the checksum of an app and use that value as the new baseline for that entry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

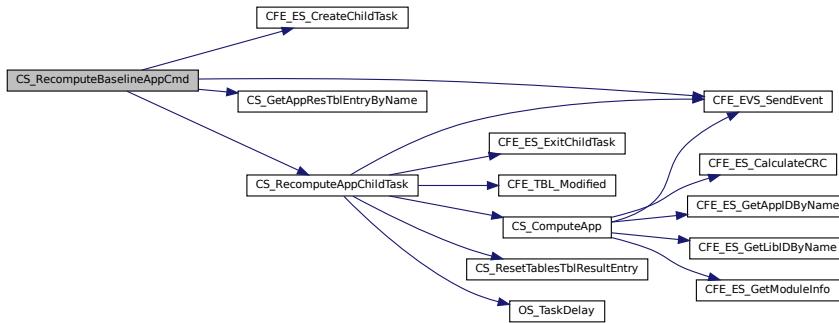
[CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#)

Definition at line 126 of file cs\_app\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_ES\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_APP\_TABLE, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_GetAppResTblEntryByName(), CS\_RECOMP\_APP\_TASK\_NAME, CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_APP\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID, CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID, CS\_RecomputeAppChildTask(), CS\_AppData\_t::HkPacket, CS\_AppNameCmd\_Payload\_t::Name, CS\_HkPacket\_Payload\_t::OneShotInProgress, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppNameCmd\_t::Payload, CS\_AppData\_t::RecomputeAppEntryPtr, and CS\_HkPacket\_Payload\_t::RecomputeInProgress.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.12.2.6 CS\_ReportBaselineAppCmd()** void CS\_ReportBaselineAppCmd ( const CS\_AppNameCmd\_t \* CmdPtr )

Process a report baseline of an App command.

#### Description

Reports the baseline checksum of the specified app if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

#### See also

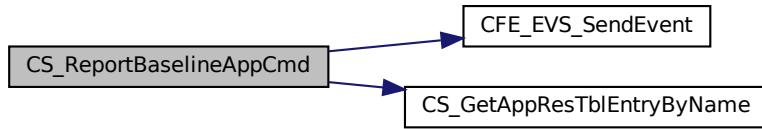
[CS\\_REPORT\\_BASELINE\\_APP\\_CC](#)

Definition at line 88 of file cs\_app\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_App\_Table\_Entry\_t::ComparisonValue, CS\_Res\_App\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_APP\_INF\_EID, CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID, CS\_GetAppResTblEntryByName(), CS\_NO\_BASELINE\_APP\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_AppNameCmd\_Payload\_t::Name, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, and CS\_AppNameCmd\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.13 apps/cs/fsw/src/cs\_app\_cmds.h File Reference

```
#include "cfe.h"
#include "cs_msg.h"
```

### Functions

- void [CS\\_DisableAppCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the App table command.*
- void [CS\\_EnableAppCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for the App table command.*
- void [CS\\_ReportBaselineAppCmd](#) (const [CS\\_AppNameCmd\\_t](#) \*CmdPtr)  
*Process a report baseline of an App command.*
- void [CS\\_RecomputeBaselineAppCmd](#) (const [CS\\_AppNameCmd\\_t](#) \*CmdPtr)  
*Process a recocompute baseline of an app command.*
- void [CS\\_DisableNameAppCmd](#) (const [CS\\_AppNameCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for an App entry command.*
- void [CS\\_EnableNameAppCmd](#) (const [CS\\_AppNameCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for an App entry command.*

### 12.13.1 Detailed Description

Specification for the CFS app cmds

### 12.13.2 Function Documentation

#### 12.13.2.1 [CS\\_DisableAppCmd\(\)](#) void CS\_DisableAppCmd (

```
    const CS\_NoArgsCmd\_t * CmdPtr )
```

Process a disable background checking for the App table command.

##### Description

Disables background checking for the App table

##### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Tables, and App) to occur, the table must be enabled and overall checksumming must be enabled.

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

### See also

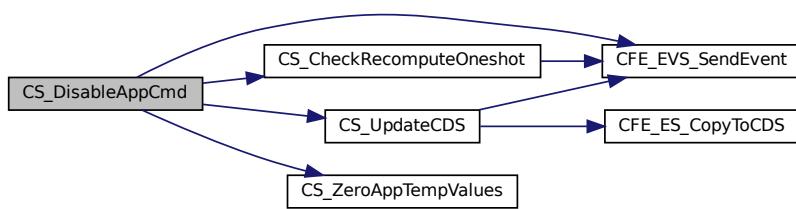
[CS\\_DISABLE\\_APPS\\_CC](#)

Definition at line 47 of file `cs_app_cmds.c`.

References `CS_HkPacket_Payload_t::AppCSState`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CS_HkPacket_Payload_t::CmdCounter`, `CS_AppData`, `CS_CheckRecomputeOneshot()`, `CS_DISABLE_APP_INF_ID`, `CS_STATE_DISABLED`, `CS_UpdateCDS()`, `CS_ZeroAppTempValues()`, `CS_AppData_t::HkPacket`, and `CS_HkPacket_t::Payload`.

Referenced by `CS_ProcessCmd()`.

Here is the call graph for this function:



**12.13.2.2 CS\_DisableNameAppCmd()** `void CS_DisableNameAppCmd (`  
 `const CS_AppNameCmd_t * CmdPtr )`

Process a disable background checking for an App entry command.

### Description

Disables the specified App entry to be background checksummed.

### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

See also

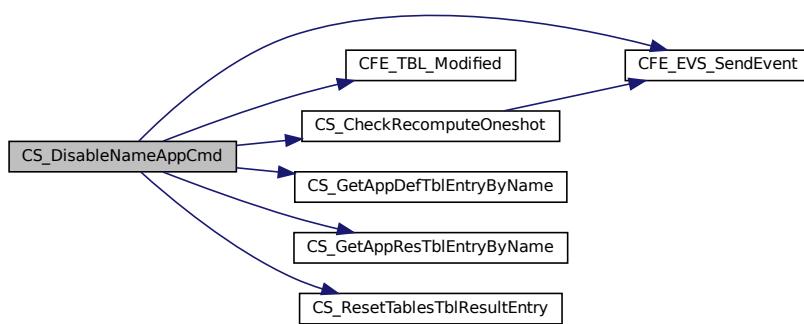
[CS\\_DISABLE\\_NAME\\_APP\\_CC](#)

Definition at line 188 of file cs\_app\_cmds.c.

References CS\_AppData\_t::AppResTablesTblPtr, CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_DISABLE\_APP\_NAME\_INF\_EID, CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetAppDefTblEntryByName(), CS\_GetAppResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::HkPacket, CS\_AppNameCmd\_Payload\_t::Name, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppNameCmd\_t::Payload, CS\_Def\_App\_Table\_Entry\_t::State, CS\_Res\_App\_Table\_Entry\_t::State, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.13.2.3 CS\_EnableAppCmd() void CS\_EnableAppCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process an enable background checking for the App table command.

#### Description

Allows the App table to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Tables, and App) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

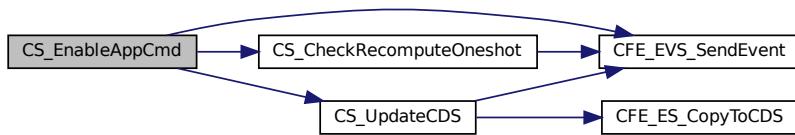
[CS\\_ENABLE\\_APPS\\_CC](#)

Definition at line 68 of file cs\_app\_cmds.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_APP\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.13.2.4 CS\_EnableNameAppCmd()** `void CS_EnableNameAppCmd (const CS_AppNameCmd_t * CmdPtr )`

Process an enable background checking for an App entry command.

## Description

Allows the specified table to be background checksummed.

## Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

[CS\\_ENABLE\\_NAME\\_APP\\_CC](#)

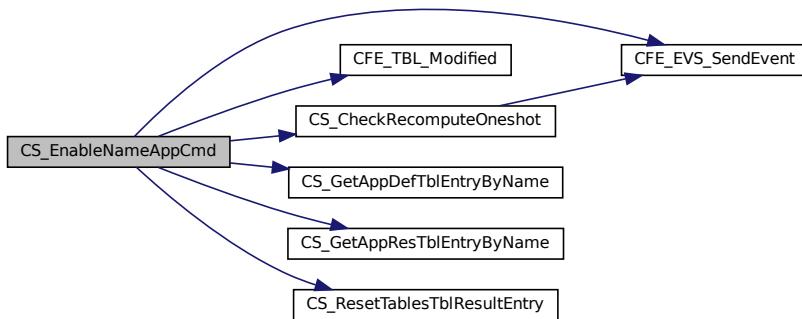
Definition at line 238 of file cs\_app\_cmds.c.

References CS\_AppData\_t::AppResTablesTblPtr, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_ENABLE\_APP\_NAME\_INF\_EID, CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetAppDefTblEntryByName(), CS\_GetAppResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_ENABLED, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::HkPacket, CS\_AppNameCmd

\_Payload\_t::Name, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppNameCmd\_t::Payload, CS\_Def\_App←Table\_Entry\_t::State, and CS\_Res\_App\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.13.2.5 CS\_RecomputeBaselineAppCmd()** `void CS_RecomputeBaselineAppCmd (const CS_AppNameCmd_t * CmdPtr )`

Process a recopmpute baseline of an app command.

#### Description

Recomputes the checksum of an app and use that value as the new baseline for that entry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

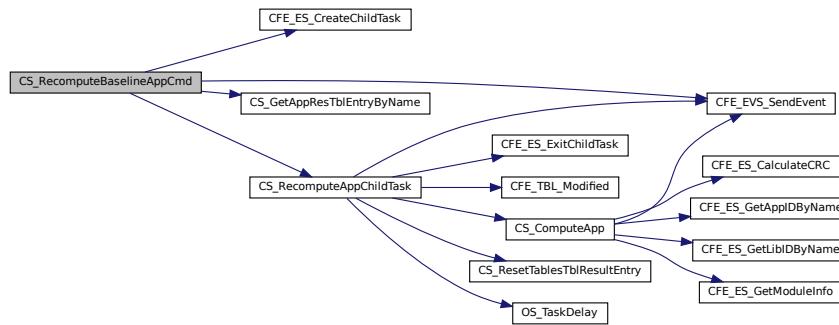
[CS\\_RECOMPUTE\\_BASELINE\\_APP\\_CC](#)

Definition at line 126 of file cs\_app\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_ES\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_APP\_TABLE, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_GetAppResTblEntryByName(), CS\_RECOMP\_APP\_TASK\_NAME, CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_APP\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID, CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID, CS\_RecomputeAppChildTask(), CS\_AppData\_t::HkPacket, CS\_AppNameCmd\_Payload\_t::Name, CS\_HkPacket\_Payload\_t::OneShotInProgress, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppNameCmd\_t::Payload, CS\_AppData\_t::RecomputeAppEntryPtr, and CS\_HkPacket\_Payload\_t::RecomputeInProgress.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.13.2.6 CS\_ReportBaselineAppCmd()** void CS\_ReportBaselineAppCmd ( const CS\_AppNameCmd\_t \* CmdPtr )

Process a report baseline of an App command.

#### Description

Reports the baseline checksum of the specified app if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

#### See also

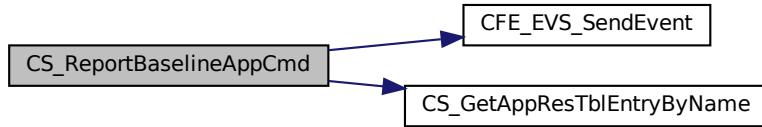
[CS\\_REPORT\\_BASELINE\\_APP\\_CC](#)

Definition at line 88 of file cs\_app\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_App\_Table\_Entry\_t::ComparisonValue, CS\_Res\_App\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_APP\_INF\_EID, CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID, CS\_GetAppResTblEntryByName(), CS\_NO\_BASELINE\_APP\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_AppNameCmd\_Payload\_t::Name, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, and CS\_AppNameCmd\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.14 apps/cs/fsw/src/cs\_cmds.c File Reference

```
#include "cfe.h"
#include "cs_app.h"
#include "cs_events.h"
#include "cs_cmds.h"
#include "cs_utils.h"
#include "cs_compute.h"
```

### Functions

- void `CS_NoopCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process noop command.*
- void `CS_ResetCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process reset counters command.*
- void `CS_BackgroundCheckCycle` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*process a background checking cycle*
- void `CS_DisableAICSCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process a disable overall background checking command.*
- void `CS_EnableAICSCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process an enable overall background checking command.*
- void `CS_DisableCfeCoreCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process a disable background checking for the cFE core code segment command.*
- void `CS_EnableCfeCoreCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process an enable background checking for the cFE core code segment command.*
- void `CS_DisableOSCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process a disable background checking for the OS code segment command.*
- void `CS_EnableOSCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process an enable background checking for the OS code segment command.*
- void `CS_ReportBaselineCfeCoreCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process a report baseline of the cFE Core command.*
- void `CS_ReportBaselineOSCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process a report baseline of the OS command.*
- void `CS_RecomputeBaselineCfeCoreCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process a recocompute baseline of the cFE core code segment command.*
- void `CS_RecomputeBaselineOSCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)

- Process a recompute baseline of the OS command.*
- void [CS\\_OneShotCmd](#) (const [CS\\_OneShotCmd\\_t](#) \*CmdPtr)  
*Process a start a one shot checksum command.*
  - void [CS\\_CancelOneShotCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a cancel one shot checksum command.*

### 12.14.1 Detailed Description

The CFS Checksum (CS) Application's commands for OS code segment, the cFE core code segment, and for CS in general

### 12.14.2 Function Documentation

#### 12.14.2.1 [CS\\_BackgroundCheckCycle\(\)](#) void CS\_BackgroundCheckCycle (

```
    const CS\_NoArgsCmd\_t * CmdPtr )
```

process a background checking cycle

##### Description

Processes a background checking cycle when the scheduler tell CS.

##### Assumptions, External Events, and Notes:

None

##### Parameters

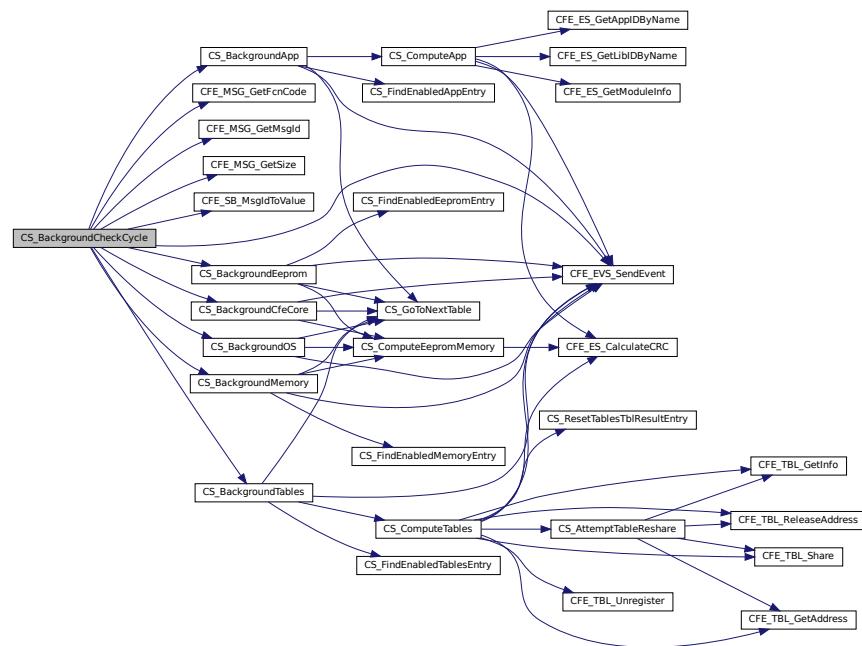
in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

Definition at line 83 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFEMSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_MSG\_GetSize(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_MsgIdToValue(), CS\_HkPacket\_Payload\_t::ChecksumState, CS\_NoArgsCmd\_t::CmdHeader, CS\_APP\_TABLE, CS\_AppData, CS\_BackgroundApp(), CS\_BackgroundCfeCore(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), CS\_BackgroundTables(), CS\_BKGND\_COMPUTE\_PROG\_INF\_EID, CS\_CFECORE, CS\_EEPROM\_TABLE, CS\_LEN\_ERR\_EID, CS\_MEMORY\_TABLE, CS\_NUM\_TABLES, CS\_OSCORE, CS\_STATE\_ENABLED, CS\_TABLES\_TABLE, CS\_HkPacket\_Payload\_t::CurrentCSTable, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_Payload\_t::PassCounter, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::RecomputeInProgress.

Referenced by CS\_AppPipe().

Here is the call graph for this function:



```
12.14.2.2 CS_CancelOneShotCmd() void CS_CancelOneShotCmd ( const CS_NoArgsCmd_t * CmdPtr )
```

Process a cancel one shot checksum command.

## Description

Cancel a one shot command, if a one shot calculation is taking place

### **Assumptions, External Events, and Notes:**

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

CS CANCEL ONE SHOT CC

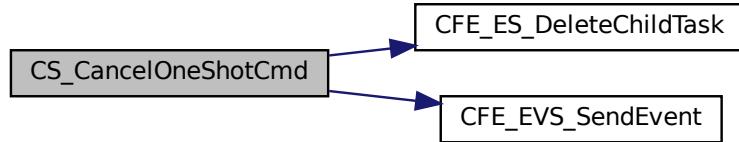
Definition at line 521 of file cs\_cmds.c.

References CFE\_ES\_DeleteChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_AppData\_t::ChildTaskID, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_CANCEL\_LLED\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload,

and CS\_HkPacket\_Payload\_t::RecomputeInProgress.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



#### 12.14.2.3 CS\_DisableAllCSCmd()

```
void CS_DisableAllCSCmd (
```

```
    const CS_NoArgsCmd_t * CmdPtr )
```

Process a disable overall background checking command.

##### Description

Disables all background checking in CS

##### Assumptions, External Events, and Notes:

##### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

##### See also

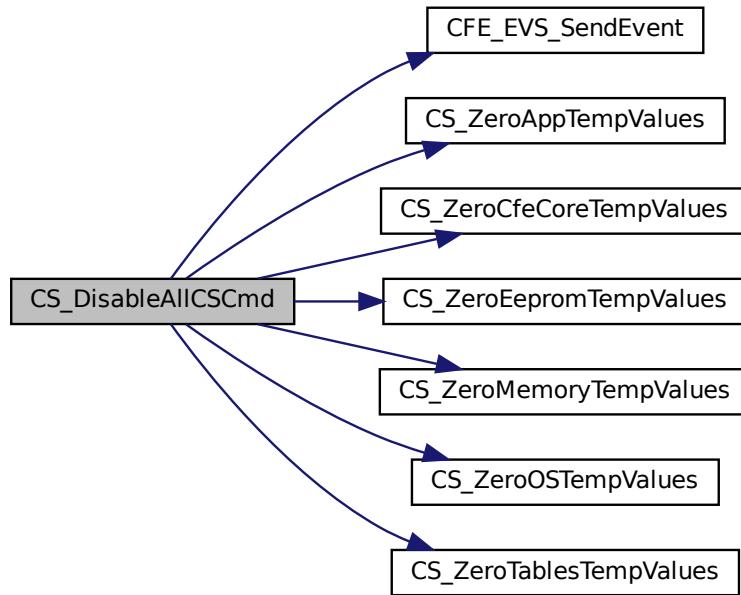
[CS\\_DISABLE\\_ALL\\_CS\\_CC](#)

Definition at line 191 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::Checksum←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_DISABLE\_ALL\_INF\_EID, CS\_STATE\_DISABLED, CS\_ZeroAppTempValues(), CS\_ZeroCfeCoreTempValues(), CS\_ZeroEepromTempValues(), CS\_ZeroMemoryTemp←Values(), CS\_ZeroOSTempValues(), CS\_ZeroTablesTempValues(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.14.2.4 CS\_DisableCfeCoreCmd()** void CS\_DisableCfeCoreCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process a disable background checking for the cFE core code segment command.

#### Description

Disables background checking for the cFE core code segment

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

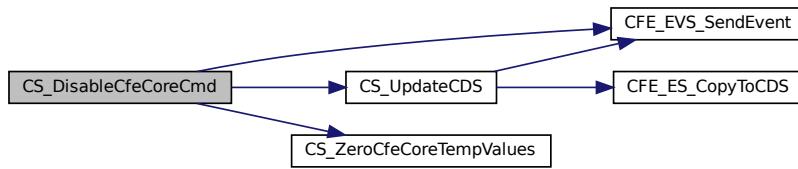
**See also**

[CS\\_DISABLE\\_CFE\\_CORE\\_CC](#)

Definition at line 229 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CfeCoreCS←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_DISABLE\_CFECore\_INF\_EID, CS\_STATE\_DIS←ABLED, CS\_UpdateCDS(), CS\_ZeroCfeCoreTempValues(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.14.2.5 CS\_DisableOSCmd()

```
void CS_DisableOSCmd (
```

```
    const CS_NoArgsCmd_t * CmdPtr )
```

Process a disable background checking for the OS code segment command.

**Description**

Disables background checking for the OS code segment

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

**Parameters**

in	<code>CmdPtr</code>	Command pointer, verified non-null in CS_AppMain
----	---------------------	--

## See also

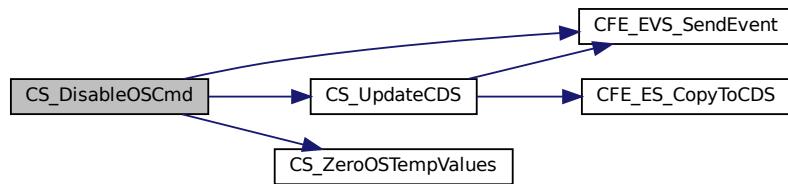
[CS\\_DISABLE\\_OS\\_CC](#)

Definition at line 268 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_DISABLE\_OS\_INF\_EID, CS\_STATE\_DISABLED, CS\_UpdateCDS(), CS\_ZeroOSTempValues(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OSCSSState, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.14.2.6 CS\_EnableAllCSCmd()** `void CS_EnableAllCSCmd (`  
`const CS_NoArgsCmd_t * CmdPtr )`

Process an enable overall background checking command.

## Description

Allows background checking to take place.

## Assumptions, External Events, and Notes:

None

## Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in CS_AppMain
----	---------------------	--

**See also**[CS\\_ENABLE\\_ALL\\_CS\\_CC](#)

Definition at line 215 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::Checksum←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_ENABLE\_ALL\_INF\_EID, CS\_STATE\_ENABLED, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.14.2.7 CS\_EnableCfeCoreCmd()** void CS\_EnableCfeCoreCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process an enable background checking for the cFE core code segment command.

**Description**

Allows the cFE Core code segment to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

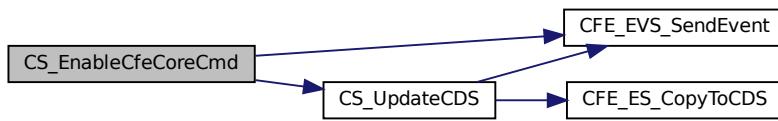
[CS\\_ENABLE\\_CFE\\_CORE\\_CC](#)

Definition at line 249 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CfeCoreCS←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_ENABLE\_CFECore\_INF\_EID, CS\_STATE\_ENA←BLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.14.2.8 CS\_EnableOSCmd()** `void CS_EnableOSCmd (`  
 `const CS_NoArgsCmd_t * CmdPtr )`

Process an enable background checking for the OS code segment command.

## Description

Allows the OS code segment to be background checksummed.

## Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

**See also**

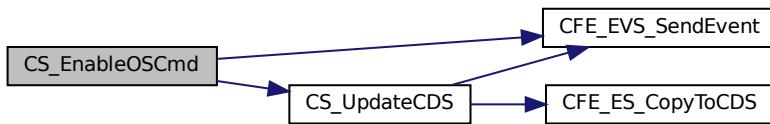
[CS\\_ENABLE\\_OS\\_CC](#)

Definition at line 288 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_ENABLE\_OS\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OSCSState, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.14.2.9 CS\_NoopCmd()** void CS\_NoopCmd (

const [CS\\_NoArgsCmd\\_t](#) \* CmdPtr )

Process noop command.

**Description**

Processes a noop ground command.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

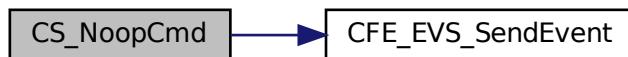
[CS\\_NOOP\\_CC](#)

Definition at line 49 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_MAJOR\_VERSION, CS\_MINOR\_VERSION, CS\_MISSION\_REV, CS\_NOOP\_INF\_EID, CS\_REV←ISION, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.14.2.10 CS\_OneShotCmd()** void CS\_OneShotCmd (   
     const CS\_OneShotCmd\_t \* CmdPtr )

Process a start a one shot checksum command.

## Description

Starts a one shot checksum on given address and size, and reports checksum in telemetry and an event message.

## Assumptions, External Events, and Notes:

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

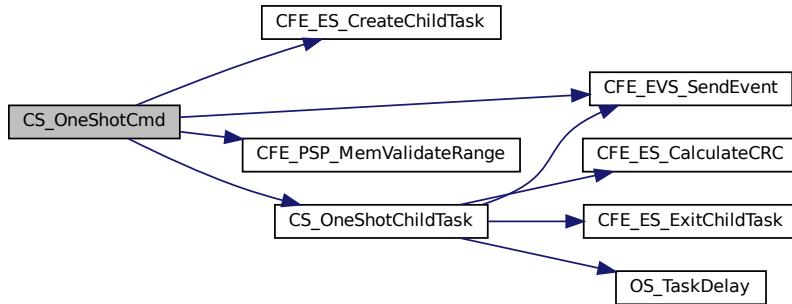
[CS\\_ONE\\_SHOT\\_CC](#)

Definition at line 445 of file cs\_cmds.c.

References CS\_OneShotCmd\_Payload\_t::Address, CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_PSP\_MEM\_ANY, CFE\_PSP\_MemValidateRange(), CFE\_SUCCESS, CS\_AppData\_t::ChildTaskID, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_ONESHOT\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID, CS\_ONESHOT\_STARTED\_DBG\_EID, CS\_ONESHOT\_TASK\_NAME, CS\_OneShotChildTask(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::LastOneShotAddress, CS\_HkPacket\_Payload\_t::LastOneShotChecksum, CS\_HkPacket\_Payload\_t::LastOneShotMaxBytesPerCycle, CS\_HkPacket\_Payload\_t::LastOneShotSize, CS\_AppData\_t::MaxBytesPerCycle, CS\_OneShotCmd\_Payload\_t::MaxBytesPerCycle, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_OneShotCmd\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, and CS\_OneShotCmd\_Payload\_t::Size.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.14.2.11 CS\_RecomputeBaselineCfeCoreCmd()

```
void CS_RecomputeBaselineCfeCoreCmd (
```

```
    const CS_NoArgsCmd_t * CmdPtr )
```

Process a recompute baseline of the cFE core code segment command.

#### Description

Recomputes the checksum of the cFE core code segment and use that value as the new baseline for the cFE core.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

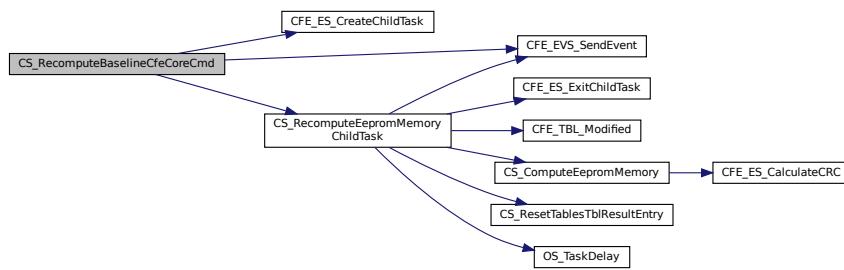
#### See also

[CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#)

Definition at line 349 of file cs\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_ES\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::CfeCoreCodeSeg, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CFECORE, CS\_CHILD\_TASK\_PRIORITY, CS\_RECOMPUTE\_CFECORE\_TASK\_NAME, CS\_RECOMPUTE\_CFECORE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_CFECORE\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_CFECORE\_STARTED\_DBG\_EID, CS\_RecomputeEepromMemoryChildTask(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, and CS\_HkPacket\_Payload\_t::RecomputeInProgress. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.14.2.12 CS\_RecomputeBaselineOSCmd()** `void CS_RecomputeBaselineOSCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process a recompute baseline of the OS command.

#### Description

Recomputes the checksum of the OS code segment and use that value as the new baseline for the OS.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in <code>CS_AppMain</code>
----	---------------------	---

#### See also

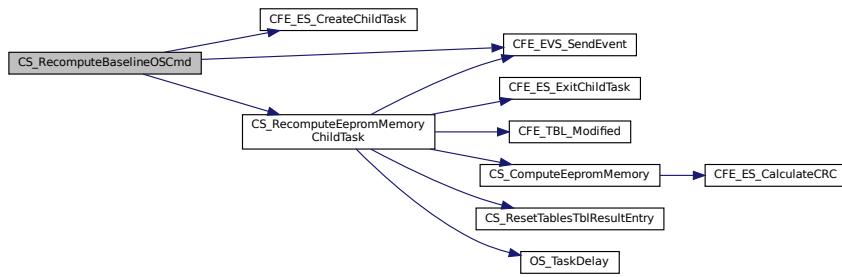
[CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#)

Definition at line 398 of file `cs_cmds.c`.

References `CFE_ES_CreateChildTask()`, `CFE_EVS_EventType_DEBUG`, `CFE_EVS_EventType_ERROR`, `CFE_ES_SendEvent()`, `CFE_PLATFORM_ES_DEFAULT_STACK_SIZE`, `CFE_SUCCESS`, `CS_AppData_t::ChildTaskEntry::ID`, `CS_AppData_t::ChildTaskTable`, `CS_HkPacket_Payload_t::CmdCounter`, `CS_HkPacket_Payload_t::CmdErrCounter`, `CS_AppData`, `CS_CHILD_TASK_PRIORITY`, `CS_OSCORE`, `CS_RECOMP_OS_TASK_NAME`, `CS_RECOMPUTE_OS_CHDTASK_ERR_EID`, `CS_RECOMPUTE_OS_CREATE_CHDTASK_ERR_EID`, `CS_RECOMPUTE_OS_STARTED_DBG_EID`, `CS_RecomputeEepromMemoryChildTask()`, `CS_AppData_t::HkPacket`, `CS_HkPacket_Payload_t::OneShotInProgress`, `CS_AppData_t::OSCodeSeg`, `CS_HkPacket_t::Payload`, `CS_AppData_t::RecomputeEepromMemoryEntryPtr`, and `CS_HkPacket_Payload_t::RecomputeInProgress`.

Referenced by `CS_ProcessCmd()`.

Here is the call graph for this function:



**12.14.2.13 CS\_ReportBaselineCfeCoreCmd()** `void CS_ReportBaselineCfeCoreCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process a report baseline of the cFE Core command.

#### Description

Reports the baseline checksum of the cFE core code segment if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

[CS\\_REPORT\\_BASELINE\\_CFE\\_CORE\\_CC](#)

Definition at line 307 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_AppData\_t::CfeCoreCodeSeg, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_CFECORE\_INF\_EID, CS\_NO\_BASELINE\_CFECORE\_INF\_EID, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



```
12.14.2.14 CS_ReportBaselineOSCmd() void CS_ReportBaselineOSCmd (
    const CS_NoArgsCmd_t * CmdPtr )
```

Process a report baseline of the OS command.

#### Description

Reports the baseline checksum of the OS code segment if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

[CS\\_REPORT\\_BASELINE\\_OS\\_CC](#)

Definition at line 328 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_OS\_INF\_EID, CS\_NO\_BASELINE\_OS\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_AppData\_t::OSCodeSeg, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



```
12.14.2.15 CS_ResetCmd() void CS_ResetCmd (
    const CS_NoArgsCmd_t * CmdPtr )
```

Process reset counters command.

#### Description

Processes a reset counters ground command which will reset the checksum command error and command execution counters to zero. It also resets all checksum error counters and the passes completed counter.

#### Assumptions, External Events, and Notes:

None

**Parameters**

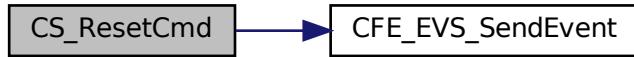
in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

**See also**[CS\\_RESET\\_CC](#)

Definition at line 62 of file cs\_cmds.c.

References CS\_HkPacket\_Payload\_t::AppCSErrCounter, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CfeCoreCSErrCounter, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_RESET\_DBG\_EID, CS\_HkPacket\_Payload\_t::EepromCSErrCounter, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSErrCounter, CS\_HkPacket\_Payload\_t::OSCSErrCounter, CS\_HkPacket\_Payload\_t::PassCounter, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSErrCounter. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.15 apps/cs/fsw/src/cs\_cmds.h File Reference

```
#include "cfe.h"
#include "cs_msg.h"
```

**Functions**

- void [CS\\_NoopCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process noop command.*
- void [CS\\_ResetCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process reset counters command.*
- void [CS\\_BackgroundCheckCycle](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*process a background checking cycle*
- void [CS\\_DisableAICSCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable overall background checking command.*
- void [CS\\_EnableAICSCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable overall background checking command.*
- void [CS\\_DisableCfeCoreCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the cFE core code segment command.*
- void [CS\\_EnableCfeCoreCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for the cFE core code segment command.*
- void [CS\\_DisableOSCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the OS code segment command.*

- void `CS_EnableOSCmd` (const `CS_NoArgsCmd_t` \*`CmdPtr`)
 

*Process an enable background checking for the OS code segment command.*
- void `CS_ReportBaselineCfeCoreCmd` (const `CS_NoArgsCmd_t` \*`CmdPtr`)
 

*Process a report baseline of the cFE Core command.*
- void `CS_ReportBaselineOSCmd` (const `CS_NoArgsCmd_t` \*`CmdPtr`)
 

*Process a report baseline of the OS command.*
- void `CS_RecomputeBaselineCfeCoreCmd` (const `CS_NoArgsCmd_t` \*`CmdPtr`)
 

*Process a recompute baseline of the cFE core code segment command.*
- void `CS_RecomputeBaselineOSCmd` (const `CS_NoArgsCmd_t` \*`CmdPtr`)
 

*Process a recompute baseline of the OS command.*
- void `CS_OneShotCmd` (const `CS_OneShotCmd_t` \*`CmdPtr`)
 

*Process a start a one shot checksum command.*
- void `CS_CancelOneShotCmd` (const `CS_NoArgsCmd_t` \*`CmdPtr`)
 

*Process a cancel one shot checksum command.*

### 12.15.1 Detailed Description

Specification for the CFS generic cmdcs

### 12.15.2 Function Documentation

**12.15.2.1 CS\_BackgroundCheckCycle()**    void `CS_BackgroundCheckCycle` (
   
    const `CS_NoArgsCmd_t` \* `CmdPtr` )
   
process a background checking cycle

#### Description

Processes a background checking cycle when the scheduler tell CS.

#### Assumptions, External Events, and Notes:

None

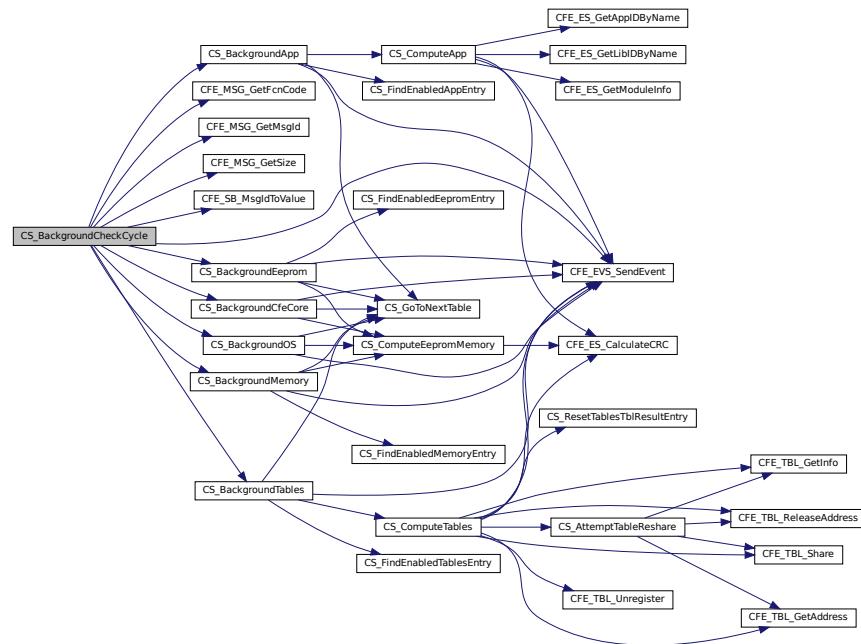
#### Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in <code>CS_AppMain</code>
----	---------------------	---

Definition at line 83 of file `cs_cmds.c`.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_MSG_GetFcnCode()`, `CFE_MSG_GetMsgId()`, `CFE_MSG_GetSize()`, `CFE_SB_INVALID_MSG_ID`, `CFE_SB_MsgIdToValue()`, `CS_HkPacket_Payload_t::ChecksumState`, `CS_NoArgsCmd_t::CmdHeader`, `CS_APP_TABLE`, `CS_AppData`, `CS_BackgroundApp()`, `CS_BackgroundCfeCore()`, `CS_BackgroundEeprom()`, `CS_BackgroundMemory()`, `CS_BackgroundOS()`, `CS_BackgroundTables()`, `CS_BKGND_COMPUTE_PROG_INF_EID`, `CS_CFECORE`, `CS_EEPROM_TABLE`, `CS_LEN_ERR_EID`, `CS_MEMORY_TABLE`, `CS_NUM_TABLES`, `CS_OSCORE`, `CS_STATE_ENABLED`, `CS_TABLES_TABLE`, `CS_HkPacket_Payload_t::CurrentCSTable`, `CS_HkPacket_Payload_t::CurrentEntryInTable`, `CS_AppData_t::HkPacket`, `CS_HkPacket_Payload_t::OneShotInProgress`, `CS_HkPacket_Payload_t::PassCounter`, `CS_HkPacket_t::Payload`, and `CS_HkPacket_Payload_t::RecomputeInProgress`.  
Referenced by `CS_AppPipe()`.

Here is the call graph for this function:



```
12.15.2.2 CS_CancelOneShotCmd() void CS_CancelOneShotCmd ( const CS_NoArgsCmd_t * CmdPtr )
```

Process a cancel one shot checksum command.

## Description

Cancel a one shot command, if a one shot calculation is taking place

### **Assumptions, External Events, and Notes:**

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

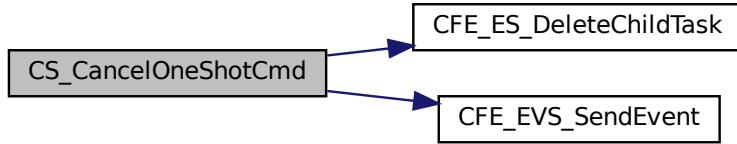
#### See also

CS CANCEL ONE SHOT CC

Definition at line 521 of file cs\_cmds.c.

References CFE\_ES\_DeleteChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_AppData\_t::ChildTaskID, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_CANCEL\_LLED\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload,

and CS\_HkPacket\_Payload\_t::RecomputeInProgress.  
Referenced by CS\_ProcessCmd().  
Here is the call graph for this function:



**12.15.2.3 CS\_DisableAllCSCmd()** `void CS_DisableAllCSCmd (`  
`const CS_NoArgsCmd_t * CmdPtr )`

Process a disable overall background checking command.

#### Description

Disables all background checking in CS

#### Assumptions, External Events, and Notes:

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

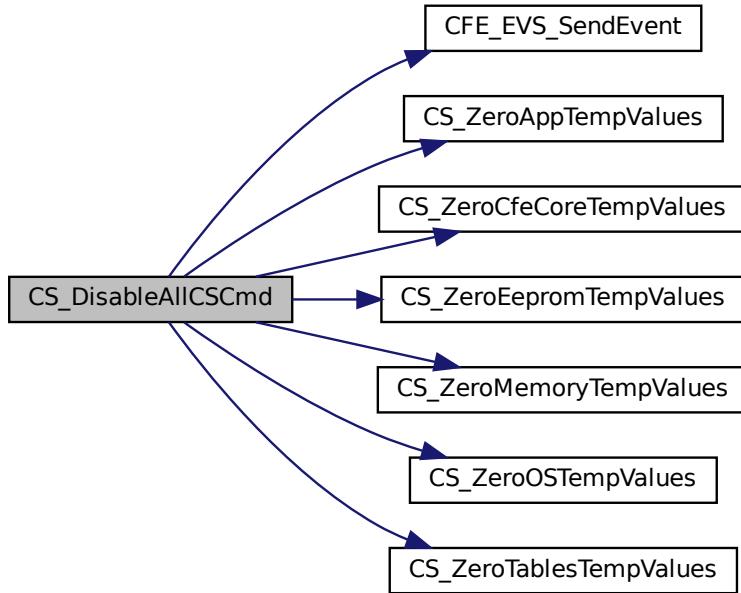
[CS\\_DISABLE\\_ALL\\_CS\\_CC](#)

Definition at line 191 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::Checksum←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_DISABLE\_ALL\_INF\_EID, CS\_STATE\_DISABLED, CS\_ZeroAppTempValues(), CS\_ZeroCfeCoreTempValues(), CS\_ZeroEepromTempValues(), CS\_ZeroMemoryTemp←Values(), CS\_ZeroOSTempValues(), CS\_ZeroTablesTempValues(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.4 CS\_DisableCfeCoreCmd()** void CS\_DisableCfeCoreCmd ( const `CS_NoArgsCmd_t` \* `CmdPtr` )

Process a disable background checking for the cFE core code segment command.

#### Description

Disables background checking for the cFE core code segment

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in <code>CS_AppMain</code>
----	---------------------	---

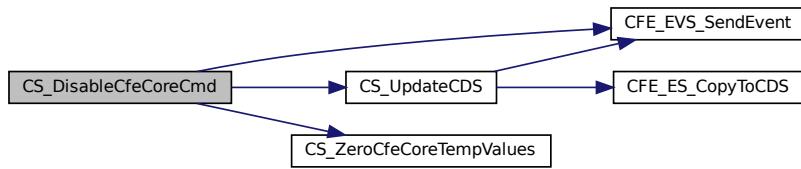
## See also

[CS\\_DISABLE\\_CFE\\_CORE\\_CC](#)

Definition at line 229 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CfeCoreCS←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_DISABLE\_CFECore\_INF\_EID, CS\_STATE\_DIS←ABLED, CS\_UpdateCDS(), CS\_ZeroCfeCoreTempValues(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.15.2.5 CS\_DisableOSCmd()

```
void CS_DisableOSCmd (
```

```
    const CS_NoArgsCmd_t * CmdPtr )
```

Process a disable background checking for the OS code segment command.

#### Description

Disables background checking for the OS code segment

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in CS_AppMain
----	---------------------	--

**See also**

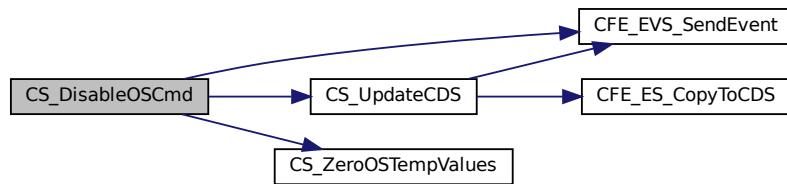
[CS\\_DISABLE\\_OS\\_CC](#)

Definition at line 268 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_DISABLE\_OS\_INF\_EID, CS\_STATE\_DISABLED, CS\_UpdateCDS(), CS\_ZeroOSTempValues(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OSCSSState, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.6 CS\_EnableAllCSCmd()** `void CS_EnableAllCSCmd (`  
 `const CS_NoArgsCmd_t * CmdPtr )`

Process an enable overall background checking command.

**Description**

Allows background checking to take place.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<code>CmdPtr</code>	Command pointer, verified non-null in CS_AppMain
----	---------------------	--

**See also**[CS\\_ENABLE\\_ALL\\_CS\\_CC](#)

Definition at line 215 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::Checksum←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_ENABLE\_ALL\_INF\_EID, CS\_STATE\_ENABLED, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.7 CS\_EnableCfeCoreCmd()** void CS\_EnableCfeCoreCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process an enable background checking for the cFE core code segment command.

**Description**

Allows the cFE Core code segment to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

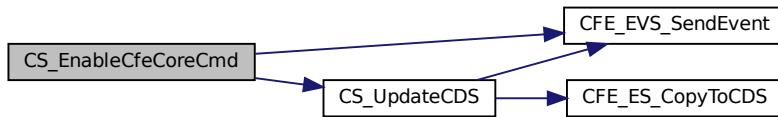
**See also**[CS\\_ENABLE\\_CFE\\_CORE\\_CC](#)

Definition at line 249 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CfeCoreCS←State, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_ENABLE\_CFECore\_INF\_EID, CS\_STATE\_ENA←BLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.8 CS\_EnableOSCmd()** `void CS_EnableOSCmd (`  
    `const CS_NoArgsCmd_t * CmdPtr )`

Process an enable background checking for the OS code segment command.

**Description**

Allows the OS code segment to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

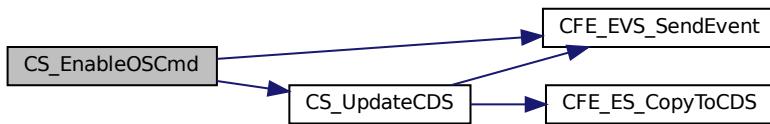
**See also**[CS\\_ENABLE\\_OS\\_CC](#)

Definition at line 288 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_ENABLE\_OS\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OSCSState, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:

**12.15.2.9 CS\_NoopCmd()** `void CS_NoopCmd (`  
`const CS_NoArgsCmd_t * CmdPtr )`

Process noop command.

**Description**

Processes a noop ground command.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

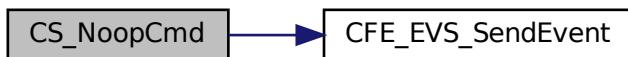
**See also**[CS\\_NOOP\\_CC](#)

Definition at line 49 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_MAJOR\_VERSION, CS\_MINOR\_VERSION, CS\_MISSION\_REV, CS\_NOOP\_INF\_EID, CS\_REV←ISION, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.10 CS\_OneShotCmd()** void CS\_OneShotCmd (   
     const CS\_OneShotCmd\_t \* CmdPtr )

Process a start a one shot checksum command.

**Description**

Starts a one shot checksum on given address and size, and reports checksum in telemetry and an event message.

**Assumptions, External Events, and Notes:****Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

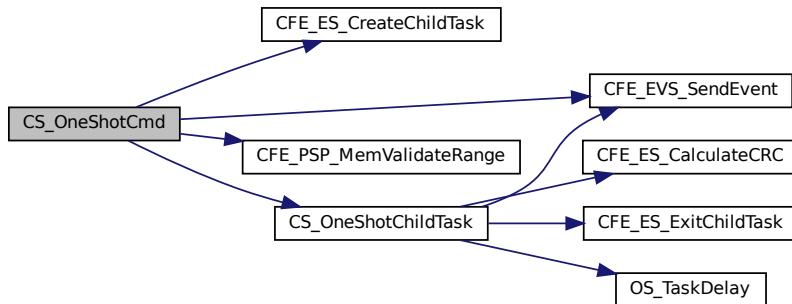
**See also**[CS\\_ONE\\_SHOT\\_CC](#)

Definition at line 445 of file cs\_cmds.c.

References CS\_OneShotCmd\_Payload\_t::Address, CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_PSP\_MEM\_ANY, CFE\_PSP\_MemValidateRange(), CFE\_SUCCESS, CS\_AppData\_t::ChildTaskID, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_ONESHOT\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID, CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID, CS\_ONESHOT\_STARTED\_DBG\_EID, CS\_ONESHOT\_TASK\_NAME, CS\_OneShotChildTask(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::LastOneShotAddress, CS\_HkPacket\_Payload\_t::LastOneShotChecksum, CS\_HkPacket\_Payload\_t::LastOneShotMaxBytesPerCycle, CS\_HkPacket\_Payload\_t::LastOneShotSize, CS\_AppData\_t::MaxBytesPerCycle, CS\_OneShotCmd\_Payload\_t::MaxBytesPerCycle, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_OneShotCmd\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, and CS\_OneShotCmd\_Payload\_t::Size.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.15.2.11 CS\_RecomputeBaselineCfeCoreCmd()

```
void CS_RecomputeBaselineCfeCoreCmd (
```

```
    const CS_NoArgsCmd_t * CmdPtr )
```

Process a recompute baseline of the cFE core code segment command.

#### Description

Recomputes the checksum of the cFE core code segment and use that value as the new baseline for the cFE core.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

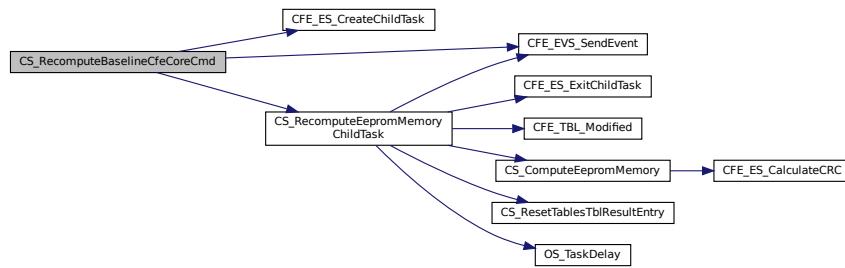
#### See also

[CS\\_RECOMPUTE\\_BASELINE\\_CFE\\_CORE\\_CC](#)

Definition at line 349 of file cs\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_ES\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::CfeCoreCodeSeg, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CFECORE, CS\_CHILD\_TASK\_PRIORITY, CS\_RECOMPUTE\_CFECORE\_TASK\_NAME, CS\_RECOMPUTE\_CFECORE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_CFECORE\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_CFECORE\_STARTED\_DBG\_EID, CS\_RecomputeEepromMemoryChildTask(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, and CS\_HkPacket\_Payload\_t::RecomputeInProgress. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.12 CS\_RecomputeBaselineOSCmd()** `void CS_RecomputeBaselineOSCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process a recompute baseline of the OS command.

#### Description

Recomputes the checksum of the OS code segment and use that value as the new baseline for the OS.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

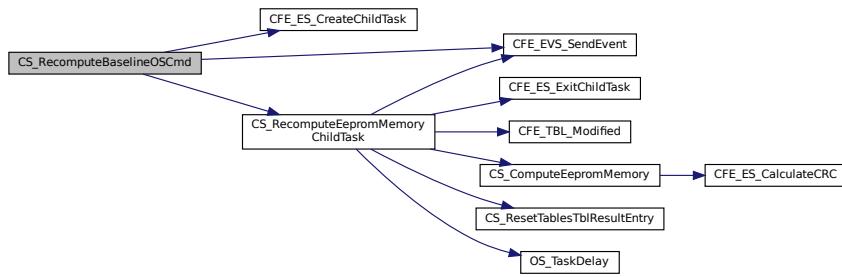
[CS\\_RECOMPUTE\\_BASELINE\\_OS\\_CC](#)

Definition at line 398 of file cs\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_ES\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_OSCORE, CS\_RECOMP\_OS\_TASK\_NAME, CS\_RECOMPUTE\_OS\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_OS\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_OS\_STARTED\_DBG\_EID, CS\_RecomputeEepromMemoryChildTask(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_AppData\_t::OSCodeSeg, CS\_HkPacket\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, and CS\_HkPacket\_Payload\_t::RecomputeInProgress.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.13 CS\_ReportBaselineCfeCoreCmd()** `void CS_ReportBaselineCfeCoreCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process a report baseline of the cFE Core command.

#### Description

Reports the baseline checksum of the cFE core code segment if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

[CS\\_REPORT\\_BASELINE\\_CFE\\_CORE\\_CC](#)

Definition at line 307 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_AppData\_t::CfeCoreCodeSeg, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_CFECORE\_INF\_EID, CS\_NO\_BASELINE\_CFECORE\_INF\_EID, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.14 CS\_ReportBaselineOSCmd()** void CS\_ReportBaselineOSCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process a report baseline of the OS command.

#### Description

Reports the baseline checksum of the OS code segment if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

#### See also

[CS\\_REPORT\\_BASELINE\\_OS\\_CC](#)

Definition at line 328 of file cs\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_OS\_INF\_EID, CS\_NO\_BASELINE\_OS\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_AppData\_t::OSCodeSeg, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.15.2.15 CS\_ResetCmd()** void CS\_ResetCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process reset counters command.

#### Description

Processes a reset counters ground command which will reset the checksum command error and command execution counters to zero. It also resets all checksum error counters and the passes completed counter.

#### Assumptions, External Events, and Notes:

None

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

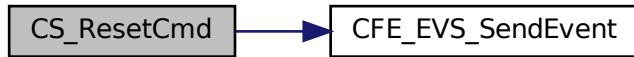
### See also

[CS\\_RESET\\_CC](#)

Definition at line 62 of file cs\_cmds.c.

References `CS_HkPacket_Payload_t::AppCSErrCounter`, `CFE_EVS_EventType_DEBUG`, `CFE_EVS_SendEvent()`, `CS_HkPacket_Payload_t::CfeCoreCSErrCounter`, `CS_HkPacket_Payload_t::CmdCounter`, `CS_HkPacket_Payload_t::CmdErrCounter`, `CS_AppData`, `CS_RESET_DBG_EID`, `CS_HkPacket_Payload_t::EepromCSErrCounter`, `CS_AppData_t::HkPacket`, `CS_HkPacket_Payload_t::MemoryCSErrCounter`, `CS_HkPacket_Payload_t::OSCSErrCounter`, `CS_HkPacket_Payload_t::PassCounter`, `CS_HkPacket_t::Payload`, and `CS_HkPacket_Payload_t::TablesCSErrCounter`. Referenced by `CS_ProcessCmd()`.

Here is the call graph for this function:



## 12.16 apps/cs/fsw/src/cs\_compute.c File Reference

```
#include "cfe.h"
#include "cs_app.h"
#include <string.h>
#include "cs_events.h"
#include "cs_compute.h"
#include "cs_utils.h"
```

### Functions

- `CFE_Status_t CS_ComputeEepromMemory (CS_Res_EepromMemory_Table_Entry_t *ResultsEntry, uint32 *ComputedCSValue, bool *DoneWithEntry)`

*Computes checksums on EEPROM or Memory types.*
- `CFE_Status_t CS_ComputeTables (CS_Res_Tables_Table_Entry_t *ResultsEntry, uint32 *ComputedCSValue, bool *DoneWithEntry)`

*Computes checksums on tables.*
- `CFE_Status_t CS_ComputeApp (CS_Res_App_Table_Entry_t *ResultsEntry, uint32 *ComputedCSValue, bool *DoneWithEntry)`

*Computes checksums on applications.*
- `void CS_RecomputeEepromMemoryChildTask (void)`

*Child task main function for recomputing baselines for EEPROM and Memory types.*
- `void CS_RecomputeAppChildTask (void)`

*Child task main function for recomputing baselines for Applications.*

- void [CS\\_RecomputeTablesChildTask](#) (void)  
*Child task main function for recomputing baselines for Tables.*
- void [CS\\_OneShotChildTask](#) (void)  
*Child task main function for computing a one shot calculation.*

### 12.16.1 Detailed Description

The CFS Checksum (CS) Application's computing checksum functions

### 12.16.2 Function Documentation

**12.16.2.1 CS\_ComputeApp()** `CFE_Status_t CS_ComputeApp (`  
    `CS_Res_App_Table_Entry_t * ResultsEntry,`  
    `uint32 * ComputedCSValue,`  
    `bool * DoneWithEntry )`

Computes checksums on applications.

#### Description

Computes checksums up to MaxBytesPerCycle bytes every call. This function is used to compute checksums for applications.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>ResultsEntry</i>	A pointer to the entry in a table that we want to compute the checksum on. Verified non-null in calling function.
out	<i>ComputedCSValue</i>	Value used to determine the computed checksum, if completed
out	<i>DoneWithEntry</i>	Value that specifies whether or not the specified entry's checksum was completed during this call.

#### Returns

Execution status

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CS_ERROR</a>	Error code returned when a checksum compare failed.
<a href="#">CS_ERR_NOT_FOUND</a>	Error code returned the app or table requested could not be found.

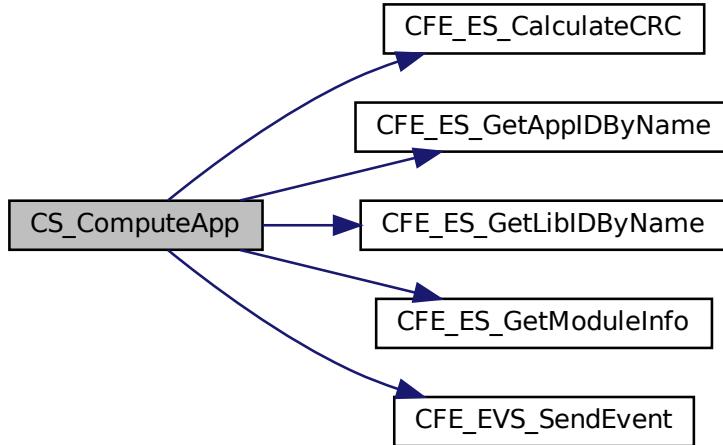
Definition at line 307 of file cs\_compute.c.

References `CFE_ES_AppInfo::AddressesAreValid`, `CS_Res_App_Table_Entry_t::ByteOffset`, `CFE_ES_CalculateCR`←  
`C()`, `CFE_ES_ERR_NAME_NOT_FOUND`, `CFE_ES_GetAppIDByName()`, `CFE_ES_GetLibIDByName()`, `CFE_ES`←  
`_GetModuleInfo()`, `CFE_EVS_EventType_DEBUG`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE`←  
`RESOURCEID_UNDEFINED`, `CFE_SUCCESS`, `CFE_ES_AppInfo::CodeAddress`, `CFE_ES_AppInfo::CodeSize`, `CS`←  
`_Res_App_Table_Entry_t::ComparisonValue`, `CS_Res_App_Table_Entry_t::ComputedYet`, `CS_AppData`, `CS_COM`←

PUT\_APP\_ERR\_EID, CS\_COMPUTE\_APP\_PLATFORM\_DBG\_EID, CS\_DEFAULT\_ALGORITHM, CS\_ERR\_NO\_T\_FOUND, CS\_ERROR, CS\_AppData\_t::MaxBytesPerCycle, CS\_Res\_App\_Table\_Entry\_t::Name, CS\_Res\_App\_Table\_Entry\_t::NumBytesToChecksum, CS\_Res\_App\_Table\_Entry\_t::StartAddress, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_BackgroundApp(), and CS\_RecomputeAppChildTask().

Here is the call graph for this function:



**12.16.2.2 CS\_ComputeEepromMemory()** `CFE_Status_t CS_ComputeEepromMemory (`  
 `CS_Res_EepromMemory_Table_Entry_t * ResultsEntry,`  
 `uint32 * ComputedCSValue,`  
 `bool * DoneWithEntry )`

Computes checksums on EEPROM or Memory types.

#### Description

Computes checksums up to MaxBytesPerCycle bytes every call. This function is used to compute checksums for EEPROM, Memory, the OS code segment and the cFE core code segment

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>ResultsEntry</code>	A pointer to the entry in a table that we want to compute the checksum on. Verified non-null by calling function.
out	<code>ComputedCSValue</code>	Value used to determine the computed checksum, if completed
out	<code>DoneWithEntry</code>	Value that specifies whether or not the specified entry's checksum was completed during this call.

**Returns**

Execution status

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CS_ERROR</a>	Error code returned when a checksum compare failed.

Definition at line 48 of file cs\_compute.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_ES\_CalculateCRC(), CFE\_SUCCESS, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_DEFAULT\_ALGORITHM, CS\_ERROR, CS\_AppData\_t::MaxBytesPerCycle, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_BackgroundCfeCore(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), and CS\_RecomputeEepromMemoryChildTask().

Here is the call graph for this function:



**12.16.2.3 CS\_ComputeTables()** [CFE\\_Status\\_t](#) CS\_ComputeTables (   
*CS\_Res\_Tables\_Table\_Entry\_t* \* *ResultsEntry*,   
*uint32* \* *ComputedCSValue*,   
*bool* \* *DoneWithEntry* )

Computes checksums on tables.

**Description**

Computes checksums up to MaxBytesPerCycle bytes every call. This function is used to compute checksums for tables.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>ResultsEntry</i>	A pointer to the entry in a table that we want to compute the checksum on. Verified non-null in calling function.
out	<i>ComputedCSValue</i>	Value used to determine the computed checksum, if completed
out	<i>DoneWithEntry</i>	Value that specifies whether or not the specified entry's checksum was completed during this call.

## Returns

Execution status

## Return values

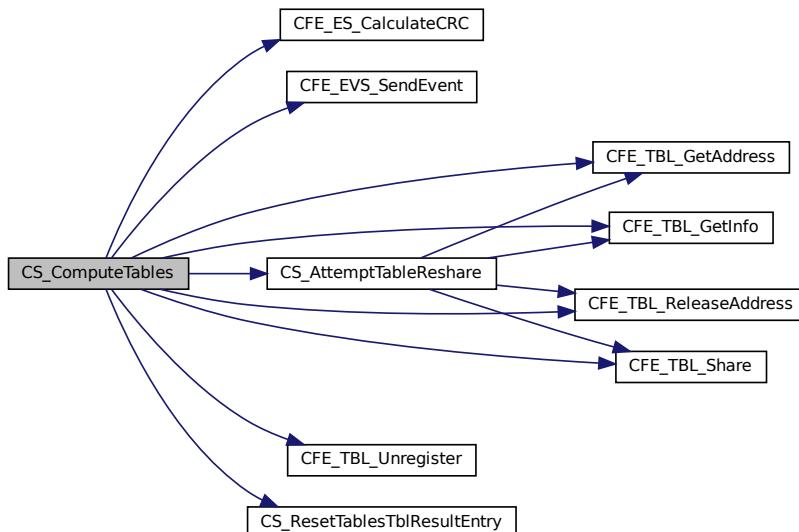
<code>CFE_SUCCESS</code>	Successful execution.
<code>CS_ERROR</code>	Error code returned when a checksum compare failed.
<code>CS_ERR_NOT_FOUND</code>	Error code returned the app or table requested could not be found.

Definition at line 119 of file cs\_compute.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CFE\_ES\_CalculateCRC(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_ERR\_NEVER\_LOADED, CFE\_TBL\_ERR\_UNREGISTERED, CFE\_TBL\_GetAddress(), CFE\_TBL\_GetInfo(), CFE\_TBL\_INFO\_UPDATED, CFE\_TBL\_ReleaseAddress(), CFE\_TBL\_Share(), CFE\_TBL\_Unregister(), CS\_Res\_Tables\_Table\_Entry\_t::ComparisonValue, CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_AttemptTableReshare(), CS\_COMPUTE\_TABLES\_ERR\_EID, CS\_COMPUTE\_TABLES\_RELEASE\_ERR\_EID, CS\_DEFAULT\_ALGORITHM, CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_ResetTablesTblResultEntry(), CS\_AppData\_t::MaxBytesPerCycle, CS\_Res\_Tables\_Table\_Entry\_t::Name, CS\_Res\_Tables\_Table\_Entry\_t::NumBytesToChecksum, CFE\_TBL\_Info::Size, CS\_Res\_Tables\_Table\_Entry\_t::StartAddress, CS\_Res\_Tables\_Table\_Entry\_t::TblHandle, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_BackgroundTables(), and CS\_RecomputeTablesChildTask().

Here is the call graph for this function:



#### 12.16.2.4 CS\_OneShotChildTask()

```
void CS_OneShotChildTask (
    void )
```

Child task main function for computing a one shot calculation.

### Description

Child task main function that is spawned when a one shot command is received.

### Assumptions, External Events, and Notes:

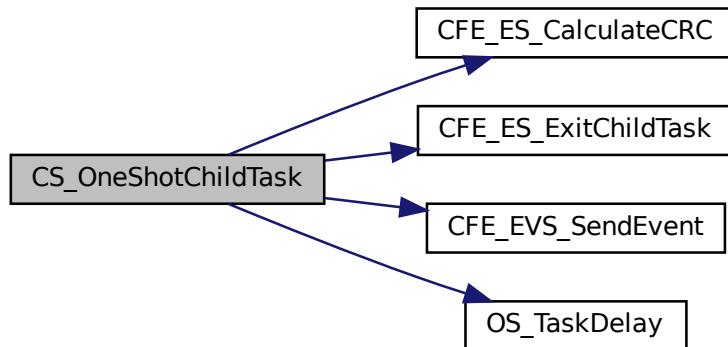
Only one child task for CS can be running at any one time.

Definition at line 782 of file cs\_compute.c.

References CFE\_ES\_CalculateCRC(), CFE\_ES\_ExitChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_AppData\_t::ChildTaskID, CS\_AppData, CS\_CHILD\_TASK\_DELETE, CS\_DEFAULT\_ALGORITHM, CS\_ONESHOT\_FINISHED\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::LastOneShotAddress, CS\_HkPacket\_Payload\_t::LastOneShotChecksum, CS\_HkPacket\_Payload\_t::LastOneShotMaxBytesPerCycle, CS\_HkPacket\_Payload\_t::LastOneShotSize, CS\_HkPacket\_Payload\_t::OneShotInProgress, OS\_TaskDelay(), and CS\_HkPacket\_t::Payload.

Referenced by CS\_OneShotCmd().

Here is the call graph for this function:



### 12.16.2.5 CS\_RecomputeAppChildTask()

```
void CS_RecomputeAppChildTask (
    void )
```

Child task main function for recomputing baselines for Applications.

### Description

Child task main function that is spawned when a recompute baseline command is received for Applications.

### Assumptions, External Events, and Notes:

Only one child task for CS can be running at any one time.

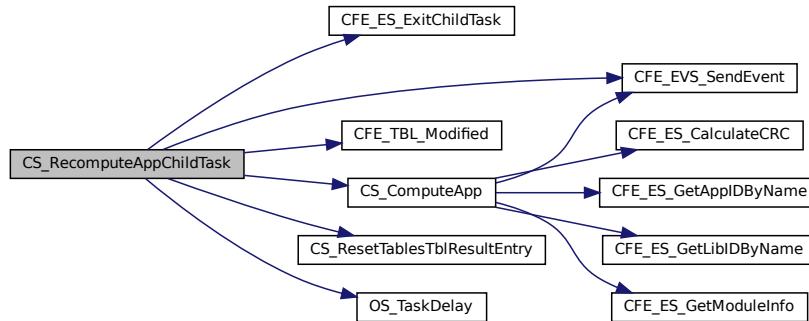
Definition at line 570 of file cs\_compute.c.

References CS\_AppData\_t::AppResTablesTblIPtr, CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CFE\_ES\_ExitChildTask(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_Modified(), CS\_Res\_App\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CHILD\_TASK\_DELETE,

SK\_DELAY, CS\_ComputeApp(), CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_S\_RECOMPUTE\_ERROR\_APP\_ERR\_EID, CS\_S\_RECOMPUTE\_FINISH\_APP\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::DefAppTblPtr, CS\_AppData\_t::HkPacket, CS\_Def\_App\_Table\_Entry\_t::Name, CS\_Res\_App\_Table\_Entry\_t::Name, OS\_MAX\_API\_NAME, OS\_TaskDelay(), CS\_HkPacket\_t::Payload, CS\_AppData\_t::RecomputeAppEntryPtr, CS\_HkPacket\_t::Payload\_t::RecomputeInProgress, CS\_Def\_App\_Table\_Entry\_t::State, CS\_Res\_App\_Table\_Entry\_t::State, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_RecomputeBaselineAppCmd().

Here is the call graph for this function:



### 12.16.2.6 CS\_RecomputeEepromMemoryChildTask()

```
void CS_RecomputeEepromMemoryChildTask (
    void )
```

Child task main function for recomputing baselines for EEPROM and Memory types.

#### Description

Child task main function that is spawned when a recompute baseline command is received for EEPROM, Memory, OS code segment or cFE core code segment.

#### Assumptions, External Events, and Notes:

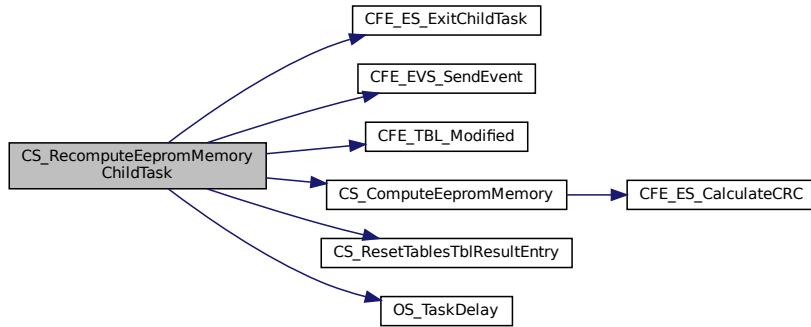
Only one child task for CS can be running at any one time.

Definition at line 440 of file cs\_compute.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_ES\_ExitChildTask(), CFE\_ES\_EventType<=INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_Modified(), CS\_HkPacket<=Payload\_t::CfeCoreBaseline, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CFECORE, CS\_CHILD\_TASK\_DELAY, CS\_ComputeEepromMemory(), CS\_EEPROM\_TABLE, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_MAX\_NUM\_MemoryRY\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_OSCORE, CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_TABLETYPE\_NAM\_E\_SIZE, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_AppData\_t::EepResTablesTblPtr, CS\_AppData\_t::HkPacket, CS\_AppData\_t::MemResTablesTblPtr, OS\_TaskDelay(), CS\_HkPacket\_Payload\_t::OSBaseline, CS\_HkPacket\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), and CS\_RecomputeBaselineOSCmd().

Here is the call graph for this function:



**12.16.2.7 CS\_RecomputeTablesChildTask()** void CS\_RecomputeTablesChildTask ( void )

Child task main function for recomputing baselines for Tables.

#### Description

Child task main function that is spawned when a recompute baseline command is received for a table.

#### Assumptions, External Events, and Notes:

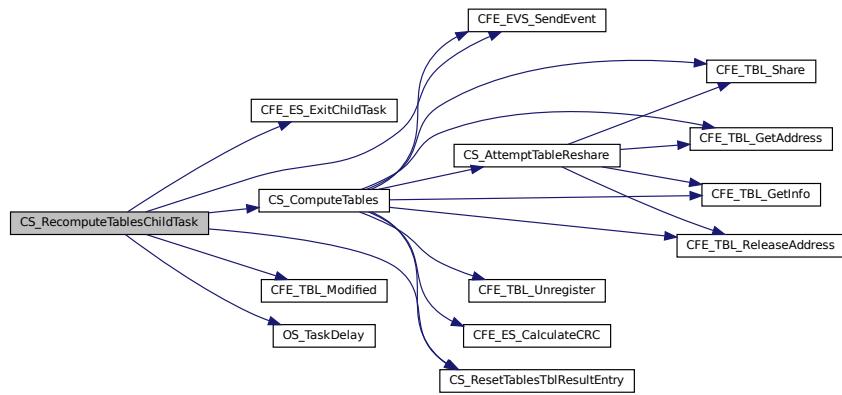
Only one child task for CS can be running at any one time.

Definition at line 676 of file cs\_compute.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CFE\_ES\_ExitChildTask(), CFE\_ES\_EventType\_ERROR, CFE\_ES\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CFE\_TBL\_Modified(), CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CHILD\_TASK\_DELAY, CS\_ComputeTables(), CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID, CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::DefTablesTblPtr, CS\_AppData\_t::HkPacket, CS\_Def\_Tables\_Table\_Entry\_t::Name, CS\_Res\_Tables\_Table\_Entry\_t::Name, OS\_TaskDelay(), CS\_HkPacket\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::RecomputeTablesEntryPtr, CS\_Def\_Tables\_Table\_Entry\_t::State, CS\_Res\_Tables\_Table\_Entry\_t::State, CS\_AppData\_t::TblResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_RecomputeBaselineTablesCmd().

Here is the call graph for this function:



## 12.17 apps/cs/fsw/src/cs\_compute.h File Reference

```
#include "cfe.h"
#include "cs_tbldefs.h"
```

### Functions

- **CFE\_Status\_t CS\_ComputeEepromMemory** (CS\_Res\_EepromMemory\_Table\_Entry\_t \*ResultsEntry, uint32 \*ComputedCSValue, bool \*DoneWithEntry)
 

*Computes checksums on EEPROM or Memory types.*
- **CFE\_Status\_t CS\_ComputeTables** (CS\_Res\_Tables\_Table\_Entry\_t \*ResultsEntry, uint32 \*ComputedCSValue, bool \*DoneWithEntry)
 

*Computes checksums on tables.*
- **CFE\_Status\_t CS\_ComputeApp** (CS\_Res\_App\_Table\_Entry\_t \*ResultsEntry, uint32 \*ComputedCSValue, bool \*DoneWithEntry)
 

*Computes checksums on applications.*
- void **CS\_RecomputeEepromMemoryChildTask** (void)
 

*Child task main function for recomputing baselines for EEPROM and Memory types.*
- void **CS\_RecomputeTablesChildTask** (void)
 

*Child task main function for recomputing baselines for Tables.*
- void **CS\_RecomputeAppChildTask** (void)
 

*Child task main function for recomputing baselines for Applications.*
- void **CS\_OneShotChildTask** (void)
 

*Child task main function for computing a one shot calculation.*

### 12.17.1 Detailed Description

Specification for the CFS computation functions.

### 12.17.2 Function Documentation

```
12.17.2.1 CS_ComputeApp() CFE_Status_t CS_ComputeApp (
    CS_Res_App_Table_Entry_t * ResultsEntry,
    uint32 * ComputedCSValue,
    bool * DoneWithEntry )
```

Computes checksums on applications.

#### Description

Computes checksums up to MaxBytesPerCycle bytes every call. This function is used to compute checksums for applications.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>ResultsEntry</i>	A pointer to the entry in a table that we want to compute the checksum on. Verified non-null in calling function.
out	<i>ComputedCSValue</i>	Value used to determine the computed checksum, if completed
out	<i>DoneWithEntry</i>	Value that specifies whether or not the specified entry's checksum was completed during this call.

#### Returns

Execution status

#### Return values

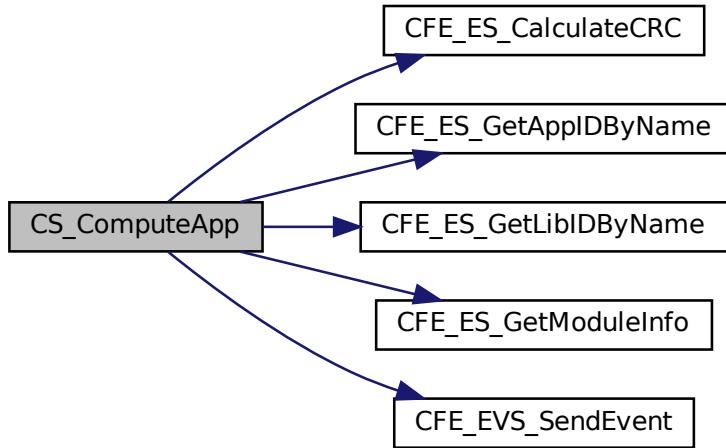
<i>CFE_SUCCESS</i>	Successful execution.
<i>CS_ERROR</i>	Error code returned when a checksum compare failed.
<i>CS_ERR_NOT_FOUND</i>	Error code returned the app or table requested could not be found.

Definition at line 307 of file cs\_compute.c.

References CFE\_ES\_AppInfo::AddressesAreValid, CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CFE\_ES\_CalculateCR←C(), CFE\_ES\_ERR\_NAME\_NOT\_FOUND, CFE\_ES\_GetAppIDByName(), CFE\_ES\_GetLibIDByName(), CFE\_ES←\_GetModuleInfo(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE←RESOURCEID\_UNDEFINED, CFE\_SUCCESS, CFE\_ES\_AppInfo::CodeAddress, CFE\_ES\_AppInfo::CodeSize, CS←\_Res\_App\_Table\_Entry\_t::ComparisonValue, CS\_Res\_App\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_COM←PUTE\_APP\_ERR\_EID, CS\_COMPUTE\_APP\_PLATFORM\_DBG\_EID, CS\_DEFAULT\_ALGORITHM, CS\_ERR\_NO←T\_FOUND, CS\_ERROR, CS\_AppData\_t::MaxBytesPerCycle, CS\_Res\_App\_Table\_Entry\_t::Name, CS\_Res\_App←Table\_Entry\_t::NumBytesToChecksum, CS\_Res\_App\_Table\_Entry\_t::StartAddress, and CS\_Res\_App\_Table\_Entry←\_t::TempChecksumValue.

Referenced by CS\_BackgroundApp(), and CS\_RecomputeAppChildTask().

Here is the call graph for this function:



**12.17.2.2 CS\_ComputeEepromMemory()** `CFE_Status_t CS_ComputeEepromMemory (`  
 `CS_Res_EepromMemory_Table_Entry_t * ResultsEntry,`  
 `uint32 * ComputedCSValue,`  
 `bool * DoneWithEntry )`

Computes checksums on EEPROM or Memory types.

#### Description

Computes checksums up to MaxBytesPerCycle bytes every call. This function is used to compute checksums for EEPROM, Memory, the OS code segment and the cFE core code segment

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>ResultsEntry</code>	A pointer to the entry in a table that we want to compute the checksum on. Verified non-null by calling function.
out	<code>ComputedCSValue</code>	Value used to determine the computed checksum, if completed
out	<code>DoneWithEntry</code>	Value that specifies whether or not the specified entry's checksum was completed during this call.

#### Returns

Execution status

**Return values**

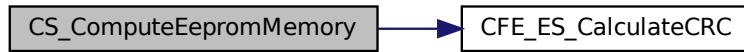
<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CS_ERROR</a>	Error code returned when a checksum compare failed.

Definition at line 48 of file cs\_compute.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_ES\_CalculateCRC(), CFE\_SUCCESS, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_DEFAULT\_ALGORITHM, CS\_ERROR, CS\_AppData\_t::MaxBytesPerCycle, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_BackgroundCfeCore(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), and CS\_RecomputeEepromMemoryChildTask().

Here is the call graph for this function:



**12.17.2.3 CS\_ComputeTables()** [CFE\\_Status\\_t](#) CS\_ComputeTables (

- [CS\\_Res\\_Tables\\_Table\\_Entry\\_t](#) \* *ResultsEntry*,
- [uint32](#) \* *ComputedCSValue*,
- [bool](#) \* *DoneWithEntry* )

Computes checksums on tables.

**Description**

Computes checksums up to MaxBytesPerCycle bytes every call. This function is used to compute checksums for tables.

**Assumptions, External Events, and Notes:**

None

**Parameters**

<a href="#">in</a>	<i>ResultsEntry</i>	A pointer to the entry in a table that we want to compute the checksum on. Verified non-null in calling function.
<a href="#">out</a>	<i>ComputedCSValue</i>	Value used to determine the computed checksum, if completed
<a href="#">out</a>	<i>DoneWithEntry</i>	Value that specifies whether or not the specified entry's checksum was completed during this call.

## Returns

Execution status

## Return values

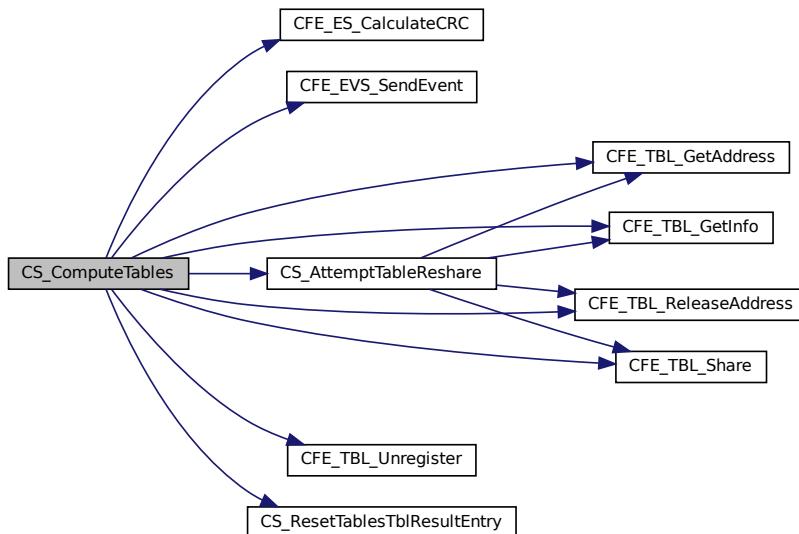
<code>CFE_SUCCESS</code>	Successful execution.
<code>CS_ERROR</code>	Error code returned when a checksum compare failed.
<code>CS_ERR_NOT_FOUND</code>	Error code returned the app or table requested could not be found.

Definition at line 119 of file cs\_compute.c.

References `CS_Res_Tables_Table_Entry_t::ByteOffset`, `CFE_ES_CalculateCRC()`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CFE_TBL_BAD_TABLE_HANDLE`, `CFE_TBL_ERR_NEVER_LOADED`, `CFE_TBL_ERR_UNREGISTERED`, `CFE_TBL_GetAddress()`, `CFE_TBL_GetInfo()`, `CFE_TBL_INFO_UPDATED`, `CFE_TBL_ReleaseAddress()`, `CFE_TBL_Share()`, `CFE_TBL_Unregister()`, `CS_Res_Tables_Table_Entry_t::ComparisonValue`, `CS_Res_Tables_Table_Entry_t::ComputedYet`, `CS_AppData`, `CS_AttemptTableReshare()`, `CS_COMPUTE_TABLES_ERR_EID`, `CS_COMPUTE_TABLES_RELEASE_ERR_EID`, `CS_DEFAULT_ALGORITHM`, `CS_ERR_NOT_FOUND`, `CS_ERROR`, `CS_ResetTablesTblResultEntry()`, `CS_AppData_t::MaxBytesPerCycle`, `CS_Res_Tables_Table_Entry_t::Name`, `CS_Res_Tables_Table_Entry_t::NumBytesToChecksum`, `CFE_TBL_Info::Size`, `CS_Res_Tables_Table_Entry_t::StartAddress`, `CS_Res_Tables_Table_Entry_t::TblHandle`, and `CS_Res_Tables_Table_Entry_t::TempChecksumValue`.

Referenced by `CS_BackgroundTables()`, and `CS_RecomputeTablesChildTask()`.

Here is the call graph for this function:



#### 12.17.2.4 CS\_OneShotChildTask()

```
void CS_OneShotChildTask (
    void )
```

Child task main function for computing a one shot calculation.

### Description

Child task main function that is spawned when a one shot command is received.

### Assumptions, External Events, and Notes:

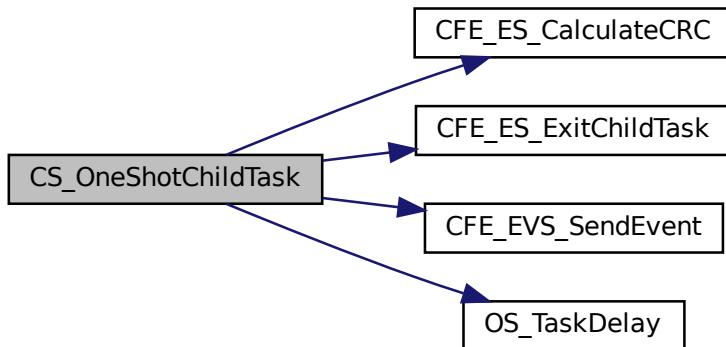
Only one child task for CS can be running at any one time.

Definition at line 782 of file cs\_compute.c.

References CFE\_ES\_CalculateCRC(), CFE\_ES\_ExitChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_AppData\_t::ChildTaskID, CS\_AppData, CS\_CHILD\_TASK\_DELETE, CS\_DEFAULT\_ALGORITHM, CS\_ONESHOT\_FINISHED\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::LastOneShotAddress, CS\_HkPacket\_Payload\_t::LastOneShotChecksum, CS\_HkPacket\_Payload\_t::LastOneShotMaxBytesPerCycle, CS\_HkPacket\_Payload\_t::LastOneShotSize, CS\_HkPacket\_Payload\_t::OneShotInProgress, OS\_TaskDelay(), and CS\_HkPacket\_t::Payload.

Referenced by CS\_OneShotCmd().

Here is the call graph for this function:



### 12.17.2.5 CS\_RecomputeAppChildTask()

```
void CS_RecomputeAppChildTask (
    void )
```

Child task main function for recomputing baselines for Applications.

### Description

Child task main function that is spawned when a recompute baseline command is received for Applications.

### Assumptions, External Events, and Notes:

Only one child task for CS can be running at any one time.

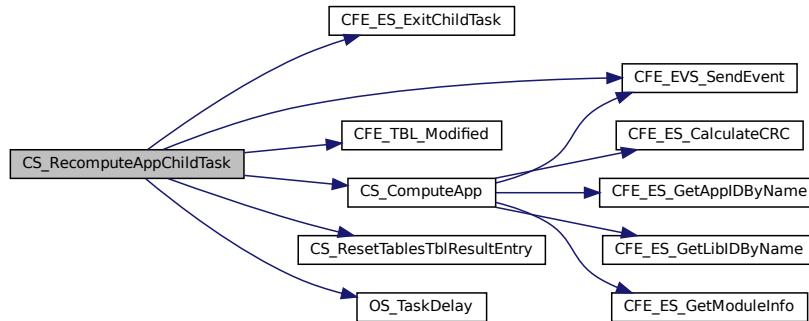
Definition at line 570 of file cs\_compute.c.

References CS\_AppData\_t::AppResTablesTblIPtr, CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CFE\_ES\_ExitChildTask(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_Modified(), CS\_Res\_App\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CHILD\_TASK\_DELETE,

SK\_DELAY, CS\_ComputeApp(), CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_Recompute\_Error\_App\_Err\_EID, CS\_Recompute\_Finish\_App\_Inf\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::DefAppTblPtr, CS\_AppData\_t::HkPacket, CS\_Def\_App\_Table\_Entry\_t::Name, CS\_Res\_App\_Table\_Entry\_t::Name, OS\_MAX\_APP\_NAME, OS\_TaskDelay(), CS\_HkPacket\_t::Payload, CS\_AppData\_t::RecomputeAppEntryPtr, CS\_HkPacket\_t::Payload\_t::RecomputeInProgress, CS\_Def\_App\_Table\_Entry\_t::State, CS\_Res\_App\_Table\_Entry\_t::State, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_RecomputeBaselineAppCmd().

Here is the call graph for this function:



### 12.17.2.6 CS\_RecomputeEepromMemoryChildTask()

```
void CS_RecomputeEepromMemoryChildTask (
    void )
```

Child task main function for recomputing baselines for EEPROM and Memory types.

#### Description

Child task main function that is spawned when a recompute baseline command is received for EEPROM, Memory, OS code segment or cFE core code segment.

#### Assumptions, External Events, and Notes:

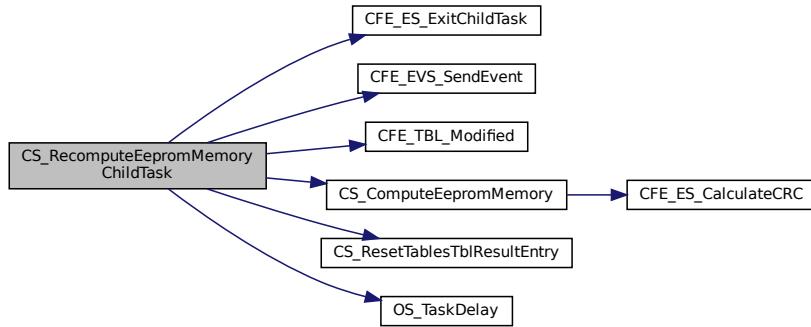
Only one child task for CS can be running at any one time.

Definition at line 440 of file cs\_compute.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_ES\_ExitChildTask(), CFE\_ES\_EventType<=INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_Modified(), CS\_HkPacket<=Payload\_t::CfeCoreBaseline, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CFECore, CS\_CHILD\_TASK\_DELAY, CS\_ComputeEepromMemory(), CS\_EEPROM\_TABLE, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_OSCORE, CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_TABLETYPE\_NAME\_SIZE, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_AppData\_t::EepResTablesTblPtr, CS\_AppData\_t::HkPacket, CS\_AppData\_t::MemResTablesTblPtr, OS\_TaskDelay(), CS\_HkPacket\_Payload\_t::OSBaseline, CS\_HkPacket\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_RecomputeBaselineCfeCoreCmd(), CS\_RecomputeBaselineEepromCmd(), CS\_RecomputeBaselineMemoryCmd(), and CS\_RecomputeBaselineOSCmd().

Here is the call graph for this function:



**12.17.2.7 CS\_RecomputeTablesChildTask()** void CS\_RecomputeTablesChildTask ( void )

Child task main function for recomputing baselines for Tables.

#### Description

Child task main function that is spawned when a recompute baseline command is received for a table.

#### Assumptions, External Events, and Notes:

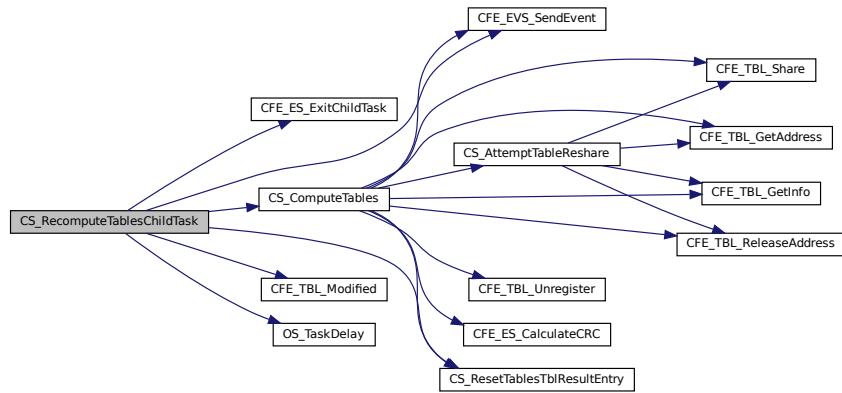
Only one child task for CS can be running at any one time.

Definition at line 676 of file cs\_compute.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CFE\_ES\_ExitChildTask(), CFE\_ES\_EventType\_ERROR, CFE\_ES\_EventType\_INFORMATION, CFE\_ES\_SendEvent(), CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CFE\_TBL\_Modified(), CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CHILD\_TASK\_DELAY, CS\_ComputeTables(), CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID, CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::DefTablesTblPtr, CS\_AppData\_t::HkPacket, CS\_Def\_Tables\_Table\_Entry\_t::Name, CS\_Res\_Tables\_Table\_Entry\_t::Name, OS\_TaskDelay(), CS\_HkPacket\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::RecomputeTablesEntryPtr, CS\_Def\_Tables\_Table\_Entry\_t::State, CS\_Res\_Tables\_Table\_Entry\_t::State, CS\_AppData\_t::TblResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_RecomputeBaselineTablesCmd().

Here is the call graph for this function:



## 12.18 apps/cs/fsw/src/cs\_eeprom\_cmds.c File Reference

```
#include "cfe.h"
#include "cs_tbldefs.h"
#include "cs_app.h"
#include "cs_events.h"
#include "cs_compute.h"
#include "cs_eeprom_cmds.h"
#include "cs_utils.h"
```

### Functions

- void `CS_DisableEepromCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process a disable background checking for the EEPROM table command.*
- void `CS_EnableEepromCmd` (const `CS_NoArgsCmd_t` \*CmdPtr)  
*Process an enable background checking for the EEPROM table command.*
- void `CS_ReportBaselineEntryIDEepromCmd` (const `CS_EntryCmd_t` \*CmdPtr)  
*Process a report baseline of an EEPROM Entry command.*
- void `CS_RecomputeBaselineEepromCmd` (const `CS_EntryCmd_t` \*CmdPtr)  
*Process a recocompute baseline of an EEPROM table entry command.*
- void `CS_EnableEntryIDEepromCmd` (const `CS_EntryCmd_t` \*CmdPtr)  
*Process an enable background checking for an EEPROM entry command.*
- void `CS_DisableEntryIDEepromCmd` (const `CS_EntryCmd_t` \*CmdPtr)  
*Process a disable background checking for an EEPROM entry command.*
- void `CS_GetEntryIDEepromCmd` (const `CS_GetEntryIDCmd_t` \*CmdPtr)  
*Process a get EEPROM Entry by Address command.*

#### 12.18.1 Detailed Description

The CFS Checksum (CS) Application's commands for checking EEPROM

## 12.18.2 Function Documentation

**12.18.2.1 CS\_DisableEepromCmd()** `void CS_DisableEepromCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process a disable background checking for the EEPROM table command.

### Description

Disables background checking for the EEPROM table

### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

### See also

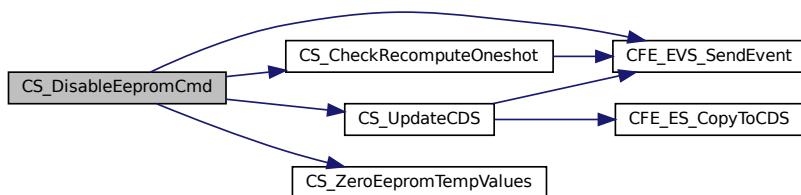
[CS\\_DISABLE\\_EEPROM\\_CC](#)

Definition at line 49 of file cs\_eeprom\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_EEPROM\_INF\_EID, CS\_STATE\_DISABLED, CS\_UpdateCDS(), CS\_ZeroEepromTempValues(), CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.18.2.2 CS\_DisableEntryIDEepromCmd()** `void CS_DisableEntryIDEepromCmd ( const CS_EntryCmd_t * CmdPtr )`

Process a disable background checking for an EEPROM entry command.

### Description

Disables the specified EEPROM entry to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

**See also**

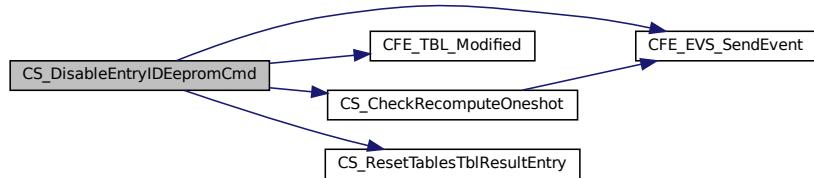
[CS\\_DISABLE\\_ENTRY\\_EEPROM\\_CC](#)

Definition at line 283 of file cs\_eeprom\_cmds.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID, CS\_DISABLE\_EEPROM\_ENTRY\_INF\_EID, CS\_DISABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::EepResTablesTblPtr, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResEepromTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.18.2.3 CS\_EnableEepromCmd()

```
void CS_EnableEepromCmd (
    const CS_NoArgsCmd_t * CmdPtr )
```

Process an enable background checking for the EEPROM table command.

**Description**

Allows the EEPROM table to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

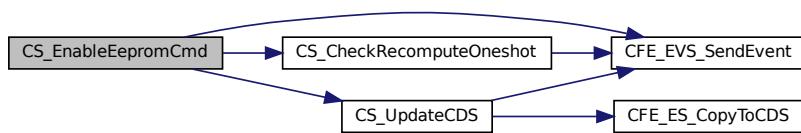
[CS\\_ENABLE\\_EEPROM\\_CC](#)

Definition at line 72 of file cs\_eeprom\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_EEPROM\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.18.2.4 CS\_EnableEntryIDEEPROMCmd()** void CS\_EnableEntryIDEEPROMCmd ( const CS\_EntryCmd\_t \* CmdPtr )

Process an enable background checking for an EEPROM entry command.

## Description

Allows the specified EEPROM entry to be background checksummed.

## Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

[CS\\_ENABLE\\_ENTRY\\_EEPROM\\_CC](#)

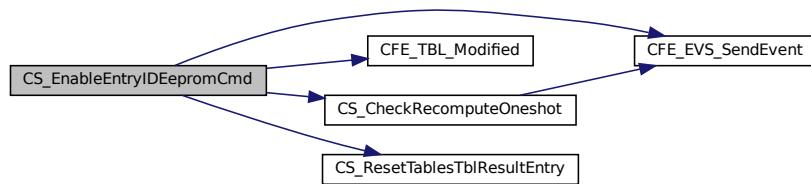
Definition at line 223 of file cs\_eeprom\_cmds.c.

References CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::

CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID, CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID, CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_ResetTablesTblResultEntry(), CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_STATE\_UNDEFINED, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::EepResTablesTblPtr, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResEepromTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.18.2.5 CS\_GetEntryIDEepromCmd()** void CS\_GetEntryIDEepromCmd ( const CS\_GetEntryIDCmd\_t \* CmdPtr )

Process a get EEPROM Entry by Address command.

#### Description

Send the entry ID of the specified address in an event message

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

#### See also

[CS\\_GET\\_ENTRY\\_ID\\_EEPROM\\_CC](#)

Definition at line 346 of file cs\_eeprom\_cmds.c.

References CS\_GetEntryIDCmd\_Payload\_t::Address, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID, CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::HkPacket, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_HkPacket\_t::Payload, CS\_GetEntryIDCmd\_t::Payload, CS\_AppData\_t::ResEepromTblPtr, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.18.2.6 CS\_RecomputeBaselineEEPROMCmd()** void CS\_RecomputeBaselineEEPROMCmd ( const CS\_EntryCmd\_t \* CmdPtr )

Process a recopmpute baseline of an EEPROM table entry command.

#### Description

Recomputes the checksum of an EEPROM table entry and use that value as the new baseline for that entry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

#### See also

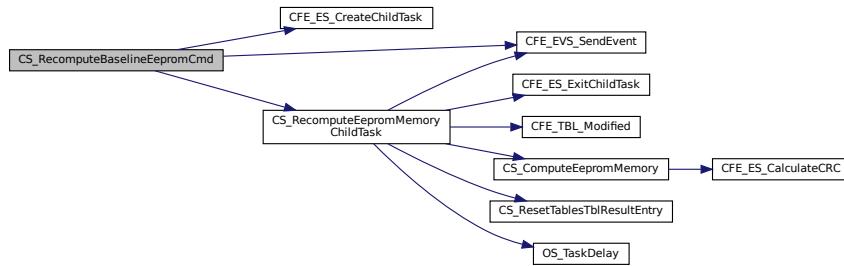
[CS\\_RECOMPUTE\\_BASELINE\\_EEPROM\\_CC](#)

Definition at line 146 of file cs\_eeprom\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_EEPROM\_TABLE, CS\_ERROR, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_RECOMP\_EEPROM\_TASK\_NAME, CS\_RECOMPUTE\_EEPROM\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID, CS\_RECOMPUTE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID, CS\_RecomputeEepromMemoryChildTask(), CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::ResEepromTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.18.2.7 CS\_ReportBaselineEntryIDEEPROMCmd()

```
void CS_ReportBaselineEntryIDEEPROMCmd (
    const CS_EntryCmd_t * CmdPtr )
```

Process a report baseline of an EEPROM Entry command.

#### Description

Reports the baseline checksum of the specified EEPROM table entry if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

[CS\\_REPORT\\_BASELINE\\_EEPROM\\_CC](#)

Definition at line 94 of file cs\_eeprom\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_EEPROM\_INF\_EID, CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_NO\_BASELINE\_EEPROM\_INF\_EID, CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResEepromTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.19 apps/cs/fsw/src/cs\_eeprom\_cmds.h File Reference

```
#include "cfe.h"
#include "cs_msg.h"
```

### Functions

- void [CS\\_DisableEepromCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the EEPROM table command.*
- void [CS\\_EnableEepromCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for the EEPROM table command.*
- void [CS\\_ReportBaselineEntryIDEepromCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a report baseline of an EEPROM Entry command.*
- void [CS\\_DisableEntryIDEepromCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for an EEPROM entry command.*
- void [CS\\_RecomputeBaselineEepromCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a recalculate baseline of an EEPROM table entry command.*
- void [CS\\_EnableEntryIDEepromCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for an EEPROM entry command.*
- void [CS\\_GetEntryIDEepromCmd](#) (const [CS\\_GetEntryIDCmd\\_t](#) \*CmdPtr)  
*Process a get EEPROM Entry by Address command.*

### 12.19.1 Detailed Description

Specification for the CFS eeprom cmd

### 12.19.2 Function Documentation

**12.19.2.1 CS\_DisableEepromCmd()** void [CS\\_DisableEepromCmd](#) (

```
    const CS\_NoArgsCmd\_t * CmdPtr )
```

Process a disable background checking for the EEPROM table command.

#### Description

Disables background checking for the EEPROM table

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

### See also

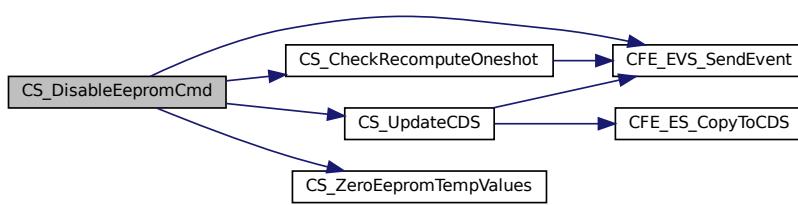
[CS\\_DISABLE\\_EEPROM\\_CC](#)

Definition at line 49 of file `cs_eeprom_cmds.c`.

References `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CS_HkPacket_Payload_t::CmdCounter`, `CS_AppData`, `CS_CheckRecomputeOneshot()`, `CS_DISABLE_EEPROM_INF_EID`, `CS_STATE_DISABLED`, `CS_UpdateCDS()`, `CS_ZeroEepromTempValues()`, `CS_HkPacket_Payload_t::EepromCSSState`, `CS_AppData_t::HkPacket`, and `CS_HkPacket_t::Payload`.

Referenced by `CS_ProcessCmd()`.

Here is the call graph for this function:



**12.19.2.2 CS\_DisableEntryIDEepromCmd()** `void CS_DisableEntryIDEepromCmd (`  
`const CS_EntryCmd_t * CmdPtr )`

Process a disable background checking for an EEPROM entry command.

### Description

Disables the specified EEPROM entry to be background checksummed.

### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

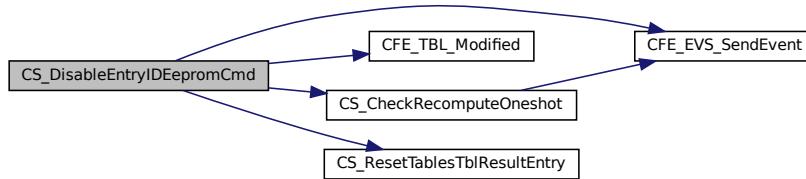
## See also

[CS\\_DISABLE\\_ENTRY\\_EEPROM\\_CC](#)

Definition at line 283 of file `cs_eeprom_cmds.c`.

References `CS_Res_EepromMemory_Table_Entry_t::ByteOffset`, `CFE_EVS_EventType_DEBUG`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_TBL_Modified()`, `CS_HkPacket_Payload_t::CmdCounter`, `CS_HkPacket_Payload_t::CmdErrCounter`, `CS_AppData`, `CS_CheckRecomputeOneshot()`, `CS_DISABLE_EEPROM_DEF_EMPTY_DBG_EID`, `CS_DISABLE_EEPROM_ENTRY_INF_EID`, `CS_DISABLE_EEPROM_INVALID_ENTRY_ERR_EID`, `CS_MAX_NUM_EEPROM_TABLE_ENTRIES`, `CS_ResetTablesTblResultEntry()`, `CS_STATE_DISABLED`, `CS_STATE_EMPTY`, `CS_STATE_UNDEFINED`, `CS_AppData_t::DefEepromTableHandle`, `CS_AppData_t::DefEepromTblPtr`, `CS_AppData_t::EepResTablesTblPtr`, `CS_EntryCmd_Payload_t::EntryID`, `CS_AppData_t::HkPacket`, `CS_HkPacket_t::Payload`, `CS_EntryCmd_t::Payload`, `CS_AppData_t::ResEepromTblPtr`, `CS_Def_EepromMemory_Table_Entry_t::State`, `CS_Res_EepromMemory_Table_Entry_t::State`, and `CS_Res_EepromMemory_Table_Entry_t::TempChecksumValue`.  
Referenced by `CS_ProcessCmd()`.

Here is the call graph for this function:



**12.19.2.3 CS\_EnableEepromCmd()** `void CS_EnableEepromCmd (`  
`const CS_NoArgsCmd_t * CmdPtr )`

Process an enable background checking for the EEPROM table command.

## Description

Allows the EEPROM table to be background checksummed.

## Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

## Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in <code>CS_AppMain</code>
----	---------------------	---

## See also

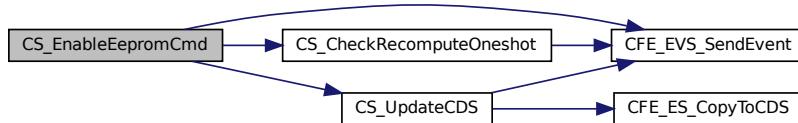
[CS\\_ENABLE\\_EEPROM\\_CC](#)

Definition at line 72 of file `cs_eeprom_cmds.c`.

References `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CS_HkPacket_Payload_t::CmdCounter`,

CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_EEPROM\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



#### 12.19.2.4 CS\_EnableEntryIDEepromCmd()

```
void CS_EnableEntryIDEepromCmd (
    const CS_EntryCmd_t * CmdPtr )
```

Process an enable background checking for an EEPROM entry command.

##### Description

Allows the specified EEPROM entry to be background checksummed.

##### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

##### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

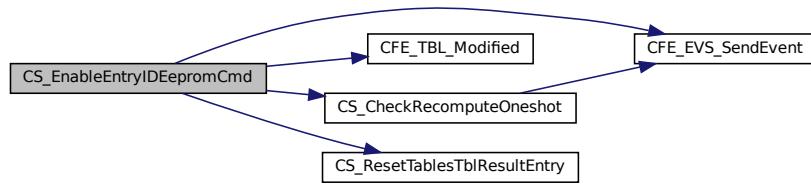
##### See also

[CS\\_ENABLE\\_ENTRY\\_EEPROM\\_CC](#)

Definition at line 223 of file cs\_eeprom\_cmds.c.

References CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID, CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID, CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_ResetTablesTblResultEntry(), CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_STATE\_UNDEFINED, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::EepResTablesTblPtr, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResEepromTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.19.2.5 CS\_GetEntryIDEEPROMCmd()** void CS\_GetEntryIDEEPROMCmd ( const CS\_GetEntryIDCmd\_t \* CmdPtr )

Process a get EEPROM Entry by Address command.

#### Description

Send the entry ID of the specified address in an event message

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

#### See also

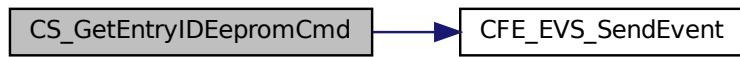
[CS\\_GET\\_ENTRY\\_ID\\_EEPROM\\_CC](#)

Definition at line 346 of file cs\_eeprom\_cmds.c.

References CS\_GetEntryIDCmd\_Payload\_t::Address, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID, CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::HkPacket, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_HkPacket\_t::Payload, CS\_GetEntryIDCmd\_t::Payload, CS\_AppData\_t::ResEepromTblPtr, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



```
12.19.2.6 CS_RecomputeBaselineEepromCmd() void CS_RecomputeBaselineEepromCmd (
    const CS_EntryCmd_t * CmdPtr )
```

Process a recompute baseline of an EEPROM table entry command.

#### Description

Recomputes the checksum of an EEPROM table entry and use that value as the new baseline for that entry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

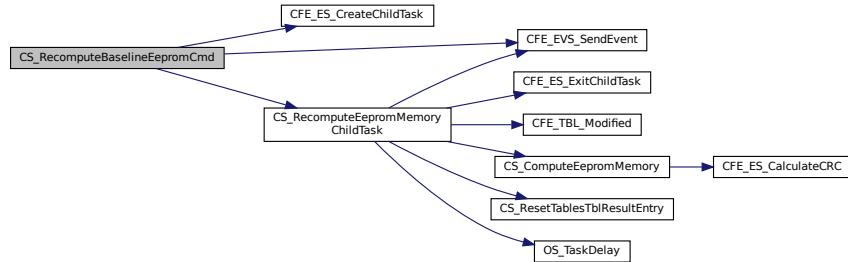
[CS\\_RECOMPUTE\\_BASELINE\\_EEPROM\\_CC](#)

Definition at line 146 of file cs\_eeprom\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_DEBUG, CFE\_ES\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_EEPROM\_TABLE, CS\_ERROR, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_RECOMP\_EEPROM\_TASK\_NAME, CS\_RECOMPUTE\_EEPROM\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID, CS\_RECOMPUTE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID, CS\_RecomputeEepromMemoryChildTask(), CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::ResEepromTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



```
12.19.2.7 CS_ReportBaselineEntryIDEepromCmd() void CS_ReportBaselineEntryIDEepromCmd (
    const CS_EntryCmd_t * CmdPtr )
```

Procces a report baseline of an EEPROM Entry command.

**Description**

Reports the baseline checksum of the specified EEPROM table entry if it has already been computed

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

**See also**

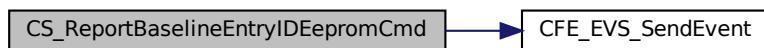
[CS\\_REPORT\\_BASELINE\\_EEPROM\\_CC](#)

Definition at line 94 of file cs\_eeprom\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_EEPROM\_INF\_EID, CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_NO\_BASELINE\_EEPROM\_INF\_EID, CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResEepromTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.20 apps/cs/fsw/src/cs\_init.c File Reference

```
#include <string.h>
#include "cfe.h"
#include "cs_app.h"
#include "cs_platform_cfg.h"
#include "cs_events.h"
#include "cs_utils.h"
#include "cs_compute.h"
#include "cs_eeprom_cmds.h"
#include "cs_table_cmds.h"
#include "cs_memory_cmds.h"
#include "cs_app_cmds.h"
#include "cs_cmds.h"
#include "cs_init.h"
```

**Functions**

- [CFE\\_Status\\_t CS\\_SblInit \(void\)](#)

*Initializes the Software Bus Pipes for the Checksum Application.*

- [CFE\\_Status\\_t CS\\_InitAllTables \(void\)](#)

*Initializes the tables for the Checksum Application.*

- [void CS\\_InitSegments \(void\)](#)

*Initializes the cFE and OS segments for the Checksum Application.*

## 12.20.1 Function Documentation

### 12.20.1.1 [CS\\_InitAllTables\(\)](#) `CFE_Status_t CS_InitAllTables ( void )`

Initializes the tables for the Checksum Application.

#### Description

Initializes all tables used by the Checksum application.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

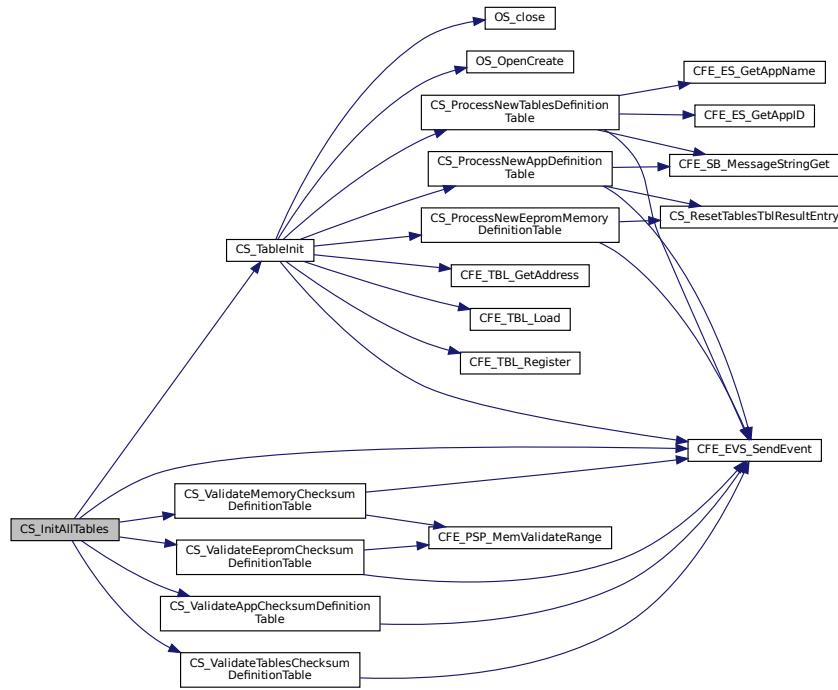
#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 100 of file cs\_init.c.

References `CS_HkPacket_Payload_t::AppCSState`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CS_APP_TABLE`, `CS_AppData`, `CS_DEF_APP_TABLE_FILENAME`, `CS_DEF_APP_TABLE_NAME`, `CS_DEF_EEPROM_TABLE_FILENAME`, `CS_DEF_EEPROM_TABLE_NAME`, `CS_DEF_MEMORY_TABLE_FILENAME`, `CS_DEF_MEMORY_TABLE_NAME`, `CS_DEF_TABLES_TABLE_FILENAME`, `CS_DEF_TABLES_TABLE_NAME`, `CS_EEPROM_TABLE`, `CS_INIT_APP_ERR_EID`, `CS_INIT_EEPROM_ERR_EID`, `CS_INIT_MEMORY_ERR_EID`, `CS_INIT_TABLES_ERR_EID`, `CS_MAX_NUM_APP_TABLE_ENTRIES`, `CS_MAX_NUM_EEPROM_TABLE_ENTRIES`, `CS_MAX_NUM_MEMORY_TABLE_ENTRIES`, `CS_MAX_NUM_TABLES_TABLE_ENTRIES`, `CS_MEMORY_TABLE`, `CS_RESULTS_APP_TABLE_NAME`, `CS_RESULTS_EEPROM_TABLE_NAME`, `CS_RESULTS_MEMORY_TABLE_NAME`, `CS_RESULTS_TABLES_TABLE_NAME`, `CS_STATE_DISABLED`, `CS_TableInit()`, `CS_TABLES_TABLE`, `CS_ValidateAppChecksumDefinitionTable()`, `CS_ValidateEepromChecksumDefinitionTable()`, `CS_ValidateMemoryChecksumDefinitionTable()`, `CS_ValidateTablesChecksumDefinitionTable()`, `CS_AppData_t::DefAppTableHandle`, `CS_AppData_t::DefAppTblPtr`, `CS_AppData_t::DefaultAppDefTable`, `CS_AppData_t::DefaultEepromDefTable`, `CS_AppData_t::DefaultMemoryDefTable`, `CS_AppData_t::DefaultTablesDefTable`, `CS_AppData_t::DefEepromTableHandle`, `CS_AppData_t::DefEepromTblPtr`, `CS_AppData_t::DefTablesTableHandle`, `CS_AppData_t::DefTablesTblPtr`, `CS_HkPacket_Payload_t::EepromCSState`, `CS_HkPacket_Payload_t::MemoryCSState`, `CS_HkPacket_t::Payload`, `CS_AppData_t::ResAppTableHandle`, `CS_AppData_t::ResAppTblPtr`, `CS_AppData_t::ResEepromTableHandle`, `CS_AppData_t::ResEepromTblPtr`, `CS_AppData_t::ResTablesTableHandle`, `CS_AppData_t::ResTablesTblPtr`, and `CS_HkPacket_Payload_t::TablesCSState`. Referenced by `CS_AppInit()`.

Here is the call graph for this function:



**12.20.1.2 CS\_InitSegments()** void CS\_InitSegments ( void )

Initializes the cFE and OS segments for the Checksum Application.

#### Description

Initializes the cFE and OS segments for the Checksum Application.

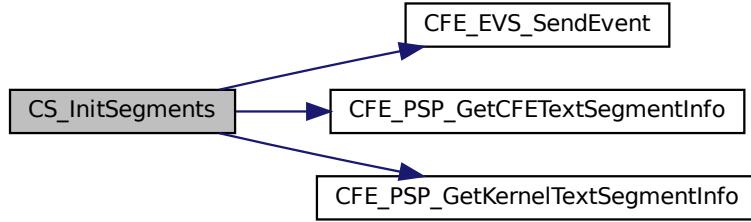
#### Assumptions, External Events, and Notes:

None

Definition at line 175 of file cs\_init.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_PSP\_GetCFETextSegmentInfo(), CFE\_PSP\_GetKernelTextSegmentInfo(), CS\_AppData\_t::CfeCoreCodeSeg, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CFE\_TEXT\_SEG\_INF\_EID, CS\_OS\_TEXT\_SEG\_INF\_EID, CS\_STAT\_E\_DISABLED, CS\_STATE\_ENABLED, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, OS\_SUCCESS, CS\_AppData\_t::OSCodeSeg, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue. Referenced by CS\_AppInit().

Here is the call graph for this function:



**12.20.1.3 CS\_SbInit()** `CFE_Status_t CS_SbInit ( void )`

Initializes the Software Bus Pipes for the Checksum Application.

#### Description

Configures cFE Software Bus resources used by the Checksum application.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

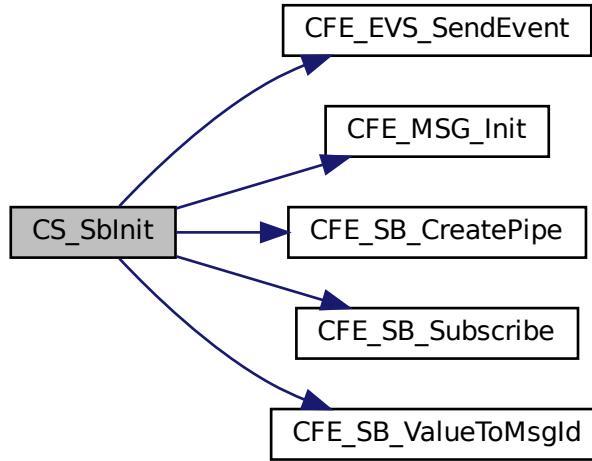
#### Return values

<code>CFE_SUCCESS</code>	Successful execution.
--------------------------	-----------------------

Definition at line 40 of file cs\_init.c.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_MSG_Init()`, `CFE_SB_CreatePipe()`, `CFE_SB_Subscribe()`, `CFE_SB_ValueToMsgId()`, `CFE_SUCCESS`, `CS_AppData_t::CmdPipe`, `CS_AppData`, `CS_BAC_KGROUND_CYCLE_MID`, `CS_CMD_MID`, `CS_CMD_PIPE_NAME`, `CS_CMD_PIPE_NAME_LEN`, `CS_HK_TLM_MID`, `CS_INIT_SB_CREATE_ERR_EID`, `CS_INIT_SB_SUBSCRIBE_BACK_ERR_EID`, `CS_INIT_SB_SUBSCRIBE_CMD_ERR_EID`, `CS_INIT_SB_SUBSCRIBE_HK_ERR_EID`, `CS_PIPE_DEPTH`, `CS_SEND_HK_MID`, `CS_AppData_t::HkPacket`, `CS_AppData_t::PipeDepth`, `CS_AppData_t::PipeName`, and `CS_HkPacket_t::TlmHeader`. Referenced by `CS_AppInit()`.

Here is the call graph for this function:



## 12.21 apps/cs/fsw/src/cs\_init.h File Reference

```
#include "cfe.h"
```

### Functions

- `CFE_Status_t CS_SbInit (void)`  
*Initializes the Software Bus Pipes for the Checksum Application.*
- `CFE_Status_t CS_InitAllTables (void)`  
*Initializes the tables for the Checksum Application.*
- `void CS_InitSegments (void)`  
*Initializes the cFE and OS segments for the Checksum Application.*

### 12.21.1 Detailed Description

Initialization subroutines for CS.

### 12.21.2 Function Documentation

#### 12.21.2.1 CS\_InitAllTables() `CFE_Status_t CS_InitAllTables (`     `void )`

Initializes the tables for the Checksum Application.

##### Description

Initializes all tables used by the Checksum application.

**Assumptions, External Events, and Notes:**

None

**Returns**

Execution status, see [cFE Return Code Defines](#)

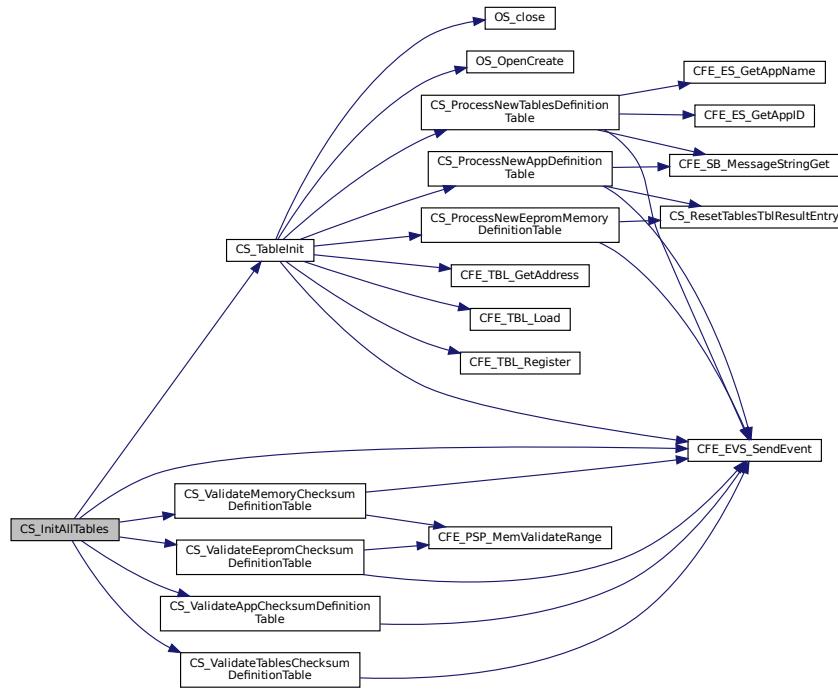
**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 100 of file cs\_init.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF↔ E\_SUCCESS, CS\_APP\_TABLE, CS\_AppData, CS\_DEF\_APP\_TABLE\_FILENAME, CS\_DEF\_APP\_TABLE\_NAME, CS\_DEF\_EEPROM\_TABLE\_FILENAME, CS\_DEF\_EEPROM\_TABLE\_NAME, CS\_DEF\_MEMORY\_TABLE\_FILENAME, CS\_DEF\_MEMORY\_TABLE\_NAME, CS\_DEF\_TABLES\_TABLE\_FILENAME, CS\_DEF\_TABLES\_TABLE\_NAME, CS\_EEPROM\_TABLE, CS\_INIT\_APP\_ERR\_EID, CS\_INIT\_EEPROM\_ERR\_EID, CS\_INIT\_MEMORY\_ERR\_EID, CS\_INIT\_TABLES\_ERR\_EID, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_RESULTS\_APP\_TABLE\_NAME, CS\_RESULTS\_EEPROM\_TABLE\_NAME, CS\_RESULTS\_MEMORY\_TABLE\_NAME, CS\_RESULTS\_TABLES\_TABLE\_NAME, CS\_STATE\_DISABLED, CS\_TableInit(), CS\_TABLES\_TABLE, CS\_ValidateAppChecksumDefinitionTable(), CS\_ValidateEepromChecksumDefinitionTable(), CS\_ValidateMemoryChecksumDefinitionTable(), CS\_ValidateTablesChecksumDefinitionTable(), CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::DefAppTblPtr, CS\_AppData\_t::DefaultAppDefTable, CS\_AppData\_t::DefaultEepromDefTable, CS\_AppData\_t::DefaultMemoryDefTable, CS\_AppData\_t::DefaultTablesDefTable, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::DefTablesTblPtr, CS\_HkPacket\_Payload\_t::EepromCState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResAppTableHandle, CS\_AppData\_t::ResAppTblPtr, CS\_AppData\_t::ResEepromTableHandle, CS\_AppData\_t::ResEepromTblPtr, CS\_AppData\_t::ResMemoryTableHandle, CS\_AppData\_t::ResMemoryTblPtr, CS\_AppData\_t::ResTablesTableHandle, CS\_AppData\_t::ResTablesTblPtr, and CS\_HkPacket\_Payload\_t::TablesCSState. Referenced by CS\_AppInit().

Here is the call graph for this function:



### 12.21.2.2 CS\_InitSegments()

```
void CS_InitSegments (
    void )
```

Initializes the cFE and OS segments for the Checksum Application.

#### Description

Initializes the cFE and OS segments for the Checksum Application.

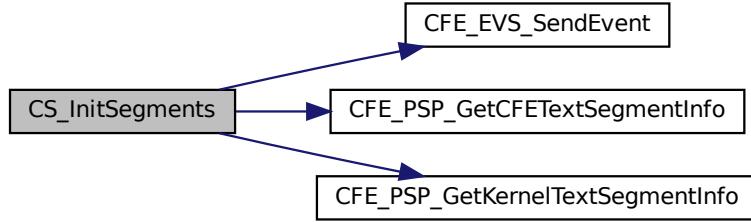
#### Assumptions, External Events, and Notes:

None

Definition at line 175 of file cs\_init.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_INFORMATION, CFE\_ES\_SendEvent(), CFE\_PSP\_GetCFETextSegmentInfo(), CFE\_PSP\_GetKernelTextSegmentInfo(), CS\_AppData\_t::CfeCoreCodeSeg, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_CFE\_TEXT\_SEG\_INF\_EID, CS\_OS\_TEXT\_SEG\_INF\_EID, CS\_STAT\_E\_DISABLED, CS\_STATE\_ENABLED, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, OS\_SUCCESS, CS\_AppData\_t::OSCodeSeg, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue. Referenced by CS\_AppInit().

Here is the call graph for this function:



**12.21.2.3 CS\_SbInit()** `CFE_Status_t` `CS_SbInit` (  
    `void` )

Initializes the Software Bus Pipes for the Checksum Application.

#### Description

Configures cFE Software Bus resources used by the Checksum application.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

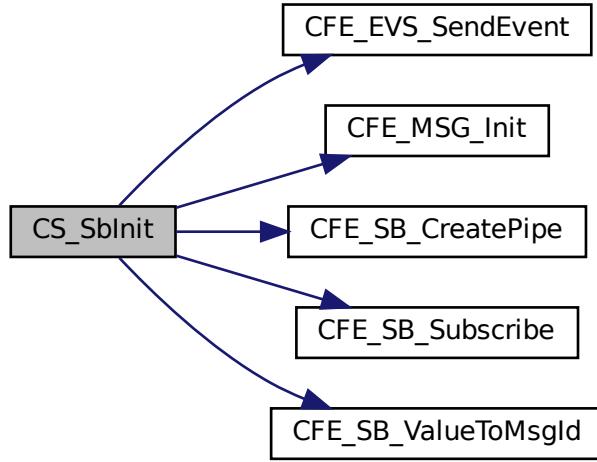
#### Return values

<code>CFE_SUCCESS</code>	Successful execution.
--------------------------	-----------------------

Definition at line 40 of file `cs_init.c`.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_MSG_Init()`, `CFE_SB_CreatePipe()`, `CFE_SB_Subscribe()`, `CFE_SB_ValueToMsgId()`, `CFE_SUCCESS`, `CS_AppData_t::CmdPipe`, `CS_AppData`, `CS_BAC_KGROUND_CYCLE_MID`, `CS_CMD_MID`, `CS_CMD_PIPE_NAME`, `CS_CMD_PIPE_NAME_LEN`, `CS_HK_TLM_MID`, `CS_INIT_SB_CREATE_ERR_EID`, `CS_INIT_SB_SUBSCRIBE_BACK_ERR_EID`, `CS_INIT_SB_SUBSCRIBE_CMD_ERR_EID`, `CS_INIT_SB_SUBSCRIBE_HK_ERR_EID`, `CS_PIPE_DEPTH`, `CS_SEND_HK_MID`, `CS_AppData_t::HkPacket`, `CS_AppData_t::PipeDepth`, `CS_AppData_t::PipeName`, and `CS_HkPacket_t::TlmHeader`.  
Referenced by `CS_AppInit()`.

Here is the call graph for this function:



## 12.22 apps/cs/fsw/src/cs\_memory\_cmds.c File Reference

```
#include "cfe.h"
#include "cs_tbldefs.h"
#include "cs_app.h"
#include "cs_events.h"
#include "cs_compute.h"
#include "cs_memory_cmds.h"
#include "cs_utils.h"
```

### Functions

- void [CS\\_DisableMemoryCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the Memory table command.*
- void [CS\\_EnableMemoryCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for the Memory table command.*
- void [CS\\_ReportBaselineEntryIDMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a report baseline of a Memory Entry command.*
- void [CS\\_RecomputeBaselineMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a recopmpute baseline of a Memory table entry command.*
- void [CS\\_EnableEntryIDMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for a Memory entry command.*
- void [CS\\_DisableEntryIDMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for a Memory entry command.*
- void [CS\\_GetEntryIDMemoryCmd](#) (const [CS\\_GetEntryIDCmd\\_t](#) \*CmdPtr)  
*Process a get Memory Entry by Address command.*

### 12.22.1 Detailed Description

The CFS Checksum (CS) Application's commands for checking Memory

### 12.22.2 Function Documentation

**12.22.2.1 CS\_DisableEntryIDMemoryCmd()** void CS\_DisableEntryIDMemoryCmd (

```
    const CS_EntryCmd_t * CmdPtr )
```

Process a disable background checking for a Memory entry command.

#### Description

Disables the specified Memory entry to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

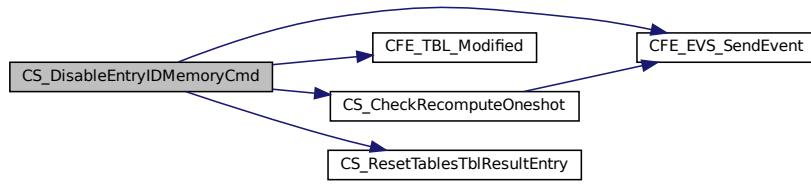
[CS\\_DISABLE\\_ENTRY\\_MEMORY\\_CC](#)

Definition at line 283 of file cs\_memory\_cmds.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID, CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID, CS\_Disable\_MEMORY\_INVALID\_ENTRY\_ERR\_EID, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_AppData\_t::MemResTablesTblPtr, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResMemoryTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.22.2.2 CS\_DisableMemoryCmd()** `void CS_DisableMemoryCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process a disable background checking for the Memory table command.

#### Description

Disables background checking for the Memory table

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

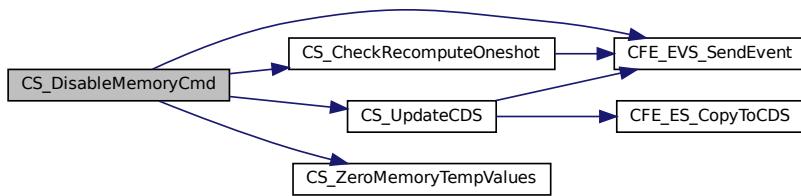
[CS\\_DISABLE\\_MEMORY\\_CC](#)

Definition at line 49 of file `cs_memory_cmds.c`.

References `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CS_HkPacket_Payload_t::CmdCounter`, `CS_AppData`, `CS_CheckRecomputeOneshot()`, `CS_DISABLE_MEMORY_INF_EID`, `CS_STATE_DISABLED`, `CS_UpdateCDS()`, `CS_ZeroMemoryTempValues()`, `CS_AppData_t::HkPacket`, `CS_HkPacket_Payload_t::MemoryCSState`, and `CS_HkPacket_t::Payload`.

Referenced by `CS_ProcessCmd()`.

Here is the call graph for this function:



**12.22.2.3 CS\_EnableEntryIDMemoryCmd()** `void CS_EnableEntryIDMemoryCmd ( const CS_EntryCmd_t * CmdPtr )`

Process an enable background checking for a Memory entry command.

#### Description

Allows the specified Memory entry to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

#### Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in CS_AppMain
----	---------------------	--

#### See also

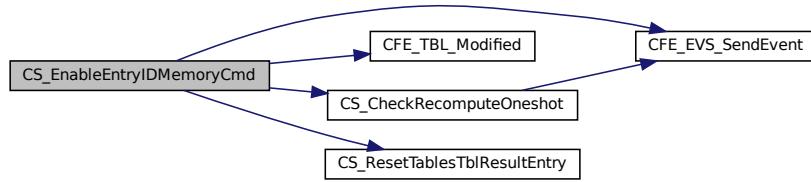
[CS\\_ENABLE\\_ENTRY\\_MEMORY\\_CC](#)

Definition at line 223 of file cs\_memory\_cmds.c.

References `CFE_EVS_EventType_DEBUG`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_TBL_Modified()`, `CS_HkPacket_Payload_t::CmdCounter`, `CS_HkPacket_Payload_t::CmdErrCounter`, `CS_AppData`, `CS_CheckRecomputeOneshot()`, `CS_ENABLE_MEMORY_DEF_EMPTY_DBG_EID`, `CS_ENABLE_MEMORY_ENTRY_INF_EID`, `CS_ENABLE_MEMORY_INVALID_ENTRY_ERR_EID`, `CS_MAX_NUM_MEMORY_TABLE_ENTRIES`, `CS_ResetTablesTblResultEntry()`, `CS_STATE_EMPTY`, `CS_STATE_ENABLED`, `CS_STATE_UNDEFINED`, `CS_AppData_t::DefMemoryTableHandle`, `CS_AppData_t::DefMemoryTblPtr`, `CS_EntryCmd_Payload_t::EntryID`, `CS_AppData_t::HkPacket`, `CS_AppData_t::MemResTablesTblPtr`, `CS_HkPacket_t::Payload`, `CS_EntryCmd_t::Payload`, `CS_AppData_t::ResMemoryTblPtr`, `CS_Def_EepromMemory_Table_Entry_t::State`, and `CS_Res_EepromMemory_Table_Entry_t::State`.

Referenced by `CS_ProcessCmd()`.

Here is the call graph for this function:



**12.22.2.4 CS\_EnableMemoryCmd()** `void CS_EnableMemoryCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process an enable background checking for the Memory table command.

#### Description

Allows the Memory table to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	<code>CmdPtr</code>	Command pointer, verified non-null in CS_AppMain
----	---------------------	--

#### See also

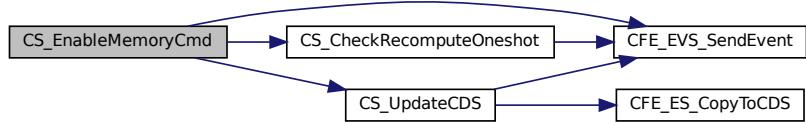
[CS\\_ENABLE\\_MEMORY\\_CC](#)

Definition at line 72 of file cs\_memory\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_MEMORY\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSSState, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



```
12.22.2.5 CS_GetEntryIDMemoryCmd() void CS_GetEntryIDMemoryCmd (
    const CS_GetEntryIDCmd_t * CmdPtr )
```

Process a get Memory Entry by Address command.

#### Description

Send the entry ID of the specified address in an event message

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

#### See also

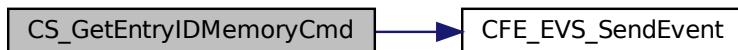
[CS\\_GET\\_ENTRY\\_ID\\_MEMORY\\_CC](#)

Definition at line 346 of file cs\_memory\_cmds.c.

References CS\_GetEntryIDCmd\_Payload\_t::Address, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID, CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::HkPacket, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_HkPacket\_t::Payload, CS\_GetEntryIDCmd\_t::Payload, CS\_AppData\_t::ResMemoryTblPtr, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



```
12.22.2.6 CS_RecomputeBaselineMemoryCmd() void CS_RecomputeBaselineMemoryCmd (
    const CS_EntryCmd_t * CmdPtr )
```

Process a recopmpute baseline of a Memory table entry command.

#### Description

Recomputes the checksum of a Memory table entry and use that value as the new baseline for that entry.

#### Assumptions, External Events, and Notes:

None

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

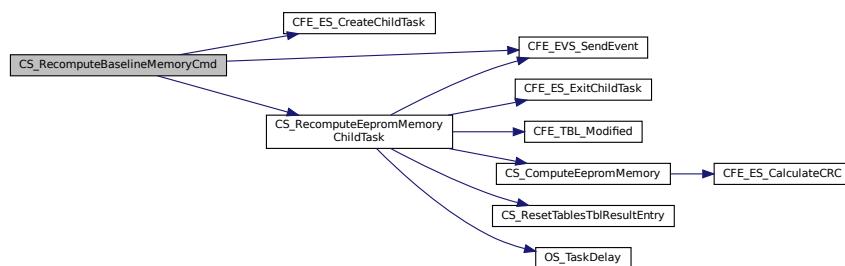
[CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#)

Definition at line 146 of file cs\_memory\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_ERROR, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_RECOMP\_MEMORY\_TASK\_NAME, CS\_RECOMPUTE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID, CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_MEMORY\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID, CS\_RecomputeEepromMemoryChildTask(), CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::ResMemoryTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.22.2.7 CS\_ReportBaselineEntryIDMemoryCmd()

```
void CS_ReportBaselineEntryIDMemoryCmd (
```

```
    const CS_EntryCmd_t * CmdPtr )
```

Process a report baseline of a Memory Entry command.

## Description

Reports the baseline checksum of the specified Memory table entry if it has already been computed

## Assumptions, External Events, and Notes:

None

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

See also

### [CS\\_REPORT\\_BASELINE\\_MEMORY\\_CC](#)

Definition at line 94 of file cs\_memory\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID, CS\_BASELINE\_MEMORY\_INF\_EID, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_NO\_BASELINE\_MEMORY\_INF\_EID, CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_Entry\_Cmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResMemoryTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.23 apps/cs/fsw/src/cs\_memory\_cmds.h File Reference

```
#include "cfe.h"
#include "cs_msg.h"
```

### Functions

- void [CS\\_DisableMemoryCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the Memory table command.*
- void [CS\\_EnableMemoryCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for the Memory table command.*
- void [CS\\_ReportBaselineEntryIDMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a report baseline of a Memory Entry command.*
- void [CS\\_DisableEntryIDMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for a Memory entry command.*
- void [CS\\_RecomputeBaselineMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process a recopmpute baseline of a Memory table entry command.*
- void [CS\\_EnableEntryIDMemoryCmd](#) (const [CS\\_EntryCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for a Memory entry command.*
- void [CS\\_GetEntryIDMemoryCmd](#) (const [CS\\_GetEntryIDCmd\\_t](#) \*CmdPtr)  
*Process a get Memory Entry by Address command.*

### 12.23.1 Detailed Description

Specification for the CFS Memory cmd

### 12.23.2 Function Documentation

**12.23.2.1 CS\_DisableEntryIDMemoryCmd()** `void CS_DisableEntryIDMemoryCmd ( const CS_EntryCmd_t * CmdPtr )`

Process a disable background checking for a Memory entry command.

#### Description

Disables the specified Memory entry to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

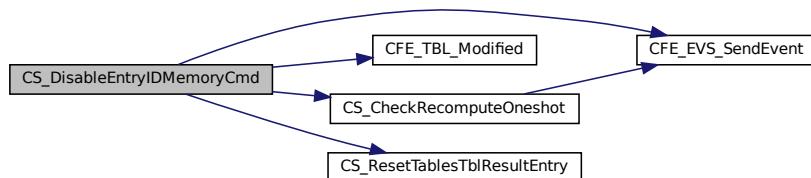
[CS\\_DISABLE\\_ENTRY\\_MEMORY\\_CC](#)

Definition at line 283 of file cs\_memory\_cmds.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID, CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID, CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_EntryCmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_AppData\_t::MemResTablesTblPtr, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResMemoryTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.23.2.2 CS\_DisableMemoryCmd()** `void CS_DisableMemoryCmd ( const CS_NoArgsCmd_t * CmdPtr )`

Process a disable background checking for the Memory table command.

### Description

Disables background checking for the Memory table

### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

### See also

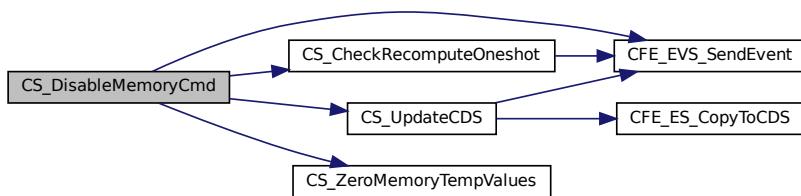
[CS\\_DISABLE\\_MEMORY\\_CC](#)

Definition at line 49 of file cs\_memory\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_MEMORY\_INF\_EID, CS\_STATE\_DISABLED, CS\_UpdateCDS(), CS\_ZeroMemoryTempValues(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, and CS\_HkPacket\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.23.2.3 CS\_EnableEntryIDMemoryCmd()

```
void CS_EnableEntryIDMemoryCmd (
    const CS_EntryCmd_t * CmdPtr )
```

Process an enable background checking for a Memory entry command.

### Description

Allows the specified Memory entry to be background checksummed.

### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

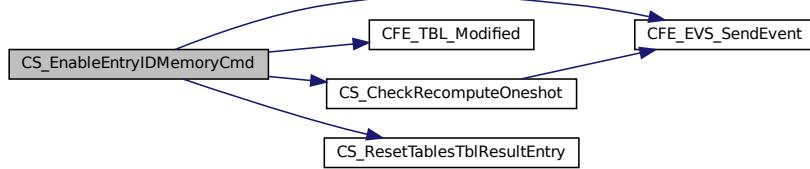
[CS\\_ENABLE\\_ENTRY\\_MEMORY\\_CC](#)

Definition at line 223 of file cs\_memory\_cmds.c.

References CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID, CS\_ENABLE\_MEMORY\_ENTRY\_INF\_EID, CS\_ENABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_ResetTablesTblResultEntry(), CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_STATE\_UNDEFINED, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_EntryCmd::Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_AppData\_t::MemResTablesTblPtr, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResMemoryTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.23.2.4 CS\_EnableMemoryCmd()** void CS\_EnableMemoryCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process an enable background checking for the Memory table command.

## Description

Allows the Memory table to be background checksummed.

## Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

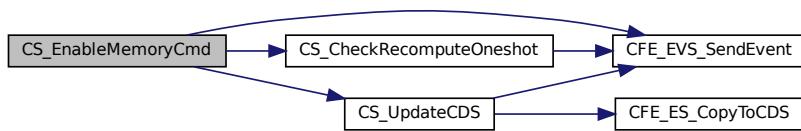
### See also

#### [CS\\_ENABLE\\_MEMORY\\_CC](#)

Definition at line 72 of file cs\_memory\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_MEMORY\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, and CS\_HkPacket\_t::Payload. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.23.2.5 CS\_GetEntryIDMemoryCmd()

```
void CS_GetEntryIDMemoryCmd (
    const CS_GetEntryIDCmd_t * CmdPtr )
```

Process a get Memory Entry by Address command.

#### Description

Send the entry ID of the specified address in an event message

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

See also

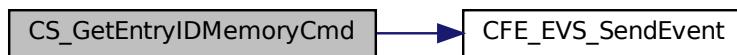
[CS\\_GET\\_ENTRY\\_ID\\_MEMORY\\_CC](#)

Definition at line 346 of file cs\_memory\_cmds.c.

References CS\_GetEntryIDCmd\_Payload\_t::Address, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID, CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::HkPacket, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_HkPacket\_t::Payload, CS\_GetEntryIDCmd\_t::Payload, CS\_AppData\_t::ResMemoryTblPtr, CS\_Res\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.23.2.6 CS\_RecomputeBaselineMemoryCmd()

```
void CS_RecomputeBaselineMemoryCmd (
```

```
    const CS_EntryCmd_t * CmdPtr )
```

Process a recopmpute baseline of a Memory table entry command.

#### Description

Recomputes the checksum of a Memory table entry and use that value as the new baseline for that entry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	CmdPtr	Command pointer, verified non-null in CS_AppMain
----	--------	--

See also

[CS\\_RECOMPUTE\\_BASELINE\\_MEMORY\\_CC](#)

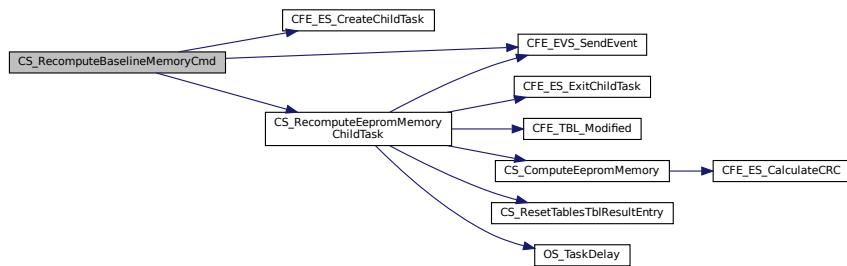
Definition at line 146 of file cs\_memory\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_ES\_TASKID\_UNDEFINED, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CS\_AppData\_t::ChildTaskEntryID, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_ERROR, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_RECOMP\_MEMORY\_TASK\_NAME, CS\_RECOMPUTE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID, CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_MEMORY\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID, CS\_RecomputeEepromMemoryChildTask(), CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_EntryCmd\_Payload\_t::EntryID, CS

\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::RecomputeEepromMemoryEntryPtr, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::ResMemoryTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.23.2.7 CS\_ReportBaselineEntryIDMemoryCmd()

```
void CS_ReportBaselineEntryIDMemoryCmd (
    const CS_EntryCmd_t * CmdPtr )
```

Process a report baseline of a Memory Entry command.

#### Description

Reports the baseline checksum of the specified Memory table entry if it has already been computed

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

[CS\\_REPORT\\_BASELINE\\_MEMORY\\_CC](#)

Definition at line 94 of file cs\_memory\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID, CS\_BASELINE\_MEMORY\_INF\_EID, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_NO\_BASELINE\_MEMORY\_INF\_EID, CS\_STATE\_EMPTY, CS\_STATE\_UNDEFINED, CS\_Entry\_Cmd\_Payload\_t::EntryID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_EntryCmd\_t::Payload, CS\_AppData\_t::ResMemoryTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.24 apps/cs/fsw/src/cs\_table\_cmds.c File Reference

```
#include "cfe.h"
#include "cs_app.h"
#include "cs_events.h"
#include "cs_utils.h"
#include "cs_compute.h"
#include "cs_table_cmds.h"
```

### Functions

- void [CS\\_DisableTablesCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the Tables table command.*
- void [CS\\_EnableTablesCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for the Tables table command.*
- void [CS\\_ReportBaselineTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process a report baseline of a Table command.*
- void [CS\\_RecomputeBaselineTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process a recompute baseline of a Table command.*
- void [CS\\_DisableNameTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for a Table entry command.*
- void [CS\\_EnableNameTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for a Table entry command.*

### 12.24.1 Detailed Description

The CFS Checksum (CS) Application's commands for checking Tables

## 12.24.2 Function Documentation

**12.24.2.1 CS\_DisableNameTablesCmd()** `void CS_DisableNameTablesCmd ( const CS_TableNameCmd_t * CmdPtr )`

Process a disable background checking for a Table entry command.

### Description

Disables the specified Tables entry to be background checksummed.

### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

### See also

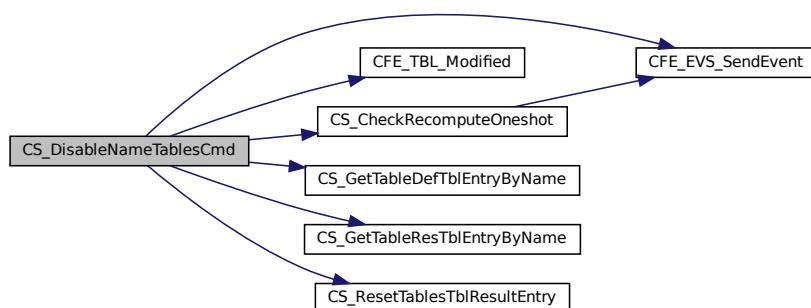
[CS\\_DISABLE\\_NAME\\_TABLE\\_CC](#)

Definition at line 188 of file cs\_table\_cmds.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_DISABLE\_TABLES\_NAME\_INF\_EID, CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetTableDefTblEntryByName(), CS\_GetTableResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::HkPacket, CS\_TableNameCmd\_Payload\_t::Name, CS\_HkPacket\_t::Payload, CS\_TableNameCmd\_t::Payload, CS\_Def\_Tables\_Table\_Entry\_t::State, CS\_Res\_Tables\_Table\_Entry\_t::State, CS\_AppData\_t::TblResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.24.2.2 CS\_DisableTablesCmd()** `void CS_DisableTablesCmd (`  
 `const CS_NoArgsCmd_t * CmdPtr )`

Process a disable background checking for the Tables table command.

#### Description

Disables background checking for the Tables table

#### Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

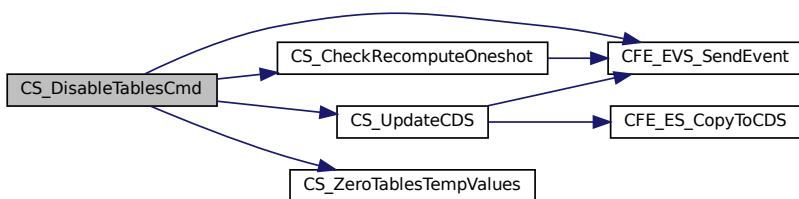
[CS\\_DISABLE\\_TABLES\\_CC](#)

Definition at line 47 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_TABLES\_INF\_EID, CS\_STATE\_DISABLED, CS\_UpdateCDS(), CS\_ZeroTablesTempValues(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.24.2.3 CS\_EnableNameTablesCmd()** `void CS_EnableNameTablesCmd (`  
 `const CS_TableNameCmd_t * CmdPtr )`

Process an enable background checking for a Table entry command.

#### Description

Allows the specified table to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

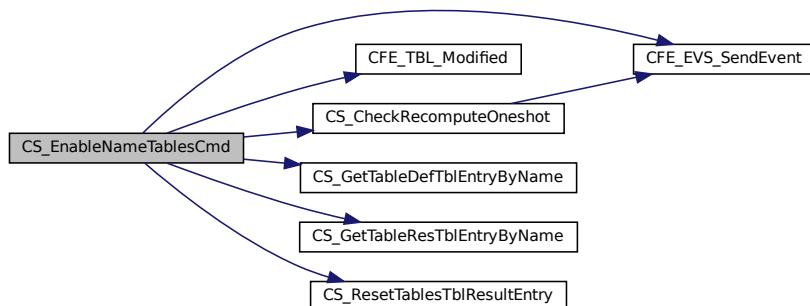
[CS\\_ENABLE\\_NAME\\_TABLE\\_CC](#)

Definition at line 236 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_ENABLE\_TABLES\_NAME\_INF\_EID, CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetTableDefTblEntryByName(), CS\_GetTableResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_ENABLED, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::HkPacket, CS\_TableNameCmd\_Payload\_t::Name, CS\_HkPacket\_t::Payload, CS\_TableNameCmd\_t::Payload, CS\_DefTablesTable\_Entry\_t::State, CS\_ResTablesTable\_Entry\_t::State, and CS\_AppData\_t::TblResTablesTblPtr.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.24.2.4 CS\_EnableTablesCmd()** void CS\_EnableTablesCmd (const [CS\\_NoArgsCmd\\_t](#) \* *CmdPtr*)

Process an enable background checking for the Tables table command.

#### Description

Allows the Tables table to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

**See also**

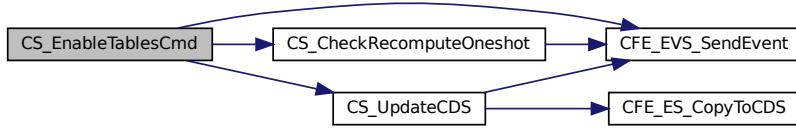
[CS\\_ENABLE\\_TABLES\\_CC](#)

Definition at line 69 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_TABLES\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSSState.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



### 12.24.2.5 CS\_RecomputeBaselineTablesCmd()

```
void CS_RecomputeBaselineTablesCmd (
    const CS_TableNameCmd_t * CmdPtr )
```

Process a recompute baseline of a Table command.

**Description**

Recomputes the checksum of a table and use that value as the new baseline for that entry.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

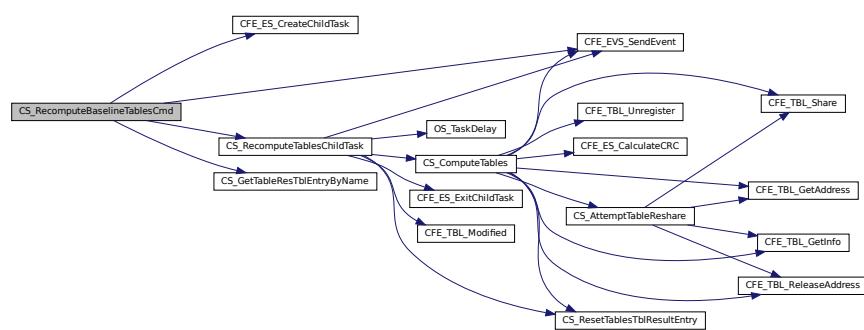
**See also**

[CS\\_RECOMPUTE\\_BASELINE\\_TABLE\\_CC](#)

Definition at line 127 of file cs\_table\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CFE\_TBL\_MAX\_FULL\_NAMELEN, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrorCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_GetTableResTblEntryByName(), CS\_RECOMP\_TABLES\_TASK\_NAME, CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID, CS\_RECOMPUTE\_UNKNOWN\_NAME\_TABLES\_ERR\_EID, CS\_RecomputeTablesChildTask(), CS\_TABLES\_TABLE, CS\_AppData\_t::HkPacket, CS\_TableName\_Cmd\_Payload\_t::Name, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_TableName\_Cmd\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, and CS\_AppData\_t::RecomputeTablesEntryPtr. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.24.2.6 CS\_ReportBaselineTablesCmd()** void CS\_ReportBaselineTablesCmd ( const CS\_TableNameCmd t \* CmdPtr )

Process a report baseline of a Table command.

Process a report baseline or a table command.

### Description

Reports the baseline checksum of the specified table if it has already been computed.

## **Assumptions, External Events, and Notes.**

None

## Parameters

in *CmdPtr* Command pointer, verified non-null in CS\_AppMain

## See also

## CS\_REPORT\_BASELINE\_TABLE\_CC

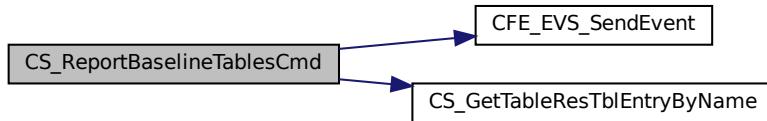
Definition at line 90 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_E\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_Tables\_Table\_Entry\_t::ComparisonValue, CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID, CS\_BASELINE\_TABLES\_INF\_EID, CS\_GetTableResTblEntry()

ByName(), CS\_NO\_BASELINE\_TABLES\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_TableNameCmd\_Payload\_t::Name, CS\_HkPacket\_t::Payload, and CS\_TableNameCmd\_t::Payload.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



## 12.25 apps/cs/fsw/src/cs\_table\_cmds.h File Reference

```
#include "cfe.h"
#include "cs_msg.h"
```

### Functions

- void [CS\\_DisableTablesCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for the Tables table command.*
- void [CS\\_EnableTablesCmd](#) (const [CS\\_NoArgsCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for the Tables table command.*
- void [CS\\_ReportBaselineTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process a report baseline of a Table command.*
- void [CS\\_RecomputeBaselineTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process a recompute baseline of a Table command.*
- void [CS\\_DisableNameTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process a disable background checking for a Table entry command.*
- void [CS\\_EnableNameTablesCmd](#) (const [CS\\_TableNameCmd\\_t](#) \*CmdPtr)  
*Process an enable background checking for a Table entry command.*

### 12.25.1 Detailed Description

Specification for the CFS table cmds

### 12.25.2 Function Documentation

#### 12.25.2.1 [CS\\_DisableNameTablesCmd\(\)](#) void CS\_DisableNameTablesCmd (

```
const CS\_TableNameCmd\_t * CmdPtr )
```

Process a disable background checking for a Table entry command.

#### Description

Disables the specified Tables entry to be background checksummed.

#### Assumptions, External Events, and Notes:

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled, and overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

#### Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

#### See also

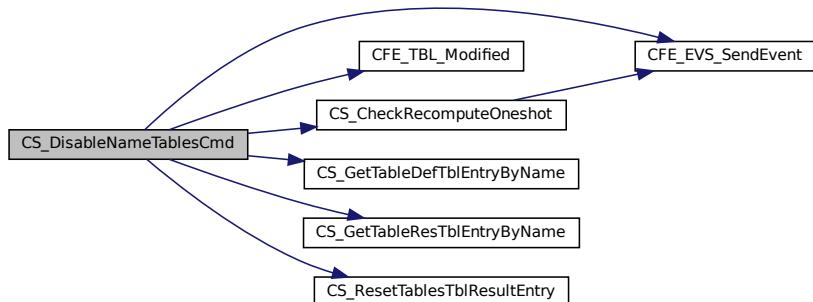
[CS\\_DISABLE\\_NAME\\_TABLE\\_CC](#)

Definition at line 188 of file cs\_table\_cmds.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_E←RROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_MAX\_FULL\_NAME\_LEN, C←FE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_App←Data, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_DISABLE\_TAB←LES\_NAME\_INF\_EID, CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetTableDefTblEntryByName(), CS\_GetTableResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_AppData\_t::←DefTablesTableHandle, CS\_AppData\_t::HkPacket, CS\_TableNameCmd\_Payload\_t::Name, CS\_HkPacket\_t::Payload, CS\_TableNameCmd\_t::Payload, CS\_Def\_Tables\_Table\_Entry\_t::State, CS\_Res\_Tables\_Table\_Entry\_t::State, CS←AppData\_t::TblResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.25.2.2 CS\_DisableTablesCmd()** void CS\_DisableTablesCmd ( const CS\_NoArgsCmd\_t \* CmdPtr )

Process a disable background checking for the Tables table command.

#### Description

Disables background checking for the Tables table

**Assumptions, External Events, and Notes:**

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

**See also**

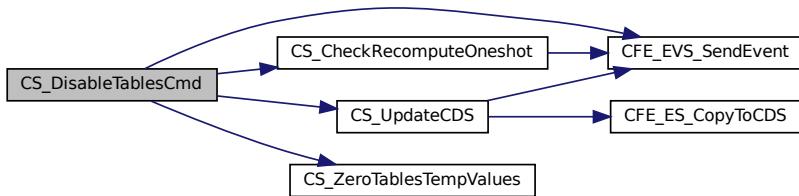
[CS\\_DISABLE\\_TABLES\\_CC](#)

Definition at line 47 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_DISABLE\_TABLES\_INF\_EID, CS\_STATE\_DISABLED, CS\_UpdateCDS(), CS\_ZeroTablesTempValues(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.25.2.3 CS\_EnableNameTablesCmd()** void CS\_EnableNameTablesCmd ( const CS\_TableNameCmd\_t \* CmdPtr )

Process an enable background checking for a Table entry command.

**Description**

Allows the specified table to be background checksummed.

**Assumptions, External Events, and Notes:**

In order for background checking of individual entries to checksum to occur, the entry must be enabled, the table must be enabled and, overall checksumming must be enabled. This command updates both the results table and the definition table. If the entry exists in the results table but not in the definition table, the command is still successful, but the definition table is not updated. If the entry does not exist in the results table, neither table is updated.

**Parameters**

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

## See also

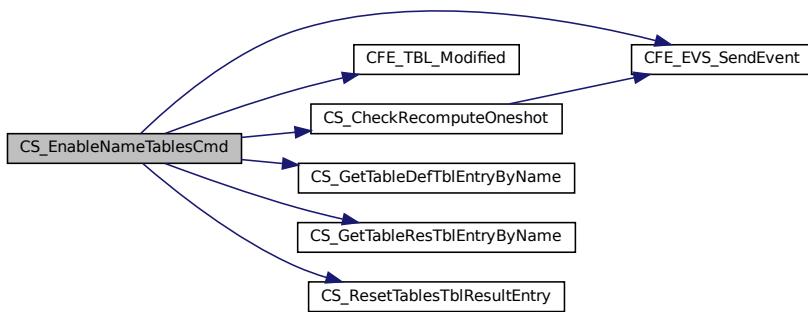
[CS\\_ENABLE\\_NAME\\_TABLE\\_CC](#)

Definition at line 236 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CFE\_TBL\_Modified(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID, CS\_ENABLE\_TABLES\_NAME\_INF\_EID, CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID, CS\_GetTableDefTblEntryByName(), CS\_GetTableResTblEntryByName(), CS\_ResetTablesTblResultEntry(), CS\_STATE\_ENABLED, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::HkPacket, CS\_TableNameCmd\_Payload\_t::Name, CS\_HkPacket\_t::Payload, CS\_TableNameCmd\_t::Payload, CS\_DefTablesTable\_Entry\_t::State, CS\_ResTablesTable\_Entry\_t::State, and CS\_AppData\_t::TblResTablesTblPtr.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.25.2.4 CS\_EnableTablesCmd()** void CS\_EnableTablesCmd (

const CS\_NoArgsCmd\_t \* CmdPtr )

Process an enable background checking for the Tables table command.

## Description

Allows the Tables table to be background checksummed.

## Assumptions, External Events, and Notes:

In order for background checking of individual areas to checksum (OS code segment, cFE core, EEPROM, Memory, Apps, and Tables) to occur, the table must be enabled and overall checksumming must be enabled.

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

See also

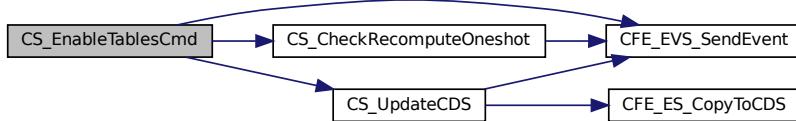
[CS\\_ENABLE\\_TABLES\\_CC](#)

Definition at line 69 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdCounter, CS\_AppData, CS\_CheckRecomputeOneshot(), CS\_ENABLE\_TABLES\_INF\_EID, CS\_STATE\_ENABLED, CS\_UpdateCDS(), CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.25.2.5 CS\_RecomputeBaselineTablesCmd()** `void CS_RecomputeBaselineTablesCmd ( const CS_TableNameCmd_t * CmdPtr )`

Process a recompute baseline of a Table command.

Description

Recomputes the checksum of a table and use that value as the new baseline for that entry.

Assumptions, External Events, and Notes:

None

Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

See also

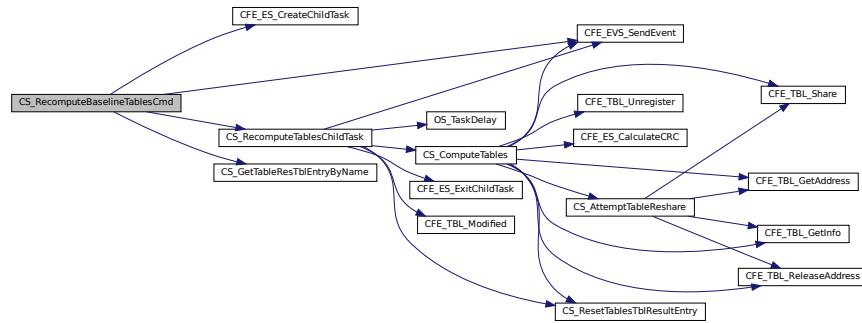
[CS\\_RECOMPUTE\\_BASELINE\\_TABLE\\_CC](#)

Definition at line 127 of file cs\_table\_cmds.c.

References CFE\_ES\_CreateChildTask(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_EventType\_ERROR, CFE\_ES\_SendEvent(), CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, CFE\_SUCCESS, CFE\_TBL\_MAX\_FULL\_NAMELEN, CS\_AppData\_t::ChildTaskTable, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrorCounter, CS\_AppData, CS\_CHILD\_TASK\_PRIORITY, CS\_GetTableResTblEntryByName(), CS\_RECOMP\_TABLES\_TASK\_NAME, CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID, CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID, CS\_RECOMPUTE\_UNKNOWN\_NAME\_TABLE\_ES\_ERR\_EID, CS\_RecomputeTablesChildTask(), CS\_TABLES\_TABLE, CS\_AppData\_t::HkPacket, CS\_TableName\_Cmd\_Payload\_t::Name, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_TableName\_Cmd\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, and CS\_AppData\_t::RecomputeTablesEntryPtr.

Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.25.2.6 CS\_ReportBaselineTablesCmd()** void CS\_ReportBaselineTablesCmd (

```
const CS_TableNameCmd_t * CmdPtr )
```

Process a report baseline of a Table command.

## Description

Reports the baseline checksum of the specified table if it has already been computed

## **Assumptions, External Events, and Notes:**

None

## Parameters

in	<i>CmdPtr</i>	Command pointer, verified non-null in CS_AppMain
----	---------------	--

### See also

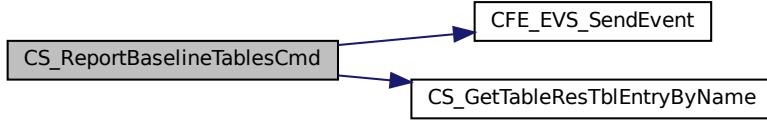
CS\_REPORT\_BASELINE\_TABLE\_CC

Definition at line 90 of file cs\_table\_cmds.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_E\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_HkPacket\_Payload\_t::CmdCounter, CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_Res\_Tables\_Table\_Entry\_t::ComparisonValue, CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID, CS\_BASELINE\_TABLES\_INF\_EID, CS\_GetTableResTblEntryByName(), CS\_NO\_BASELINE\_TABLES\_INF\_EID, CS\_AppData\_t::HkPacket, CS\_TableNameCmd\_Payload\_t::Name, CS\_HkPacket\_t::Payload, and CS\_TableNameCmd\_t::Payload.

Referenced by CS ProcessCmd().

Here is the call graph for this function:



## 12.26 apps/cs/fsw/src/cs\_table\_processing.c File Reference

```
#include "cfe.h"
#include "cs_app.h"
#include "cs_events.h"
#include "cs_tbldefs.h"
#include "cs_utils.h"
#include <string.h>
```

### Functions

- **CFE\_Status\_t CS\_ValidateEepromChecksumDefinitionTable (void \*TblPtr)**  
*Validate EEPROM definition table.*
- **CFE\_Status\_t CS\_ValidateMemoryChecksumDefinitionTable (void \*TblPtr)**  
*Validate Memory definition table.*
- **CFE\_Status\_t CS\_ValidateTablesChecksumDefinitionTable (void \*TblPtr)**  
*Validate Tables definition table.*
- **CFE\_Status\_t CS\_ValidateAppChecksumDefinitionTable (void \*TblPtr)**  
*Validate App definition table.*
- void **CS\_ProcessNewEepromMemoryDefinitionTable (const CS\_Def\_EepromMemory\_Table\_Entry\_t \*DefinitionTblPtr, const CS\_Res\_EepromMemory\_Table\_Entry\_t \*ResultsTblPtr, uint16 NumEntries, uint16 Table)**  
*Processes a new definition table for EEPROM or Memory tables.*
- void **CS\_ProcessNewTablesDefinitionTable (const CS\_Def\_Tables\_Table\_Entry\_t \*DefinitionTblPtr, const CS\_Res\_Tables\_Table\_Entry\_t \*ResultsTblPtr)**  
*Processes a new definition table for the Tables table.*
- void **CS\_ProcessNewAppDefinitionTable (const CS\_Def\_App\_Table\_Entry\_t \*DefinitionTblPtr, const CS\_Res\_App\_Table\_Entry\_t \*ResultsTblPtr)**  
*Processes a new definition table for the App table.*
- **CFE\_Status\_t CS\_TableInit (CFE\_TBL\_Handle\_t \*DefinitionTableHandle, CFE\_TBL\_Handle\_t \*ResultsTableHandle, void \*DefinitionTblPtr, void \*ResultsTblPtr, uint16 Table, const char \*DefinitionTableName, const char \*ResultsTableName, uint16 NumEntries, const char \*DefinitionFileName, const void \*DefaultDefTableAddress, uint16 SizeofDefinitionTableEntry, uint16 SizeofResultsTableEntry, CFE\_TBL\_CallbackFuncPtr\_t CallBackFunction)**  
*Initializes the results and definition table on startup.*
- **CFE\_Status\_t CS\_HandleTableUpdate (void \*DefinitionTblPtr, void \*ResultsTblPtr, CFE\_TBL\_Handle\_t DefinitionTableHandle, CFE\_TBL\_Handle\_t ResultsTableHandle, uint16 Table, uint16 NumEntries)**  
*Handles table updates for all CS tables.*

### 12.26.1 Detailed Description

The CFS Checksum (CS) Application's table updating functions

### 12.26.2 Function Documentation

#### 12.26.2.1 CS\_HandleTableUpdate() [CFE\\_Status\\_t](#) CS\_HandleTableUpdate (

```
    void * DefinitionTblPtr,
    void * ResultsTblPtr,
    const CFE_TBL_Handle_t DefinitionTableHandle,
    const CFE_TBL_Handle_t ResultsTableHandle,
    const uint16 Table,
    const uint16 NumEntries )
```

Handles table updates for all CS tables.

#### Description

Completes the handshake with Table Services that releases Addresses for the tables and checks for updates

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>DefinitionTblPtr</i>	A pointer to the definition table that we are operating on
in	<i>ResultsTblPtr</i>	A pointer to the result table to operate on
in	<i>DefinitionTableHandle</i>	A table handle to the definition table
in	<i>ResultsTableHandle</i>	A table handle to the results table
in	<i>NumEntries</i>	The number of entries in the table
in	<i>Table</i>	The specific table we are operating on

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

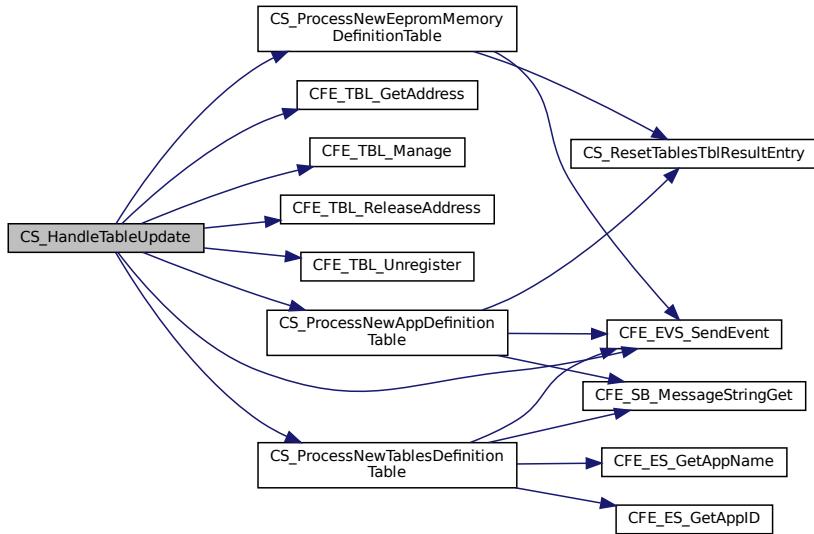
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 957 of file cs\_table\_processing.c.

References [CFE\\_EVS\\_EventType\\_ERROR](#), [CFE\\_EVS\\_SendEvent\(\)](#), [CFE\\_SUCCESS](#), [CFE\\_TBL\\_BAD\\_TABLE\\_HANDLE](#), [CFE\\_TBL\\_GetAddress\(\)](#), [CFE\\_TBL\\_INFO\\_UPDATED](#), [CFE\\_TBL\\_Manage\(\)](#), [CFE\\_TBL\\_ReleaseAddress\(\)](#), [CFE\\_TBL\\_Unregister\(\)](#), [CS\\_APP\\_TABLE](#), [CS\\_AppData](#), [CS EEPROM\\_TABLE](#), [CS\\_MAX\\_NUM\\_TABLES](#), [TABLE\\_ENTRIES](#), [CS\\_MEMORY\\_TABLE](#), [CS\\_ProcessNewAppDefinitionTable\(\)](#), [CS\\_ProcessNewEepromMemoryDefinitionTable\(\)](#), [CS\\_ProcessNewTablesDefinitionTable\(\)](#), [CS\\_TABLES\\_TABLE](#), [CS\\_TABLETYPE\\_NAME\\_SIZE](#), [CS\\_TBL\\_UPDATE\\_ERR\\_EID](#), [CS\\_Res\\_Tables\\_Table\\_Entry\\_t::IsCSCOwner](#), [CS\\_AppData\\_t::ResTablesTblPtr](#), and [CS\\_Res\\_Tables\\_Table\\_Entry\\_t::TblHandle](#).

Referenced by [CS\\_HandleRoutineTableUpdates\(\)](#).

Here is the call graph for this function:



**12.26.2.2 CS\_ProcessNewAppDefinitionTable()** `void CS_ProcessNewAppDefinitionTable ( const CS_Def_App_Table_Entry_t * DefinitionTblPtr, const CS_Res_App_Table_Entry_t * ResultsTblPtr )`

Processes a new definition table for the App table.

#### Description

Copies data from the definition table to the results table because the results table is where CS keeps all of its checksum data

#### Assumptions, External Events, and Notes:

None

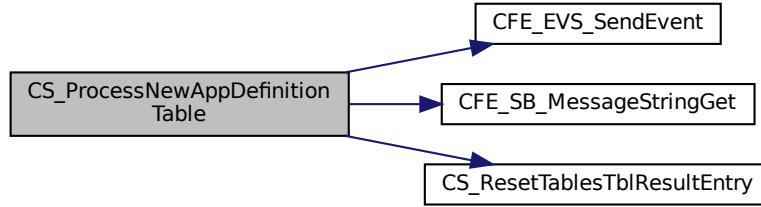
#### Parameters

in	<i>DefinitionTblPtr</i>	A pointer to the definiton table ( <a href="#">CS_Def_App_Table_Entry_t</a> ) that we are operating on
in	<i>ResultsTblPtr</i>	A pointer to the result table ( <a href="#">CS_Res_App_Table_Entry_t</a> ) to operate on

Definition at line 741 of file `cs_table_processing.c`.

References `CS_HkPacket_Payload_t::AppCSSState`, `CS_AppData_t::AppResTablesTblPtr`, `CS_Res_App_Table_Entry_t::ByteOffset`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_SB_MessageStringGet()`, `CS_Res_App_Table_Entry_t::ComparisonValue`, `CS_Res_App_Table_Entry_t::ComputedYet`, `CS_AppData`, `CS_MAX_NUM_APP_TABLE_ENTRIES`, `CS_PROCESS_APP_NO_ENTRIES_INF_EID`, `CS_ResetTablesTblResultEntry()`, `CS_STATE_DISABLED`, `CS_STATE_EMPTY`, `CS_AppData_t::HkPacket`, `CS_Def_App_Table_Entry_t::Name`, `CS_Res_App_Table_Entry_t::Name`, `CS_Res_App_Table_Entry_t::NumBytesToChecksum`, `CS_HkPacket_t::Payload`, `CS_Res_App_Table_Entry_t::StartAddress`, `CS_Def_App_Table_Entry_t::State`, `CS_Res_App_Table_Entry_t::State`,

and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.  
Referenced by CS\_HandleTableUpdate(), and CS\_TableInit().  
Here is the call graph for this function:



**12.26.2.3 CS\_ProcessNewEepromMemoryDefinitionTable()** void CS\_ProcessNewEepromMemoryDefinitionTable (

```

const CS_Def_EepromMemory_Table_Entry_t * DefinitionTblPtr,
const CS_Res_EepromMemory_Table_Entry_t * ResultsTblPtr,
const uint16 NumEntries,
const uint16 Table )
  
```

Processes a new definition table for EEPROM or Memory tables.

#### Description

Copies data from the definition table to the results table because the results table is where CS keeps all of its checksum data

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>DefinitionTblPtr</i>	A pointer to the definition table ( <a href="#">CS_Def_EepromMemory_Table_Entry_t</a> ) that we are operating on
in	<i>ResultsTblPtr</i>	A pointer to the result table ( <a href="#">CS_Res_EepromMemory_Table_Entry_t</a> ) to operate on
in	<i>NumEntries</i>	The number of entries in the table
in	<i>Table</i>	The specific table we are operating on

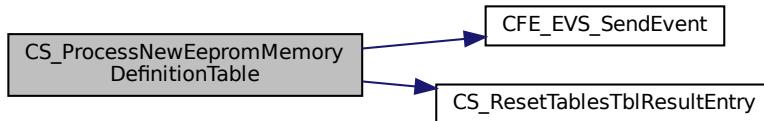
Definition at line 453 of file cs\_table\_processing.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_EEPROM\_TABLE, CS\_MEMORY\_TABLE, CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID, CS\_ResetTablesTblResultEntry(), CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_TABLETYPE\_NAME\_SIZE, CS\_AppData\_t::EepResTablesTblPtr, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_AppData\_t::MemResTablesTblPtr, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_Res\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_HkPacket\_t::Payload, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Res\_Eeprom

Memory\_Table\_Entry\_t::StartAddress, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Res\_EepromMemory\_Table\_Entry\_t::State, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_HandleTableUpdate(), and CS\_TableInit().

Here is the call graph for this function:



**12.26.2.4 CS\_ProcessNewTablesDefinitionTable()** void CS\_ProcessNewTablesDefinitionTable ( const CS\_Def\_Tables\_Table\_Entry\_t \* DefinitionTblPtr, const CS\_Res\_Tables\_Table\_Entry\_t \* ResultsTblPtr )

Processes a new definition table for the Tables table.

#### Description

Copies data from the definition table to the results table because the results table is where CS keeps all of its checksum data

#### Assumptions, External Events, and Notes:

None

#### Parameters

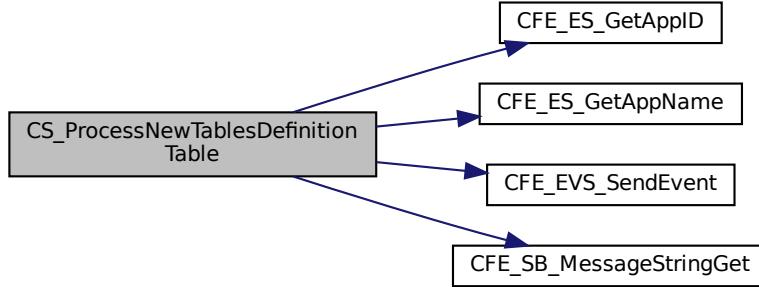
in	<i>DefinitionTblPtr</i>	A pointer to the definition table ( <a href="#">CS_Def_Tables_Table_Entry_t</a> ) that we are operating on
in	<i>ResultsTblPtr</i>	A pointer to the results table ( <a href="#">CS_Res_Tables_Table_Entry_t</a> ) to operate on

Definition at line 548 of file cs\_table\_processing.c.

References CS\_AppData\_t::AppResTablesTblPtr, CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CFE\_ES\_APPID\_UNDEFINED, CFE\_ES\_GetAppID(), CFE\_ES\_GetAppName(), CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, CFE\_SB\_MessageStringGet(), CFE\_TBL\_BAD\_TABLE\_HANDLE, CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_Res\_Tables\_Table\_Entry\_t::ComparisonValue, CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, CS\_AppData, CS\_DEF\_APP\_TABLE\_NAME, CS\_DEF\_EEPROM\_TABLE\_NAME, CS\_DEF\_MEMORY\_TABLE\_NAME, CS\_DEF\_TABLES\_TABLE\_NAME, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID, CS\_RESULTS\_APP\_TABLE\_NAME, CS\_RESULTS\_EEPROM\_TABLE\_NAME, CS\_RESULTS\_MEMORY\_TABLE\_NAME, CS\_RESULTS\_TABLES\_TABLE\_NAME, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::EepromTablesTblPtr, CS\_AppData\_t::HkPacket, CS\_Res\_Tables\_Table\_Entry\_t::IsCsowner, CS\_AppData\_t::MemResTablesTblPtr, CS\_Def\_Tables\_Table\_Entry\_t::Name, CS\_Res\_Tables\_Table\_Entry\_t::Name, CS\_Res\_Tables\_Table\_Entry\_t::NumBytesToChecksum, OS\_MAX\_API\_NAME, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResAppTableHandle, CS\_AppData\_t::ResEepromTableHandle, CS\_AppData\_t::ResMemoryTableHandle, CS\_AppData\_t::ResTablesTableHandle, CS\_Res\_Tables\_Table\_Entry\_t::StartAddress, CS\_Def\_Tables\_Table\_Entry\_t::State, CS\_Res\_Tables\_Table\_Entry\_t::State, CS\_HkPacket\_Payload\_t::TablesCSState, CS\_Res\_Tables\_Table\_Entry\_t::TblHandle,

CS\_AppData\_t::TblResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue. Referenced by CS\_HandleTableUpdate(), and CS\_TableInit().

Here is the call graph for this function:



```

12.26.2.5 CS_TableInit() CFE_Status_t CS_TableInit (
    CFE_TBL_Handle_t * DefinitionTableHandle,
    CFE_TBL_Handle_t * ResultsTableHandle,
    void * DefinitionTblPtr,
    void * ResultsTblPtr,
    const uint16 Table,
    const char * DefinitionTableName,
    const char * ResultsTableName,
    const uint16 NumEntries,
    const char * DefinitionTableFileName,
    const void * DefaultDefTableAddress,
    const uint16 sizeofDefinitionTableEntry,
    const uint16 sizeofResultsTableEntry,
    const CFE_TBL_CallbackFuncPtr_t CallBackFunction )
  
```

Initializes the results and definition table on startup.

#### Description

Completes the Table Services registration and table load for the definition table and the registration for the results table

#### Assumptions, External Events, and Notes:

This function is used on all four type of tables individually. Also, if the default table file is not found for the definition table, this function loads a 'blank' table from memory

#### Parameters

in	<i>DefinitionTableHandle</i>	A <b>CFE_TBL_Handle_t</b> pointer that will get filled in with the table handle to the definition table
in	<i>ResultsTableHandle</i>	A <b>CFE_TBL_Handle_t</b> pointer that will get filled in with the table handle to the results table

**Parameters**

in	<i>DefinitionTblPtr</i>	A pointer to the definition table that we are operating on, it will get assigned during this call
in	<i>ResultsTblPtr</i>	A pointer to the result table to operate on, it will get assigned during this call
in	<i>Table</i>	The specific table we are operating on
in	<i>DefinitionTableName</i>	The name of the definition table
in	<i>ResultsTableName</i>	The name of the results table
in	<i>NumEntries</i>	The number of entries in the table
in	<i>DefinitionTableFileName</i>	The name of the file to load the definition table from
in	<i>DefaultDefTableAddress</i>	The address of the default definition table that we may have to load from memory if the file is absent
in	<i>SizeofDefinitionTableEntry</i>	The sizeof an entry in the definition table
in	<i>SizeofResultsTableEntry</i>	The size of an entry in the results table
in	<i>CallBackFunction</i>	A pointer to a function used to validate the definition table
out	*	DefinitionTableHandle A <a href="#">CFE_TBL_Handle_t</a> pointer that will get filled in with the tbl handle to the definition table
out	*	ResultsTableHandle A <a href="#">CFE_TBL_Handle_t</a> pointer that will get filled in with the tbl handle to the results table
out	*	DefinitionTblPtr A pointer to the definition table that we are operating on
in	*	ResultsTblPtr A pointer to the result table to operate on, it will get be used to access the table

**Returns**

Execution status, see [cFE Return Code Defines](#)

**Return values**

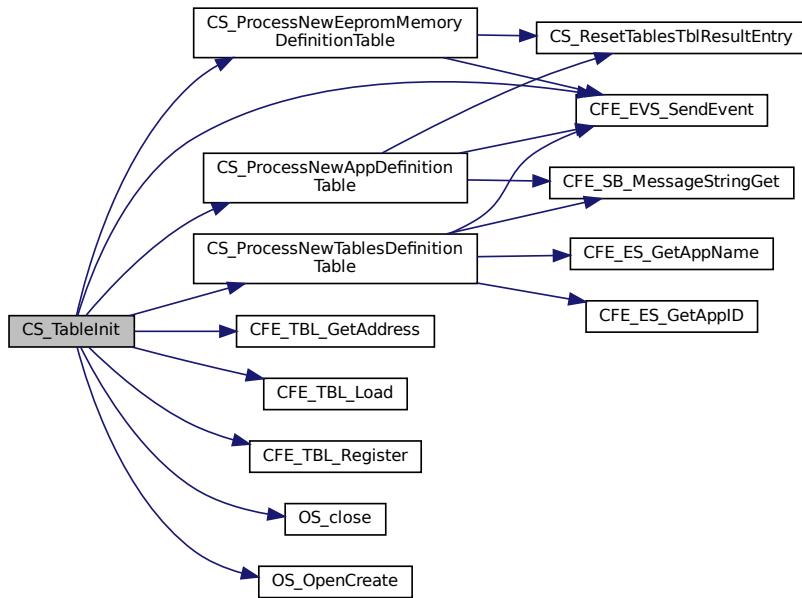
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 813 of file cs\_table\_processing.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_GetAddress(), CFE\_TBL\_INFO\_UPDATED, CFE\_TBL\_Load(), CFE\_TBL\_OPT\_DUMP\_ONLY, CFE\_TBL\_OPT\_LOAD\_DUMP, CFE\_TBL\_OPT\_SNGL\_BUFFER, CFE\_TBL\_Register(), CFE\_TBL\_SRC\_ADDRESS, CFE\_TBL\_SRC\_FILE, CS\_APP\_TABLE, CS\_AppData, CS\_EEPROM\_TABLE, CS\_MEMORY\_TABLE, CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEepromMemoryDefinitionTable(), CS\_ProcessNewTablesDefinitionTable(), CS\_STATE\_DISABLED, CS\_TABLES\_TABLE, CS\_TABLETYPE\_NAME\_SIZE, CS\_TBL\_INIT\_ERR\_EID, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, OS\_close(), OS\_ERROR, OS\_FILE\_FLAG\_NONE, OS\_OBJECT\_ID\_UNDEFINED, OS\_OpenCreate(), OS\_READ\_ONLY, OS\_SUCCESS, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_InitAllTables().

Here is the call graph for this function:



**12.26.2.6 CS\_ValidateAppChecksumDefinitionTable()** `CFE_Status_t CS_ValidateAppChecksumDefinitionTable( void * TblPtr )`

Validate App definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>TblPtr</code>	A pointer to the table to be validated
----	---------------------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

#### Return values

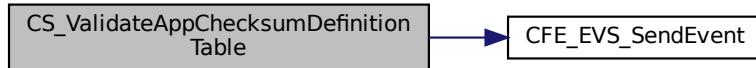
<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CS_TABLE_ERROR</code>	Error code returned on table validation error.

Definition at line 328 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID, CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID, CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID, CS\_VAL\_APP\_INF\_EID, CS\_VAL\_APP\_STAT\_E\_ERR\_EID, CS\_Def\_App\_Table\_Entry\_t::Name, and CS\_Def\_App\_Table\_Entry\_t::State.

Referenced by CS\_InitAllTables().

Here is the call graph for this function:



**12.26.2.7 CS\_ValidateEepromChecksumDefinitionTable()** [CFE\\_Status\\_t](#) CS\_ValidateEepromChecksumDefinitionTable (void \* TblPtr )

Validate EEPROM definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	TblPtr	A pointer to the table to be validated
----	--------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

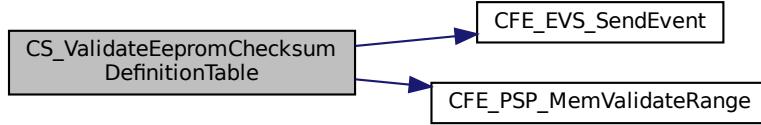
#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
<a href="#">CS_TABLE_ERROR</a>	Error code returned on table validation error.

Definition at line 48 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_PSP\_MEM\_EEPROM, CFE\_PSP\_MemValidateRange(), CFE\_SUCCESS, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_EEPROM\_INF\_EID, CS\_VAL\_EEPROM\_RANGE\_ERR\_EID, CS\_VAL\_EEPROM\_STATE\_ERR\_EID, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, OS\_ERROR, OS\_SUCCESS, CS\_Def\_EepromMemory\_Table\_

Entry\_t::StartAddress, and CS\_Def\_EepromMemory\_Table\_Entry\_t::State.  
 Referenced by CS\_InitAllTables().  
 Here is the call graph for this function:



**12.26.2.8 CS\_ValidateMemoryChecksumDefinitionTable()** [CFE\\_Status\\_t](#) CS\_ValidateMemoryChecksum<  
 DefinitionTable (   
     void \* TblPtr )  
 Validate Memory definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	TblPtr	A pointer to the table to be validated
----	--------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

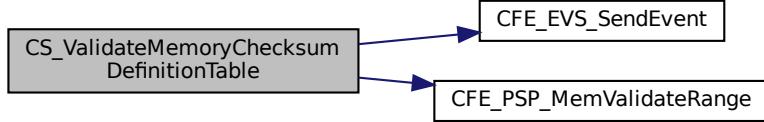
#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
<a href="#">CS_TABLE_ERROR</a>	Error code returned on table validation error.

Definition at line 128 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_PSP\_MEM\_ANY, CFE\_PSP\_MemValidateRange(), CFE\_SUCCESS, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_MEMORY\_INF\_EID, CS\_VAL\_MEMORY\_RANGE\_ERR\_EID, CS\_VAL\_MEMORY\_STATE\_ERR\_EID, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, OS\_ERROR, OS\_SUCCESS, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, and CS\_Def\_EepromMemory\_Table\_Entry\_t::State.  
 Referenced by CS\_InitAllTables().

Here is the call graph for this function:



**12.26.2.9 CS\_ValidateTablesChecksumDefinitionTable()** [CFE\\_Status\\_t](#) CS\_ValidateTablesChecksumDefinitionTable ( [void \\* TblPtr](#) )  
Validate Tables definition table.

#### Description

This function is a callback to cFE Table Services that gets called when a validation is requested.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>TblPtr</i>	A pointer to the table to be validated
----	---------------	--

#### Returns

Execution status see [cFE Return Code Defines](#)

#### Return values

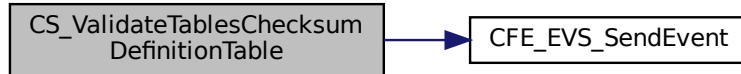
<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
<a href="#">CS_TABLE_ERROR</a>	Error code returned on table validation error.

Definition at line 212 of file cs\_table\_processing.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_DISABLED, CS\_STATE\_EMPTY, CS\_STATE\_ENABLED, CS\_TABLE\_ERROR, CS\_VAL\_TABLES\_DEF\_TBL\_DUPLEX\_EID, CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID, CS\_VAL\_TABLES\_INF\_EID, CS\_VAL\_TABLES\_STATUS\_ERR\_EID, CS\_DefTables\_Table\_Entry\_t::Name, and CS\_DefTables\_Table\_Entry\_t::State.

Referenced by CS\_InitAllTables().

Here is the call graph for this function:



## 12.27 apps/cs/fsw/src/cs\_utils.c File Reference

```
#include "cfe.h"
#include "cs_app.h"
#include "cs_events.h"
#include "cs_compute.h"
#include "cs_utils.h"
#include <string.h>
```

### Functions

- void [CS\\_ZeroEepromTempValues](#) (void)  
*Zeros out temporary checksum values of EEPROM table entries.*
- void [CS\\_ZeroMemoryTempValues](#) (void)  
*Zeros out temporary checksum values of Memory table entries.*
- void [CS\\_ZeroTablesTempValues](#) (void)  
*Zeros out temporary checksum values of Tables table entries.*
- void [CS\\_ZeroAppTempValues](#) (void)  
*Zeros out temporary checksum values of App table entries.*
- void [CS\\_ZeroCfeCoreTempValues](#) (void)  
*Zeros out temporary checksum values of the cFE Core.*
- void [CS\\_ZeroOSTempValues](#) (void)  
*Zeros out temporary checksum values of the OS code segment.*
- void [CS\\_InitializeDefaultTables](#) (void)  
*Initializes the default definition tables.*
- void [CS\\_GoToNextTable](#) (void)  
*Moves global variables to point to the next table.*
- bool [CS\\_GetTableResTblEntryByName](#) ([CS\\_Res\\_Tables\\_Table\\_Entry\\_t](#) \*\*EntryPtr, const char \*Name)  
*Gets a pointer to the results entry given a table name.*
- bool [CS\\_GetTableDefTblEntryByName](#) ([CS\\_Def\\_Tables\\_Table\\_Entry\\_t](#) \*\*EntryPtr, const char \*Name)  
*Gets a pointer to the definition entry given a table name.*
- bool [CS\\_GetAppResTblEntryByName](#) ([CS\\_Res\\_App\\_Table\\_Entry\\_t](#) \*\*EntryPtr, const char \*Name)  
*Gets a pointer to the results entry given an app name.*
- bool [CS\\_GetAppDefTblEntryByName](#) ([CS\\_Def\\_App\\_Table\\_Entry\\_t](#) \*\*EntryPtr, const char \*Name)  
*Gets a pointer to the definition entry given an app name.*
- bool [CS\\_FindEnabledEepromEntry](#) (uint16 \*EnabledEntry)  
*Find an enabled EEPROM entry.*

- bool `CS_FindEnabledMemoryEntry` (`uint16 *EnabledEntry`)  
*Find an enabled Memory entry.*
- bool `CS_FindEnabledTablesEntry` (`uint16 *EnabledEntry`)  
*Find an enabled Tables entry.*
- bool `CS_FindEnabledAppEntry` (`uint16 *EnabledEntry`)  
*Find an enabled App entry.*
- bool `CS_VerifyCmdLength` (`const CFE_MSG_Message_t *msg, size_t ExpectedLength`)  
*Verify command message length.*
- bool `CS_BackgroundCfeCore` (`void`)  
*Compute a background check cycle on the cFE Core.*
- bool `CS_BackgroundOS` (`void`)  
*Compute a background check cycle on the OS.*
- bool `CS_BackgroundEeprom` (`void`)  
*Compute a background check cycle on EEPROM.*
- bool `CS_BackgroundMemory` (`void`)  
*Compute a background check cycle on the Memory.*
- bool `CS_BackgroundTables` (`void`)  
*Compute a background check cycle on Tables.*
- bool `CS_BackgroundApp` (`void`)  
*Compute a background check cycle on Apps.*
- void `CS_ResetTablesTblResultEntry` (`CS_Res_Tables_Table_Entry_t *TablesTblResultEntry`)  
*Reset the checksum for a CS table entry in the CS tables table.*
- `CFE_Status_t CS_HandleRoutineTableUpdates` (`void`)  
*Update All CS Tables.*
- `CFE_Status_t CS_AttemptTableReshare` (`CS_Res_Tables_Table_Entry_t *ResultsEntry, CFE_TBL_Handle_t *LocalTblHandle, CFE_TBL_Info_t *TblInfo, cpuaddr *LocalAddress, int32 *ResultGetInfo`)  
*Attempts to re-share a table.*
- bool `CS_CheckRecomputeOneshot` (`void`)

### 12.27.1 Detailed Description

The CFS Checksum (CS) Application's utility functions

### 12.27.2 Function Documentation

**12.27.2.1 `CS_AttemptTableReshare()`** `CFE_Status_t CS_AttemptTableReshare (`  
`CS_Res_Tables_Table_Entry_t * ResultsEntry,`  
`CFE_TBL_Handle_t * LocalTblHandle,`  
`CFE_TBL_Info_t * TblInfo,`  
`cpuaddr * LocalAddress,`  
`int32 * ResultGetInfo )`

Attempts to re-share a table.

#### Description

This function is called if the first attempt to share the table is unsuccessful. This function attempts to share the table again to see if it reappeared. Calling function ensures that parameters are non-null.

**Assumptions, External Events, and Notes:**

None

**Returns**

Execution status, see [cFE Return Code Defines](#)

**Return values**

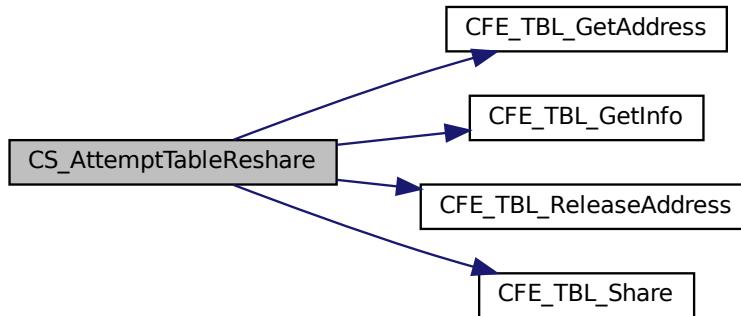
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 1004 of file cs\_utils.c.

References CFE\_SUCCESS, CFE\_TBL\_ERR\_NEVER\_LOADED, CFE\_TBL\_GetAddress(), CFE\_TBL\_GetInfo(), CFE\_TBL\_ReleaseAddress(), CFE\_TBL\_Share(), CS\_Res\_Tables\_Table\_Entry\_t::Name, and CS\_Res\_Tables\_Table\_Entry\_t::TblHandle.

Referenced by CS\_ComputeTables().

Here is the call graph for this function:



### 12.27.2.2 CS\_BackgroundApp()

```
bool CS_BackgroundApp (
    void )
```

Compute a background check cycle on Apps.

**Description**

This routine will try and complete a cycle of background checking

**Assumptions, External Events, and Notes:**

None

**Returns**

Boolean checksum performed response

#### Return values

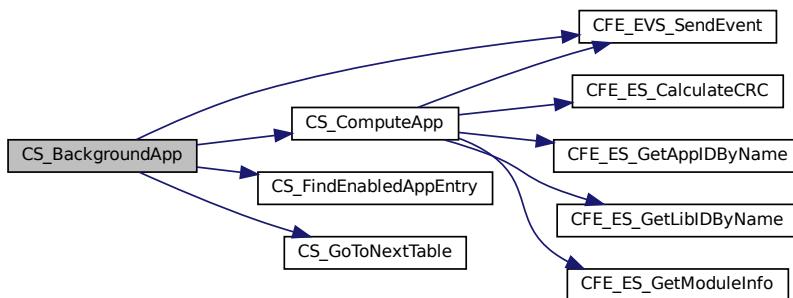
<i>true</i>	Checksum was done
<i>false</i>	Checksum was not done

Definition at line 830 of file cs\_utils.c.

References CS\_HkPacket\_Payload\_t::AppCSErrCounter, CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_Res\_App\_Table\_Entry\_t::ComparisonValue, CS\_APP\_MISCOMPARE\_ERR\_EID, CS\_AppData, CS\_COMPUTE\_APP\_NOT\_FOUND\_ERR\_EID, CS\_ComputeApp(), CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_FindEnabledAppEntry(), CS\_GoToNextTable(), CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_Res\_App\_Table\_Entry\_t::Name, CS\_HkPacket\_t::Payload, and CS\_AppData\_t::ResAppTblPtr.

Referenced by CS\_BackgroundCheckCycle().

Here is the call graph for this function:



**12.27.2.3 CS\_BackgroundCfeCore()** `bool CS_BackgroundCfeCore ( void )`

Compute a background check cycle on the cFE Core.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

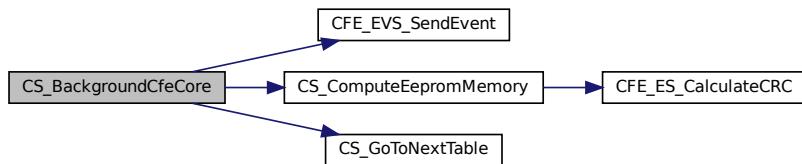
<i>true</i>	Checksum was done
<i>false</i>	Checksum was not done

Definition at line 477 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CfeCoreBaseline, CS\_AppData\_t::CfeCoreCodeSeg, CS\_HkPacket\_Payload\_t::CfeCoreCSErrCounter, CS\_HkPacket\_Payload\_t::CfeCoreCSState, CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_AppData, CS\_CFECORE\_MISC\_OMPARE\_ERR\_EID, CS\_ComputeEepromMemory(), CS\_ERROR, CS\_GoToNextTable(), CS\_STATE\_ENABLED, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundCheckCycle().

Here is the call graph for this function:



#### 12.27.2.4 CS\_BackgroundEeprom() `bool CS_BackgroundEeprom ( void )`

Compute a background check cycle on EEPROM.

##### Description

This routine will try and complete a cycle of background checking

##### Assumptions, External Events, and Notes:

None

##### Returns

Boolean checksum performed response

##### Return values

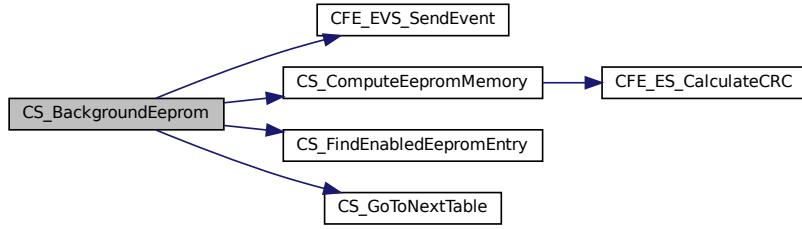
<code>true</code>	Checksum was done
<code>false</code>	Checksum was not done

Definition at line 609 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_AppData, CS\_ComputeEepromMemory(), CS\_EEPROM\_MISCOMPARE\_ERR\_EID, CS\_ERROR, CS\_FindEnabledEepromEntry(), CS\_GoToNextTable(), CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_HkPacket\_Payload\_t::EepromBaseline, CS\_HkPacket\_Payload\_t::EepromCSErrCounter, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, and CS\_AppData\_t::ResEepromTblPtr.

Referenced by CS\_BackgroundCheckCycle().

Here is the call graph for this function:



**12.27.2.5 CS\_BackgroundMemory()** `bool CS_BackgroundMemory ( void )`

Compute a background check cycle on the Memory.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

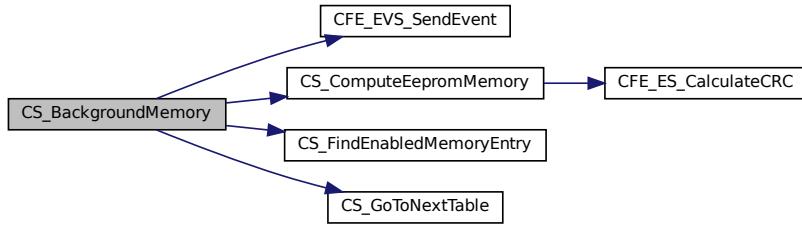
<i>true</i>	Checksum was done
<i>false</i>	Checksum was not done

Definition at line 684 of file cs\_utils.c.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CS_Res_EepromMemory_Table_Entry_t::ComparisonValue`, `CS_AppData`, `CS_ComputeEepromMemory()`, `CS_ERROR`, `CS_FindEnabledMemoryEntry()`, `CS_GoToNextTable()`, `CS_MAX_NUM_MEMORY_TABLE_ENTRIES`, `CS_MEMORY_MISCOMPARE_ERR_EID`, `CS_STATE_ENABLED`, `CS_HkPacket_Payload_t::CurrentEntryInTable`, `CS_AppData_t::HkPacket`, `CS_HkPacket_Payload_t::MemoryCSErrCounter`, `CS_HkPacket_Payload_t::MemoryCSState`, `CS_HkPacket_t::Payload`, and `CS_AppData_t::ResMemoryTblPtr`.

Referenced by `CS_BackgroundCheckCycle()`.

Here is the call graph for this function:



**12.27.2.6 CS\_BackgroundOS()** `bool CS_BackgroundOS ( void )`

Compute a background check cycle on the OS.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

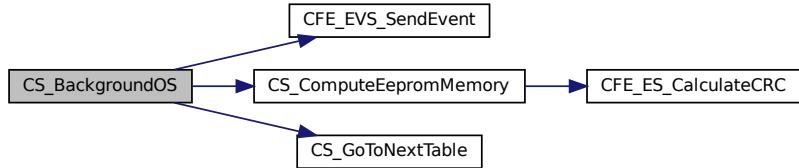
<code>true</code>	Checksum was done
<code>false</code>	Checksum was not done

Definition at line 544 of file `cs_utils.c`.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CS_Res_EepromMemory_Table_Entry_t::ComparisonValue`, `CS_AppData`, `CS_ComputeEepromMemory()`, `CS_ERROR`, `CS_GoToNextTable()`, `CS_OS_M::ISCOMPARE_ERR_EID`, `CS_STATE_ENABLED`, `CS_HkPacket_Payload_t::CurrentEntryInTable`, `CS_AppData_t::HkPacket`, `CS_HkPacket_Payload_t::OSBaseline`, `CS_AppData_t::OSCodeSeg`, `CS_HkPacket_Payload_t::OSCS::ErrCounter`, `CS_HkPacket_Payload_t::OSCSSState`, `CS_HkPacket_t::Payload`, and `CS_Res_EepromMemory_Table_Entry_t::State`.

Referenced by `CS_BackgroundCheckCycle()`.

Here is the call graph for this function:



**12.27.2.7 CS\_BackgroundTables()** `bool CS_BackgroundTables ( void )`

Compute a background check cycle on Tables.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

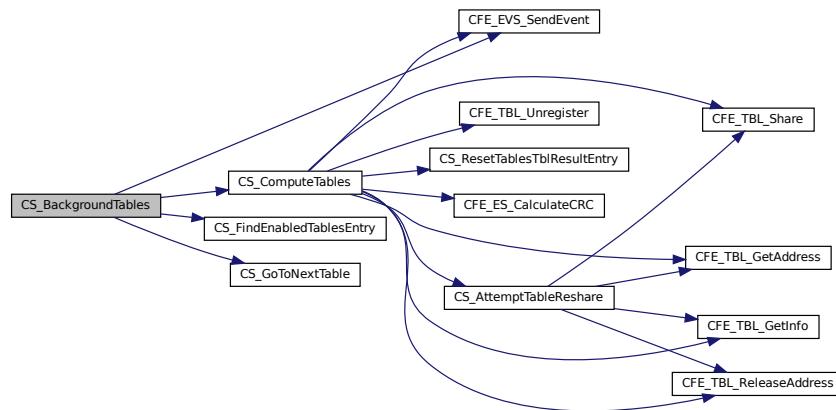
#### Return values

<code>true</code>	Checksum was done
<code>false</code>	Checksum was not done

Definition at line 753 of file cs\_utils.c.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CS_Res_Tables_Table_Entry_t::ComparisonValue`, `CS_AppData`, `CS_COMPUTE_TABLES_NOT_FOUND_ERR_EID`, `CS_ComputeTables()`, `CS_ERR_NOT_FOUND`, `CS_ERROR`, `CS_FindEnabledTablesEntry()`, `CS_GoToNextTable()`, `CS_MAX_NUM_TABLES_TABLE_ENTRIES`, `CS_STATE_ENABLED`, `CS_TABLES_MISCOMPARE_ERR_EID`, `CS_HkPacket_Payload_t::CurrentEntryInTable`, `CS_AppData_t::HkPacket`, `CS_Res_Tables_Table_Entry_t::Name`, `CS_HkPacket_t::Payload`, `CS_AppData_t::ResTablesTblPtr`, `CS_HkPacket_Payload_t::TablesCSErrCounter`, and `CS_HkPacket_Payload_t::TablesCSState`. Referenced by `CS_BackgroundCheckCycle()`.

Here is the call graph for this function:



### 12.27.2.8 CS\_CheckRecomputeOneshot()

```
bool CS_CheckRecomputeOneshot (
    void )
```

Definition at line 1032 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CMD\_COMPUTE\_PROG\_ERR\_EID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::RecomputeInProgress.

Referenced by CS\_DisableAppCmd(), CS\_DisableEepromCmd(), CS\_DisableEntryIDEepromCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableTablesCmd(), CS\_EnableAppCmd(), CS\_EnableEepromCmd(), CS\_EnableEntryIDEepromCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), and CS\_EnableTablesCmd().

Here is the call graph for this function:



### 12.27.2.9 CS\_FindEnabledAppEntry()

```
bool CS_FindEnabledAppEntry (
    uint16 * EnabledEntry )
```

Find an enabled App entry.

#### Description

This routine will look from the current position to the end of the table to find an enabled entry.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

**Returns**

Boolean enabled entry found response

**Return values**

<i>true</i>	Enabled entry was found in the table
<i>false</i>	Enabled entry was not found in the table

Definition at line 413 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_Payload<\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResAppTblPtr, and CS\_Res\_App\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundApp().

**12.27.2.10 CS\_FindEnabledEepromEntry()** `bool CS_FindEnabledEepromEntry (`  
`uint16 * EnabledEntry )`

Find an enabled EEPROM entry.

**Description**

This routine will look from the current position to the end of the table to find an enabled entry.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

**Returns**

Boolean entry found response

**Return values**

<i>true</i>	Entry was found in the table
<i>false</i>	Entry was not found in the table

Definition at line 312 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_↔ Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResEeprom↔ TblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundEeprom().

**12.27.2.11 CS\_FindEnabledMemoryEntry()** `bool CS_FindEnabledMemoryEntry (`  
`uint16 * EnabledEntry )`

Find an enabled Memory entry.

#### Description

This routine will look from the current position to the end of the table to find an enabled entry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

#### Returns

Boolean enabled entry found response

#### Return values

<i>true</i>	Enabled entry was found in the table
<i>false</i>	Enabled entry was not found in the table

Definition at line 346 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_↔ Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResMemory↔ TblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundMemory().

**12.27.2.12 CS\_FindEnabledTablesEntry()** `bool CS_FindEnabledTablesEntry (`  
`uint16 * EnabledEntry )`

Find an enabled Tables entry.

#### Description

This routine will look from the current position to the end of the table to find an enabled entry.

#### Assumptions, External Events, and Notes:

None

**Parameters**

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

**Returns**

Boolean enabled entry found response

**Return values**

<i>true</i>	Enabled entry was found in the table
<i>false</i>	Enabled entry was not found in the table

Definition at line 379 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket ↪ Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResTablesTbl ↪ Ptr, and CS\_Res\_Tables\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundTables().

**12.27.2.13 CS\_GetAppDefTblEntryByName()** `bool CS_GetAppDefTblEntryByName (`

```
    CS\_Def\_App\_Table\_Entry\_t ** EntryPtr,
    const char * Name )
```

Gets a pointer to the definition entry given an app name.

**Description**

This routine will look through the App Definition table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Name</i>	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	<i>EntryPtr</i>	A pointer to a <a href="#">CS_Def_App_Table_Entry_t</a> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

**Returns**

Boolean name found response

**Return values**

<i>true</i>	Name was found in the table
<i>false</i>	Name was not found in the table

Definition at line 282 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::DefAppTblPtr, CS\_Def\_App\_Table\_Entry\_t::Name, OS\_MAX\_API\_NAME, and CS\_Def\_App\_Table\_Entry\_t::State. Referenced by CS\_DisableNameAppCmd(), and CS\_EnableNameAppCmd().

**12.27.2.14 CS\_GetAppResTblEntryByName()** `bool CS_GetAppResTblEntryByName (`  
`CS_Res_App_Table_Entry_t ** EntryPtr,`  
`const char * Name )`

Gets a pointer to the results entry given an app name.

#### Description

This routine will look through the App Results table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>Name</i>	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	<i>EntryPtr</i>	A pointer to a <code>CS_Res_App_Table_Entry_t</code> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

#### Returns

Boolean name found response

#### Return values

<i>true</i>	Name was found in the table
<i>false</i>	Name was not found in the table

Definition at line 252 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_Res\_App\_Table\_Entry\_t::Name, OS\_MAX\_API\_NAME, CS\_AppData\_t::ResAppTblPtr, and CS\_Res\_App\_Table\_Entry\_t::State. Referenced by CS\_DisableNameAppCmd(), CS\_EnableNameAppCmd(), CS\_RecomputeBaselineAppCmd(), and CS\_ReportBaselineAppCmd().

**12.27.2.15 CS\_GetTableDefTblEntryByName()** `bool CS_GetTableDefTblEntryByName (`  
`CS_Def_Tables_Table_Entry_t ** EntryPtr,`  
`const char * Name )`

Gets a pointer to the definition entry given a table name.

#### Description

This routine will look through the Tables definition table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Name</i>	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	<i>EntryPtr</i>	A pointer to a <a href="#">CS_Def_Tables_Table_Entry_t</a> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

**Returns**

Boolean name found response

**Return values**

<i>true</i>	Name was found in the table
<i>false</i>	Name was not found in the table

Definition at line 221 of file cs\_utils.c.

References CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::DefTablesTblPtr, CS\_Def\_Tables\_Table\_Entry\_t::Name, and CS\_Def\_Tables\_Table\_Entry\_t::State.

Referenced by CS\_DisableNameTablesCmd(), and CS\_EnableNameTablesCmd().

```
12.27.2.16 CS_GetTableResTblEntryByName() bool CS_GetTableResTblEntryByName (
    CS_Res_Tables_Table_Entry_t ** EntryPtr,
    const char * Name )
```

Gets a pointer to the results entry given a table name.

**Description**

This routine will look through the Tables results table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Name</i>	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	<i>EntryPtr</i>	A pointer to a <a href="#">CS_Res_Tables_Table_Entry_t</a> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

**Returns**

Boolean name found response

**Return values**

<i>true</i>	Name was found in the table
<i>false</i>	Name was not found in the table

Definition at line 191 of file cs\_utils.c.

References CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_Res\_Tables\_Table\_Entry\_t::Name, CS\_AppData\_t::ResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::State.

Referenced by CS\_DisableNameTablesCmd(), CS\_EnableNameTablesCmd(), CS\_RecomputeBaselineTablesCmd(), and CS\_ReportBaselineTablesCmd().

**12.27.2.17 CS\_GoToNextTable()** `void CS_GoToNextTable (`  
 `void )`

Moves global variables to point to the next table.

**Description**

Moves the global variables to point to the next table to checksum

**Assumptions, External Events, and Notes:**

None

Definition at line 169 of file cs\_utils.c.

References CS\_AppData, CS\_NUM\_TABLES, CS\_HkPacket\_Payload\_t::CurrentCSTable, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::PassCounter, and CS\_HkPacket\_t::Payload.

Referenced by CS\_BackgroundApp(), CS\_BackgroundCfeCore(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), and CS\_BackgroundTables().

**12.27.2.18 CS\_HandleRoutineTableUpdates()** `CFE_Status_t CS_HandleRoutineTableUpdates (`  
 `void )`

Update All CS Tables.

**Description**

Updates all CS tables if no recompute is happening on that table.

This is called as part of the regular housekeeping cycle or at the wakeup interval if no housekeeping request is received.

**Assumptions, External Events, and Notes:**

None

**Returns**

Execution status, see [CFE Return Code Defines](#)

**Return values**

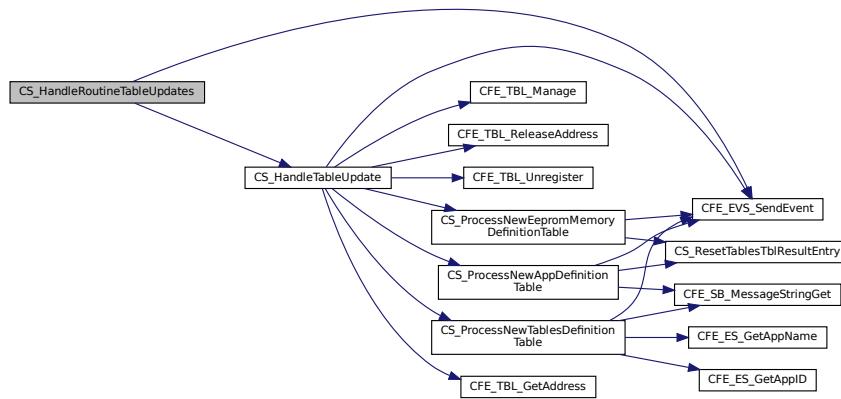
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 920 of file cs\_utils.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_AppData\_t::ChildTaskTable, CS\_APP\_TABLE, CS\_AppData, CS\_EEPROM\_TABLE, CS\_HandleTableUpdate(), CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_MAX\_NUM EEPROM\_TABLE\_ENTRIES, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_STATE\_DISABLED, CS\_TABLES\_TABLE, CS\_UPDATE\_APP\_ERR\_EID, CS\_UPDATE\_EEPROM\_ERR\_EID, CS\_UPDATE\_MEMORY\_ERR\_EID, CS\_UPDATE\_TABLES\_ERR\_EID, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::DefAppTblPtr, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::DefTablesTblPtr, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::ResAppTableHandle, CS\_AppData\_t::ResAppTblPtr, CS\_AppData\_t::ResEepromTableHandle, CS\_AppData\_t::ResEepromTblPtr, CS\_AppData\_t::ResMemoryTableHandle, CS\_AppData\_t::ResMemoryTblPtr, CS\_AppData\_t::ResTablesTableHandle, CS\_AppData\_t::ResTablesTblPtr, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_AppMain(), and CS\_AppPipe().

Here is the call graph for this function:



### 12.27.2.19 CS\_InitializeDefaultTables()

```
void CS_InitializeDefaultTables (
    void )
```

Initializes the default definition tables.

#### Description

Sets all of the entries in the default definitions tables for EEPROM, Memory, Tables, and Apps to zero and sets their states to 'empty'.

#### Assumptions, External Events, and Notes:

None

Definition at line 133 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_MAX\_NUM EEPROM\_TABLE\_ENTRIES, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::DefaultAppDefTable, CS\_AppData\_t::DefaultEepromDefTable, CS\_AppData\_t::DefaultMemoryDefTable, CS\_AppData\_t::DefaultTablesDefTable, CS\_Def\_Tables\_Table\_Entry\_t::Name, CS\_Def\_App\_Table\_Entry\_t::

Name, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Def\_Tables\_Table\_Entry\_t::State, and CS\_Def\_App\_Table\_Entry\_t::State.  
Referenced by CS\_AppInit().

### **12.27.2.20 CS\_ResetTablesTblResultEntry()** void CS\_ResetTablesTblResultEntry (     CS\_Res\_Tables\_Table\_Entry\_t \* TablesTblResultEntry )

Reset the checksum for a CS table entry in the CS tables table.

#### Description

If CS tables are listed in the CS tables table, then those tables must have their checksums recomputed when any of their entries have their enable/disable state flags modified.

This function will set ByteOffset and TempChecksumValue to zero, and ComputedYet to false for the specified CS tables table entry.

#### Assumptions, External Events, and Notes:

None

Definition at line 904 of file cs\_utils.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ComputeTables(), CS\_DisableEntryIDEEPROMCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEEPROMMemoryDefinitionTable(), CS\_RecomputeAppChildTask(), CS\_RecomputeEEPROMMemoryChildTask(), and CS\_RecomputeTablesChildTask().

### **12.27.2.21 CS\_VerifyCmdLength()** bool CS\_VerifyCmdLength (     const CFE\_MSG\_Message\_t \* msg,     size\_t ExpectedLength )

Verify command message length.

#### Description

This routine will check if the actual length of a software bus command message matches the expected length and send an error event message if a mismatch occurs

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>msg</i>	A <a href="#">CFE_MSG_Message_t</a> * pointer that references the software bus message. Calling function ensures this is non-null.
in	<i>ExpectedLength</i>	The expected length of the message based upon the command code

**Returns**

Boolean length valid response

**Return values**

<i>true</i>	Message length matches ExpectedLength
<i>false</i>	Message length ExpectedLength mismatch

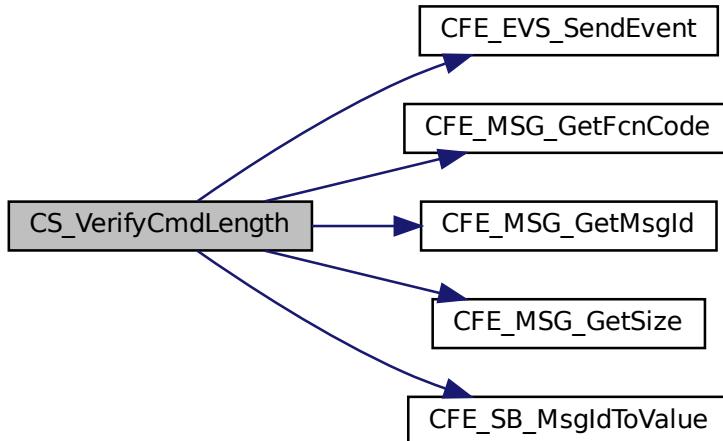
**See also**

[CS\\_LEN\\_ERR\\_EID](#)

Definition at line 447 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_MSG\_GetSize(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_MsgIdToValue(), CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_LEN\_ERR\_EID, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.27.2.22 CS\_ZeroAppTempValues()** void CS\_ZeroAppTempValues ( void )

Zeros out temporary checksum values of App table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 95 of file cs\_utils.c.

References CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_AppData\_t::ResAppTblPtr, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableAppCmd().

**12.27.2.23 CS\_ZeroCfeCoreTempValues()** `void CS_ZeroCfeCoreTempValues ( void )`

Zeros out temporary checksum values of the cFE Core.

**Description**

Zeros the TempChecksumValue and the byte offset for the cFE core. This allows the cFE core checksum to be started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 111 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData\_t::CfeCoreCodeSeg, CS\_AppData, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableCfeCoreCmd().

**12.27.2.24 CS\_ZeroEepromTempValues()** `void CS_ZeroEepromTempValues ( void )`

Zeros out temporary checksum values of EEPROM table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 47 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_AppData\_t::ResEepromTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableEepromCmd().

**12.27.2.25 CS\_ZeroMemoryTempValues()** `void CS_ZeroMemoryTempValues ( void )`

Zeros out temporary checksum values of Memory table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 63 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_AppData\_t::ResMemoryTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue. Referenced by CS\_DisableAllCSCmd(), and CS\_DisableMemoryCmd().

**12.27.2.26 CS\_ZeroOSTempValues()** void CS\_ZeroOSTempValues ( void )

Zeros out temporary checksum values of the OS code segment.

**Description**

Zeros the TempChecksumValue and the byte offset for the OS. This allows the OS checksum to be started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 122 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_AppData\_t::OSCodeSeg, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableOSCmd().

**12.27.2.27 CS\_ZeroTablesTempValues()** void CS\_ZeroTablesTempValues ( void )

Zeros out temporary checksum values of Tables table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 79 of file cs\_utils.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_AppData\_t::ResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableTablesCmd().

## 12.28 apps/cs/fsw/src/cs\_utils.h File Reference

```
#include "cfe.h"
#include "cs_tbldefs.h"
```

### Functions

- void [CS\\_ZeroEepromTempValues](#) (void)  
*Zeros out temporary checksum values of EEPROM table entries.*
- void [CS\\_ZeroMemoryTempValues](#) (void)

- `Zeros out temporary checksum values of Memory table entries.`
  - void [CS\\_ZeroTablesTempValues](#) (void)
- `Zeros out temporary checksum values of Tables table entries.`
  - void [CS\\_ZeroAppTempValues](#) (void)
- `Zeros out temporary checksum values of App table entries.`
  - void [CS\\_ZeroCfeCoreTempValues](#) (void)
- `Zeros out temporary checksum values of the cFE Core.`
  - void [CS\\_ZeroOSTempValues](#) (void)
- `Zeros out temporary checksum values of the OS code segment.`
  - void [CS\\_InitializeDefaultTables](#) (void)
    - Initializes the default definition tables.*
- void [CS\\_GoToNextTable](#) (void)
  - Moves global variables to point to the next table.*
- bool [CS\\_GetTableResTblEntryByName](#) (`CS_Res_Tables_Table_Entry_t` \*\*EntryPtr, const char \*Name)
  - Gets a pointer to the results entry given a table name.*
- bool [CS\\_GetTableDefTblEntryByName](#) (`CS_Def_Tables_Table_Entry_t` \*\*EntryPtr, const char \*Name)
  - Gets a pointer to the definition entry given a table name.*
- bool [CS\\_GetAppResTblEntryByName](#) (`CS_Res_App_Table_Entry_t` \*\*EntryPtr, const char \*Name)
  - Gets a pointer to the results entry given an app name.*
- bool [CS\\_GetAppDefTblEntryByName](#) (`CS_Def_App_Table_Entry_t` \*\*EntryPtr, const char \*Name)
  - Gets a pointer to the definition entry given an app name.*
- bool [CS\\_FindEnabledEepromEntry](#) (uint16 \*EnabledEntry)
  - Find an enabled EEPROM entry.*
- bool [CS\\_FindEnabledMemoryEntry](#) (uint16 \*EnabledEntry)
  - Find an enabled Memory entry.*
- bool [CS\\_FindEnabledTablesEntry](#) (uint16 \*EnabledEntry)
  - Find an enabled Tables entry.*
- bool [CS\\_FindEnabledAppEntry](#) (uint16 \*EnabledEntry)
  - Find an enabled App entry.*
- bool [CS\\_VerifyCmdLength](#) (const `CFE_MSG_Message_t` \*msg, size\_t ExpectedLength)
  - Verify command message length.*
- bool [CS\\_BackgroundOS](#) (void)
  - Compute a background check cycle on the OS.*
- bool [CS\\_BackgroundCfeCore](#) (void)
  - Compute a background check cycle on the cFE Core.*
- bool [CS\\_BackgroundEeprom](#) (void)
  - Compute a background check cycle on EEPROM.*
- bool [CS\\_BackgroundMemory](#) (void)
  - Compute a background check cycle on the Memory.*
- bool [CS\\_BackgroundTables](#) (void)
  - Compute a background check cycle on Tables.*
- bool [CS\\_BackgroundApp](#) (void)
  - Compute a background check cycle on Apps.*
- void [CS\\_ResetTablesTblResultEntry](#) (`CS_Res_Tables_Table_Entry_t` \*TablesTblResultEntry)
  - Reset the checksum for a CS table entry in the CS tables table.*
- `CFE_Status_t` [CS\\_HandleRoutineTableUpdates](#) (void)
  - Update All CS Tables.*

- `CFE_Status_t CS_AttemptTableReshare (CS_Res_Tables_Table_Entry_t *ResultsEntry, CFE_TBL_Handle_t *LocalTblHandle, CFE_TBL_Info_t *TblInfo, cpuaddr *LocalAddress, int32 *ResultGetInfo)`  
*Attempts to re-share a table.*
- bool `CS_CheckRecomputeOneshot (void)`

### 12.28.1 Detailed Description

Specification for the CFS utility functions

### 12.28.2 Function Documentation

**12.28.2.1 CS\_AttemptTableReshare()** `CFE_Status_t CS_AttemptTableReshare (`  
`CS_Res_Tables_Table_Entry_t * ResultsEntry,`  
`CFE_TBL_Handle_t * LocalTblHandle,`  
`CFE_TBL_Info_t * TblInfo,`  
`cpuaddr * LocalAddress,`  
`int32 * ResultGetInfo )`

Attempts to re-share a table.

#### Description

This function is called if the first attempt to share the table is unsuccessful. This function attempts to share the table again to see if it reappeared. Calling function ensures that parameters are non-null.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status, see [cFE Return Code Defines](#)

#### Return values

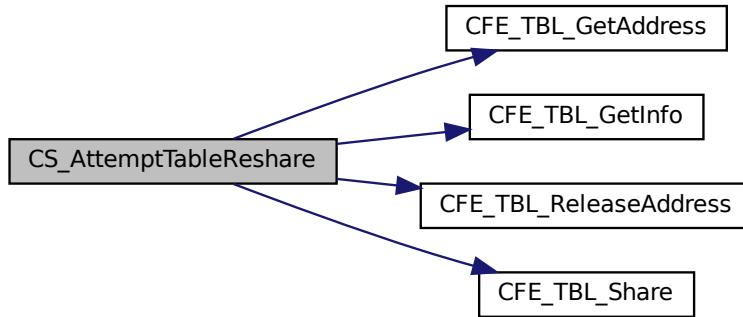
<code>CFE_SUCCESS</code>	Successful execution.
--------------------------	-----------------------

Definition at line 1004 of file cs\_utils.c.

References `CFE_SUCCESS`, `CFE_TBL_ERR_NEVER_LOADED`, `CFE_TBL_GetAddress()`, `CFE_TBL_GetInfo()`, `CFE_TBL_ReleaseAddress()`, `CFE_TBL_Share()`, `CS_Res_Tables_Table_Entry_t::Name`, and `CS_Res_Tables_Table_Entry_t::TblHandle`.

Referenced by `CS_ComputeTables()`.

Here is the call graph for this function:



**12.28.2.2 CS\_BackgroundApp()** `bool CS_BackgroundApp ( void )`

Compute a background check cycle on Apps.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

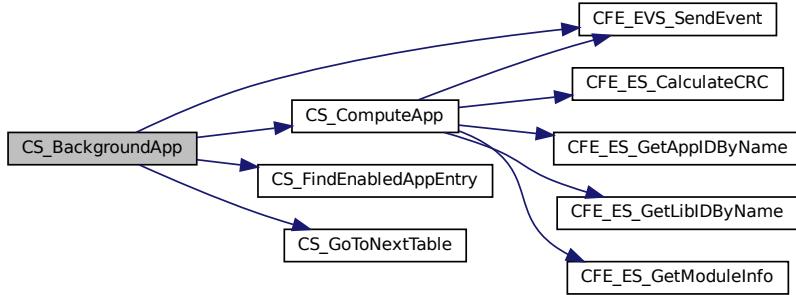
<i>true</i>	Checksum was done
<i>false</i>	Checksum was not done

Definition at line 830 of file cs\_utils.c.

References CS\_HkPacket\_Payload\_t::AppCSErrCounter, CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_Res\_App\_Table\_Entry\_t::ComparisonValue, CS\_APP\_MISCOMPARE\_ERR\_EID, CS\_AppData, CS\_COMPUTE\_APP\_NOT\_FOUND\_ERR\_EID, CS\_ComputeApp(), CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_FindEnabledAppEntry(), CS\_GoToNextTable(), CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_Res\_App\_Table\_Entry\_t::Name, CS\_HkPacket\_t::Payload, and CS\_AppData\_t::ResAppTblPtr.

Referenced by CS\_BackgroundCheckCycle().

Here is the call graph for this function:



**12.28.2.3 CS\_BackgroundCfeCore()** `bool CS_BackgroundCfeCore ( void )`

Compute a background check cycle on the cFE Core.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

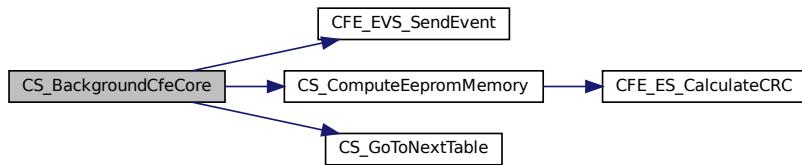
<code>true</code>	Checksum was done
<code>false</code>	Checksum was not done

Definition at line 477 of file cs\_utils.c.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CS_HkPacket_Payload_t::CfeCoreBaseline`, `CS_AppData_t::CfeCoreCodeSeg`, `CS_HkPacket_Payload_t::CfeCoreCSErrCounter`, `CS_HkPacket_Payload_t::CfeCoreCSState`, `CS_Res_EepromMemory_Table_Entry_t::ComparisonValue`, `CS_AppData`, `CS_CFECORE_MISC_OMPARE_ERR_EID`, `CS_ComputeEepromMemory()`, `CS_ERROR`, `CS_GoToNextTable()`, `CS_STATE_ENABLED`, `CS_HkPacket_Payload_t::CurrentEntryInTable`, `CS_AppData_t::HkPacket`, `CS_HkPacket_t::Payload`, and `CS_Res_EepromMemory_Table_Entry_t::State`.

Referenced by `CS_BackgroundCheckCycle()`.

Here is the call graph for this function:



**12.28.2.4 CS\_BackgroundEeprom()** `bool CS_BackgroundEeprom ( void )`

Compute a background check cycle on EEPROM.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

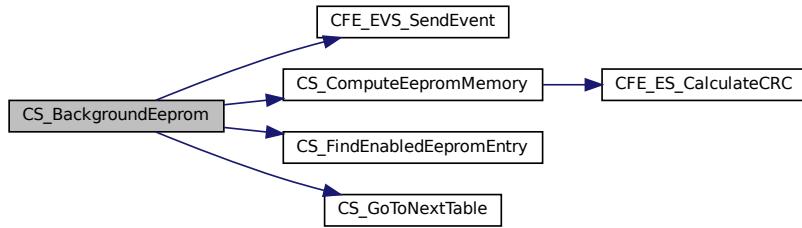
<code>true</code>	Checksum was done
<code>false</code>	Checksum was not done

Definition at line 609 of file `cs_utils.c`.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CS_Res_EepromMemory_Table_Entry_t::ComparisonValue`, `CS_AppData`, `CS_ComputeEepromMemory()`, `CS_EEPROM_MISCOMPARE_ERR_EID`, `CS_Error`, `CS_FindEnabledEepromEntry()`, `CS_GoToNextTable()`, `CS_MAX_NUM_EEPROM_TABLE_ENTRIES`, `CS_STATE_ENABLED`, `CS_HkPacket_Payload_t::CurrentEntryInTable`, `CS_HkPacket_Payload_t::EepromBaseline`, `CS_HkPacket_Payload_t::EepromCSErrCounter`, `CS_HkPacket_Payload_t::EepromCSState`, `CS_AppData_t::HkPacket`, `CS_HkPacket_t::Payload`, and `CS_AppData_t::ResEepromTblPtr`.

Referenced by `CS_BackgroundCheckCycle()`.

Here is the call graph for this function:



**12.28.2.5 CS\_BackgroundMemory()** `bool CS_BackgroundMemory ( void )`

Compute a background check cycle on the Memory.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

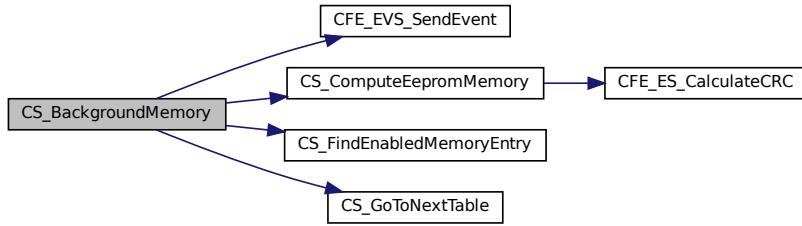
<i>true</i>	Checksum was done
<i>false</i>	Checksum was not done

Definition at line 684 of file cs\_utils.c.

References `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CS_Res_EepromMemory_Table_Entry_t::ComparisonValue`, `CS_AppData`, `CS_ComputeEepromMemory()`, `CS_ERROR`, `CS_FindEnabledMemoryEntry()`, `CS_GoToNextTable()`, `CS_MAX_NUM_MEMORY_TABLE_ENTRIES`, `CS_MEMORY_MISCOMPARE_ERR_EID`, `CS_STATE_ENABLED`, `CS_HkPacket_Payload_t::CurrentEntryInTable`, `CS_AppData_t::HkPacket`, `CS_HkPacket_Payload_t::MemoryCSErrCounter`, `CS_HkPacket_Payload_t::MemoryCSState`, `CS_HkPacket_t::Payload`, and `CS_AppData_t::ResMemoryTblPtr`.

Referenced by `CS_BackgroundCheckCycle()`.

Here is the call graph for this function:



**12.28.2.6 CS\_BackgroundOS()** `bool CS_BackgroundOS ( void )`

Compute a background check cycle on the OS.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

#### Return values

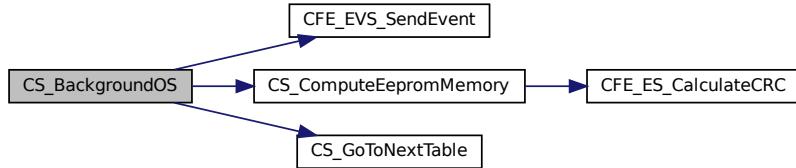
<code>true</code>	Checksum was done
<code>false</code>	Checksum was not done

Definition at line 544 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_Res\_EepromMemory\_Table\_Entry\_t::ComparisonValue, CS\_AppData, CS\_ComputeEepromMemory(), CS\_ERROR, CS\_GoToNextTable(), CS\_OS\_M::ISCOMPARE\_ERR\_EID, CS\_STATE\_ENABLED, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OSBaseline, CS\_AppData\_t::OSCodeSeg, CS\_HkPacket\_Payload\_t::OSCS::ErrCounter, CS\_HkPacket\_Payload\_t::OSCSSState, CS\_HkPacket\_t::Payload, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundCheckCycle().

Here is the call graph for this function:



**12.28.2.7 CS\_BackgroundTables()** `bool CS_BackgroundTables ( void )`

Compute a background check cycle on Tables.

#### Description

This routine will try and complete a cycle of background checking

#### Assumptions, External Events, and Notes:

None

#### Returns

Boolean checksum performed response

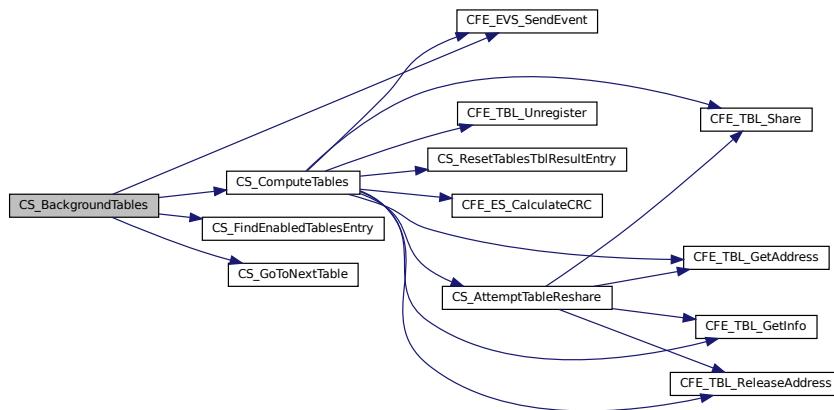
#### Return values

<code>true</code>	Checksum was done
<code>false</code>	Checksum was not done

Definition at line 753 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_Res\_Tables\_Table\_Entry\_t::ComparisonValue, CS\_AppData, CS\_COMPUTE\_TABLES\_NOT\_FOUND\_ERR\_EID, CS\_ComputeTables(), CS\_ERR\_NOT\_FOUND, CS\_ERROR, CS\_FindEnabledTablesEntry(), CS\_GoToNextTable(), CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_TABLES\_MISCOMPARE\_ERR\_EID, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_Res\_Tables\_Table\_Entry\_t::Name, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResTablesTblPtr, CS\_HkPacket\_Payload\_t::TablesCSErrCounter, and CS\_HkPacket\_Payload\_t::TablesCSState. Referenced by CS\_BackgroundCheckCycle().

Here is the call graph for this function:



### 12.28.2.8 CS\_CheckRecomputeOneshot()

```
bool CS_CheckRecomputeOneshot (
    void )
```

Definition at line 1032 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_CMD\_COMPUTE\_PROG\_ERR\_EID, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, and CS\_HkPacket\_Payload\_t::RecomputeInProgress.

Referenced by CS\_DisableAppCmd(), CS\_DisableEepromCmd(), CS\_DisableEntryIDEepromCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_DisableTablesCmd(), CS\_EnableAppCmd(), CS\_EnableEepromCmd(), CS\_EnableEntryIDEepromCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), and CS\_EnableTablesCmd().

Here is the call graph for this function:



### 12.28.2.9 CS\_FindEnabledAppEntry()

```
bool CS_FindEnabledAppEntry (
    uint16 * EnabledEntry )
```

Find an enabled App entry.

#### Description

This routine will look from the current position to the end of the table to find an enabled entry.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

**Returns**

Boolean enabled entry found response

**Return values**

<i>true</i>	Enabled entry was found in the table
<i>false</i>	Enabled entry was not found in the table

Definition at line 413 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_Payload<\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResAppTblPtr, and CS\_Res\_App\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundApp().

**12.28.2.10 CS\_FindEnabledEepromEntry()** `bool CS_FindEnabledEepromEntry (`  
`uint16 * EnabledEntry )`

Find an enabled EEPROM entry.

**Description**

This routine will look from the current position to the end of the table to find an enabled entry.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

**Returns**

Boolean entry found response

**Return values**

<i>true</i>	Entry was found in the table
<i>false</i>	Entry was not found in the table

Definition at line 312 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_↔ Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResEeprom↔ TblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundEeprom().

**12.28.2.11 CS\_FindEnabledMemoryEntry()** `bool CS_FindEnabledMemoryEntry (`  
`uint16 * EnabledEntry )`

Find an enabled Memory entry.

#### Description

This routine will look from the current position to the end of the table to find an enabled entry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

#### Returns

Boolean enabled entry found response

#### Return values

<i>true</i>	Enabled entry was found in the table
<i>false</i>	Enabled entry was not found in the table

Definition at line 346 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket\_↔ Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResMemory↔ TblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundMemory().

**12.28.2.12 CS\_FindEnabledTablesEntry()** `bool CS_FindEnabledTablesEntry (`  
`uint16 * EnabledEntry )`

Find an enabled Tables entry.

#### Description

This routine will look from the current position to the end of the table to find an enabled entry.

#### Assumptions, External Events, and Notes:

None

**Parameters**

in	<i>EnabledEntry</i>	A pointer to a uint16 that will be assigned an enabled entry ID, if one exists. Calling function ensures this is non-null.
out	*	EnabledEntry The ID of an enabled entry in the table, if the function returns true

**Returns**

Boolean enabled entry found response

**Return values**

<i>true</i>	Enabled entry was found in the table
<i>false</i>	Enabled entry was not found in the table

Definition at line 379 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_ENABLED, CS\_HkPacket ↪ Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_t::Payload, CS\_AppData\_t::ResTablesTbl ↪ Ptr, and CS\_Res\_Tables\_Table\_Entry\_t::State.

Referenced by CS\_BackgroundTables().

**12.28.2.13 CS\_GetAppDefTblEntryByName()** `bool CS_GetAppDefTblEntryByName (`

```
    CS_Def_App_Table_Entry_t ** EntryPtr,  
    const char * Name )
```

Gets a pointer to the definition entry given an app name.

**Description**

This routine will look through the App Definition table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Name</i>	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	<i>EntryPtr</i>	A pointer to a <a href="#">CS_Def_App_Table_Entry_t</a> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

**Returns**

Boolean name found response

**Return values**

<i>true</i>	Name was found in the table
<i>false</i>	Name was not found in the table

Definition at line 282 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::DefAppTblPtr, CS\_Def\_App\_Table\_Entry\_t::Name, OS\_MAX\_API\_NAME, and CS\_Def\_App\_Table\_Entry\_t::State. Referenced by CS\_DisableNameAppCmd(), and CS\_EnableNameAppCmd().

**12.28.2.14 CS\_GetAppResTblEntryByName()** `bool CS_GetAppResTblEntryByName (`  
`CS_Res_App_Table_Entry_t ** EntryPtr,`  
`const char * Name )`

Gets a pointer to the results entry given an app name.

#### Description

This routine will look through the App Results table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	Name	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	EntryPtr	A pointer to a <code>CS_Res_App_Table_Entry_t</code> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

#### Returns

Boolean name found response

#### Return values

<code>true</code>	Name was found in the table
<code>false</code>	Name was not found in the table

Definition at line 252 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_Res\_App\_Table\_Entry\_t::Name, OS\_MAX\_API\_NAME, CS\_AppData\_t::ResAppTblPtr, and CS\_Res\_App\_Table\_Entry\_t::State. Referenced by CS\_DisableNameAppCmd(), CS\_EnableNameAppCmd(), CS\_RecomputeBaselineAppCmd(), and CS\_ReportBaselineAppCmd().

**12.28.2.15 CS\_GetTableDefTblEntryByName()** `bool CS_GetTableDefTblEntryByName (`  
`CS_Def_Tables_Table_Entry_t ** EntryPtr,`  
`const char * Name )`

Gets a pointer to the definition entry given a table name.

#### Description

This routine will look through the Tables definition table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Name</i>	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	<i>EntryPtr</i>	A pointer to a <a href="#">CS_Def_Tables_Table_Entry_t</a> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

**Returns**

Boolean name found response

**Return values**

<i>true</i>	Name was found in the table
<i>false</i>	Name was not found in the table

Definition at line 221 of file cs\_utils.c.

References CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::DefTablesTblPtr, CS\_Def\_Tables\_Table\_Entry\_t::Name, and CS\_Def\_Tables\_Table\_Entry\_t::State.

Referenced by CS\_DisableNameTablesCmd(), and CS\_EnableNameTablesCmd().

```
12.28.2.16 CS_GetTableResTblEntryByName() bool CS_GetTableResTblEntryByName (
    CS_Res_Tables_Table_Entry_t ** EntryPtr,
    const char * Name )
```

Gets a pointer to the results entry given a table name.

**Description**

This routine will look through the Tables results table to find an entry that has the given name. It returns a pointer to the entry through a parameter.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Name</i>	The name associated with the entry we want to find. Calling function ensures this is non-null.
out	<i>EntryPtr</i>	A pointer to a <a href="#">CS_Res_Tables_Table_Entry_t</a> pointer that contains the start address of the entry whose name field matches the name passed in in the table passed in.

**Returns**

Boolean name found response

**Return values**

<i>true</i>	Name was found in the table
<i>false</i>	Name was not found in the table

Definition at line 191 of file cs\_utils.c.

References CFE\_TBL\_MAX\_FULL\_NAME\_LEN, CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_Res\_Tables\_Table\_Entry\_t::Name, CS\_AppData\_t::ResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::State.

Referenced by CS\_DisableNameTablesCmd(), CS\_EnableNameTablesCmd(), CS\_RecomputeBaselineTablesCmd(), and CS\_ReportBaselineTablesCmd().

**12.28.2.17 CS\_GoToNextTable()** `void CS_GoToNextTable (`  
 `void )`

Moves global variables to point to the next table.

**Description**

Moves the global variables to point to the next table to checksum

**Assumptions, External Events, and Notes:**

None

Definition at line 169 of file cs\_utils.c.

References CS\_AppData, CS\_NUM\_TABLES, CS\_HkPacket\_Payload\_t::CurrentCSTable, CS\_HkPacket\_Payload\_t::CurrentEntryInTable, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::PassCounter, and CS\_HkPacket\_t::Payload.

Referenced by CS\_BackgroundApp(), CS\_BackgroundCfeCore(), CS\_BackgroundEeprom(), CS\_BackgroundMemory(), CS\_BackgroundOS(), and CS\_BackgroundTables().

**12.28.2.18 CS\_HandleRoutineTableUpdates()** `CFE_Status_t CS_HandleRoutineTableUpdates (`  
 `void )`

Update All CS Tables.

**Description**

Updates all CS tables if no recompute is happening on that table.

This is called as part of the regular housekeeping cycle or at the wakeup interval if no housekeeping request is received.

**Assumptions, External Events, and Notes:**

None

**Returns**

Execution status, see [CFE Return Code Defines](#)

**Return values**

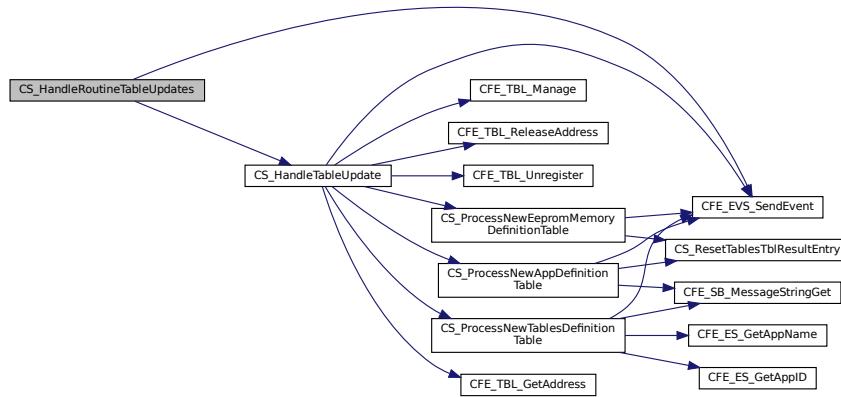
<a href="#">CFE_SUCCESS</a>	Successful execution.
-----------------------------	-----------------------

Definition at line 920 of file cs\_utils.c.

References CS\_HkPacket\_Payload\_t::AppCSState, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CS\_AppData\_t::ChildTaskTable, CS\_APP\_TABLE, CS\_AppData, CS\_EEPROM\_TABLE, CS\_HandleTableUpdate(), CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_MAX\_NUM EEPROM\_TABLE\_ENTRIES, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_MEMORY\_TABLE, CS\_STATE\_DISABLED, CS\_TABLES\_TABLE, CS\_UPDATE\_APP\_ERR\_EID, CS\_UPDATE\_EEPROM\_ERR\_EID, CS\_UPDATE\_MEMORY\_ERR\_EID, CS\_UPDATE\_TABLES\_ERR\_EID, CS\_AppData\_t::DefAppTableHandle, CS\_AppData\_t::DefAppTblPtr, CS\_AppData\_t::DefEepromTableHandle, CS\_AppData\_t::DefEepromTblPtr, CS\_AppData\_t::DefMemoryTableHandle, CS\_AppData\_t::DefMemoryTblPtr, CS\_AppData\_t::DefTablesTableHandle, CS\_AppData\_t::DefTablesTblPtr, CS\_HkPacket\_Payload\_t::EepromCSState, CS\_AppData\_t::HkPacket, CS\_HkPacket\_Payload\_t::MemoryCSState, CS\_HkPacket\_Payload\_t::OneShotInProgress, CS\_HkPacket\_t::Payload, CS\_HkPacket\_Payload\_t::RecomputeInProgress, CS\_AppData\_t::ResAppTableHandle, CS\_AppData\_t::ResAppTblPtr, CS\_AppData\_t::ResEepromTableHandle, CS\_AppData\_t::ResEepromTblPtr, CS\_AppData\_t::ResMemoryTableHandle, CS\_AppData\_t::ResMemoryTblPtr, CS\_AppData\_t::ResTablesTableHandle, CS\_AppData\_t::ResTablesTblPtr, and CS\_HkPacket\_Payload\_t::TablesCSState.

Referenced by CS\_AppMain(), and CS\_AppPipe().

Here is the call graph for this function:



### 12.28.2.19 CS\_InitializeDefaultTables()

```
void CS_InitializeDefaultTables (
    void )
```

Initializes the default definition tables.

#### Description

Sets all of the entries in the default definitions tables for EEPROM, Memory, Tables, and Apps to zero and sets their states to 'empty'.

#### Assumptions, External Events, and Notes:

None

Definition at line 133 of file cs\_utils.c.

References CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_MAX\_NUM EEPROM\_TABLE\_ENTRIES, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_STATE\_EMPTY, CS\_AppData\_t::DefaultAppDefTable, CS\_AppData\_t::DefaultEepromDefTable, CS\_AppData\_t::DefaultMemoryDefTable, CS\_AppData\_t::DefaultTablesDefTable, CS\_Def\_Tables\_Table\_Entry\_t::Name, CS\_Def\_App\_Table\_Entry\_t::

Name, CS\_Def\_EepromMemory\_Table\_Entry\_t::NumBytesToChecksum, CS\_Def\_EepromMemory\_Table\_Entry\_t::StartAddress, CS\_Def\_EepromMemory\_Table\_Entry\_t::State, CS\_Def\_Tables\_Table\_Entry\_t::State, and CS\_Def\_App\_Table\_Entry\_t::State.  
Referenced by CS\_AppInit().

### **12.28.2.20 CS\_ResetTablesTblResultEntry()** void CS\_ResetTablesTblResultEntry ( CS\_Res\_Tables\_Table\_Entry\_t \* TablesTblResultEntry )

Reset the checksum for a CS table entry in the CS tables table.

#### Description

If CS tables are listed in the CS tables table, then those tables must have their checksums recomputed when any of their entries have their enable/disable state flags modified.

This function will set ByteOffset and TempChecksumValue to zero, and ComputedYet to false for the specified CS tables table entry.

#### Assumptions, External Events, and Notes:

None

Definition at line 904 of file cs\_utils.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CS\_Res\_Tables\_Table\_Entry\_t::ComputedYet, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_ComputeTables(), CS\_DisableEntryIDEEPROMCmd(), CS\_DisableEntryIDMemoryCmd(), CS\_DisableNameAppCmd(), CS\_DisableNameTablesCmd(), CS\_EnableEntryIDEEPROMCmd(), CS\_EnableEntryIDMemoryCmd(), CS\_EnableNameAppCmd(), CS\_EnableNameTablesCmd(), CS\_ProcessNewAppDefinitionTable(), CS\_ProcessNewEEPROMMemoryDefinitionTable(), CS\_RecomputeAppChildTask(), CS\_RecomputeEEPROMMemoryChildTask(), and CS\_RecomputeTablesChildTask().

### **12.28.2.21 CS\_VerifyCmdLength()** bool CS\_VerifyCmdLength (

```
    const CFE_MSG_Message_t * msg,
    size_t ExpectedLength )
```

Verify command message length.

#### Description

This routine will check if the actual length of a software bus command message matches the expected length and send an error event message if a mismatch occurs

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>msg</i>	A <a href="#">CFE_MSG_Message_t</a> * pointer that references the software bus message. Calling function ensures this is non-null.
in	<i>ExpectedLength</i>	The expected length of the message based upon the command code

**Returns**

Boolean length valid response

**Return values**

<i>true</i>	Message length matches ExpectedLength
<i>false</i>	Message length ExpectedLength mismatch

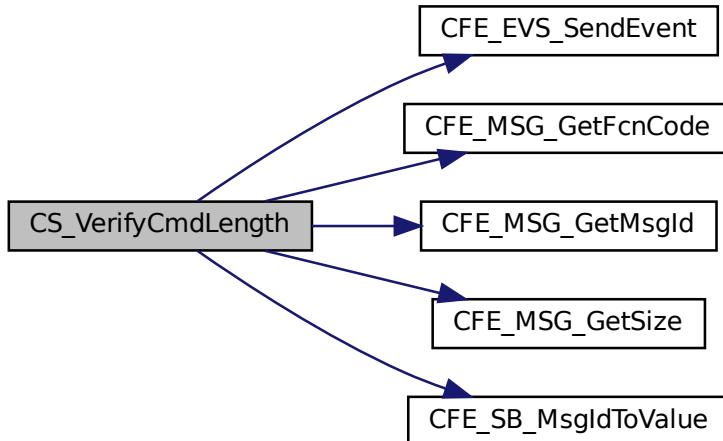
**See also**

[CS\\_LEN\\_ERR\\_EID](#)

Definition at line 447 of file cs\_utils.c.

References CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_MSG\_GetSize(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_MsgIdToValue(), CS\_HkPacket\_Payload\_t::CmdErrCounter, CS\_AppData, CS\_LEN\_ERR\_EID, CS\_AppData\_t::HkPacket, and CS\_HkPacket\_t::Payload. Referenced by CS\_ProcessCmd().

Here is the call graph for this function:



**12.28.2.22 CS\_ZeroAppTempValues()** void CS\_ZeroAppTempValues (void )

Zeros out temporary checksum values of App table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 95 of file cs\_utils.c.

References CS\_Res\_App\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, CS\_AppData\_t::ResAppTblPtr, and CS\_Res\_App\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableAppCmd().

**12.28.2.23 CS\_ZeroCfeCoreTempValues()** `void CS_ZeroCfeCoreTempValues ( void )`

Zeros out temporary checksum values of the cFE Core.

**Description**

Zeros the TempChecksumValue and the byte offset for the cFE core. This allows the cFE core checksum to be started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 111 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData\_t::CfeCoreCodeSeg, CS\_AppData, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableCfeCoreCmd().

**12.28.2.24 CS\_ZeroEepromTempValues()** `void CS_ZeroEepromTempValues ( void )`

Zeros out temporary checksum values of EEPROM table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 47 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, CS\_AppData\_t::ResEepromTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue.

Referenced by CS\_DisableAllCSCmd(), and CS\_DisableEepromCmd().

**12.28.2.25 CS\_ZeroMemoryTempValues()** `void CS_ZeroMemoryTempValues ( void )`

Zeros out temporary checksum values of Memory table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 63 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, CS\_AppData\_t::ResMemoryTblPtr, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue. Referenced by CS\_DisableAllCSCmd(), and CS\_DisableMemoryCmd().

**12.28.2.26 CS\_ZeroOSTempValues()** `void CS_ZeroOSTempValues ( void )`

Zeros out temporary checksum values of the OS code segment.

**Description**

Zeros the TempChecksumValue and the byte offset for the OS. This allows the OS checksum to be started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 122 of file cs\_utils.c.

References CS\_Res\_EepromMemory\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_AppData\_t::OSCodeSeg, and CS\_Res\_EepromMemory\_Table\_Entry\_t::TempChecksumValue. Referenced by CS\_DisableAllCSCmd(), and CS\_DisableOSCmd().

**12.28.2.27 CS\_ZeroTablesTempValues()** `void CS_ZeroTablesTempValues ( void )`

Zeros out temporary checksum values of Tables table entries.

**Description**

Zeros the TempChecksumValue and the byte offset for every entry in the table. This allows all entries in the table to have their checksum started 'fresh'.

**Assumptions, External Events, and Notes:**

None

Definition at line 79 of file cs\_utils.c.

References CS\_Res\_Tables\_Table\_Entry\_t::ByteOffset, CS\_AppData, CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, CS\_AppData\_t::ResTablesTblPtr, and CS\_Res\_Tables\_Table\_Entry\_t::TempChecksumValue. Referenced by CS\_DisableAllCSCmd(), and CS\_DisableTablesCmd().

**12.29 apps/cs/fsw/src/cs\_verify.h File Reference**

```
#include "cs_platform_cfg.h"
#include "cs_mission_cfg.h"
#include <stdint.h>
```

**12.29.1 Detailed Description**

Contains CFS Checksum macros that run preprocessor checks on mission configurable parameters

## 12.30 apps/cs/fsw/src/cs\_version.h File Reference

## Macros

- `#define CS_MAJOR_VERSION 2`  
*Major version number.*
  - `#define CS_MINOR_VERSION 5`  
*Minor version number.*
  - `#define CS_REVISION 99`  
*Revision number.*

### **12.30.1 Detailed Description**

Contains CFS Checksum macros that specify CS's version

## 12.31 apps/cs/fsw/tables/cs\_apptbl.c File Reference

```
#include "cfe.h"  
#include "cfe_tbl_filedef.h"  
#include "cs_msgdefs.h"  
#include "cs_platform_cfg.h"  
#include "cs_tbldefs.h"
```

## Variables

- `CS_Def_App_Table_Entry_t CS_AppTable [CS_MAX_NUM_APP_TABLE_ENTRIES]`

### **12.31.1 Detailed Description**

## The CFS Checksum (CS) Application Default Apps Table Definition

### **12.31.2 Variable Documentation**

**12.31.2.1 CS\_AppTable** `CS_Def_App_Table_Entry_t` `CS_AppTable[CS_MAX_NUM_APP_TABLE_ENTRIES]`

### Initial value:

```
    {.State = CS_STATE_EMPTY, .Name = ""}}
```

Definition at line 34 of file cs\_apptbl.c.

## 12.32 apps/cs/fsw/tables/cs\_eepromtbl.c File Reference

```
#include "cfe.h"
#include "cfe_tbl_filedef.h"
#include "cs_msgdefs.h"
#include "cs_platform_cfg.h"
#include "cs_tbldefs.h"
```

### Variables

- [CS\\_Def\\_EepromMemory\\_Table\\_Entry\\_t CS\\_EepromTable \[CS\\_MAX\\_NUM\\_EEPROM\\_TABLE\\_ENTRIES\]](#)

#### 12.32.1 Detailed Description

The CFS Checksum (CS) Application Default EEPROM Table Definition

#### 12.32.2 Variable Documentation

**12.32.2.1 CS\_EepromTable** [CS\\_Def\\_EepromMemory\\_Table\\_Entry\\_t CS\\_EepromTable \[CS\\_MAX\\_NUM\\_EEPROM\\_TABLE\\_ENTRIES\]](#)  
Definition at line 34 of file cs\_eepromtbl.c.

## 12.33 apps/cs/fsw/tables/cs\_memorytbl.c File Reference

```
#include "cfe.h"
#include "cfe_tbl_filedef.h"
#include "cs_msgdefs.h"
#include "cs_platform_cfg.h"
#include "cs_tbldefs.h"
```

### Variables

- [CS\\_Def\\_EepromMemory\\_Table\\_Entry\\_t CS\\_MemoryTable \[CS\\_MAX\\_NUM\\_MEMORY\\_TABLE\\_ENTRIES\]](#)

#### 12.33.1 Detailed Description

The CFS Checksum (CS) Application Default Memory Table Definition

#### 12.33.2 Variable Documentation

**12.33.2.1 CS\_MemoryTable** [CS\\_Def\\_EepromMemory\\_Table\\_Entry\\_t CS\\_MemoryTable \[CS\\_MAX\\_NUM\\_MEMORY\\_TABLE\\_ENTRIES\]](#)  
Definition at line 34 of file cs\_memorytbl.c.

## 12.34 apps/cs/fsw/tables/cs\_tablestbl.c File Reference

```
#include "cfe.h"
#include "cfe_tbl_filedef.h"
#include "cs_msgdefs.h"
#include "cs_platform_cfg.h"
#include "cs_tbldefs.h"
```

## Variables

- `CS_DefTables_Table_Entry_t CS_TablesTable[CS_MAX_NUM_TABLES_TABLE_ENTRIES]`

### **12.34.1 Detailed Description**

## The CFS Checksum (CS) Application Default Tables Table Definition

## 12.34.2 Variable Documentation

**12.34.2.1 CS\_TablesTable** **CS\_DefTables\_Table\_Entry\_t** CS\_TablesTable[CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES]

### Initial value:

= {

Definition at line 34 of file cs\_tablestbl.c.

## 12.35 buildosal\_public\_apiincosconfig.h File Reference

## Macros

- `#define OSAL_CONFIG_INCLUDE_DYNAMIC_LOADER`  
*Configuration file Operating System Abstraction Layer.*
  - `#define OSAL_CONFIG_INCLUDE_NETWORK`
  - `#define OSAL_CONFIG_INCLUDE_STATIC_LOADER`
  - `#define OSAL_CONFIG_CONSOLE_ASYNC`
  - `#define OS_MAX_TASKS 64`  
*The maximum number of tasks to support.*
  - `#define OS_MAX_QUEUES 64`

- `#define OS_MAX_COUNT_SEMAPHORES 20`  
*The maximum number of queues to support.*
- `#define OS_MAX_BIN_SEMAPHORES 20`  
*The maximum number of counting semaphores to support.*
- `#define OS_MAX_MUTEXES 20`  
*The maximum number of binary semaphores to support.*
- `#define OS_MAX_CONDVARs 4`  
*The maximum number of mutexes to support.*
- `#define OS_MAX_MODULES 20`  
*The maximum number of condition variables to support.*
- `#define OS_MAX_TIMEBASES 5`  
*The maximum number of modules to support.*
- `#define OS_MAX_TIMERS 10`  
*The maximum number of timebases to support.*
- `#define OS_MAX_NUM_OPEN_FILES 50`  
*The maximum number of timer callbacks to support.*
- `#define OS_MAX_NUM_OPEN_DIRS 4`  
*The maximum number of concurrently open files to support.*
- `#define OS_MAX_FILE_SYSTEMS 14`  
*The maximum number of concurrently open directories to support.*
- `#define OS_MAX_SYM_LEN 64`  
*The maximum number of file systems to support.*
- `#define OS_MAX_FILE_NAME 20`  
*The maximum length of symbols.*
- `#define OS_MAX_PATH_LEN 64`  
*The maximum length of OSAL file names.*
- `#define OS_MAX_API_NAME 20`  
*The maximum length of OSAL path names.*
- `#define OS_SOCKADDR_MAX_LEN 28`  
*The maximum length of OSAL resource names.*
- `#define OS_BUFFER_SIZE 172`  
*The maximum size of the socket address structure.*
- `#define OS_BUFFER_MSG_DEPTH 100`  
*The maximum size of output produced by a single `OS_printf()`.*
- `#define OS.UtilityTask_PRIORITY 245`  
*Priority level of the background utility task.*
- `#define OS.UtilityTask_STACK_SIZE 2048`  
*The stack size of the background utility task.*
- `#define OS_MAX_CMD_LEN 1000`  
*The maximum size of a shell command.*
- `#define OS_QUEUE_MAX_DEPTH 50`  
*The maximum depth of OSAL queues.*
- `#define OS_Shell_Cmd_Input_File_Name ""`  
*The name of the temporary file used to store shell commands.*
- `#define OS_Printf_Console_Name ""`  
*The name of the primary console device.*

- `#define OS_ADD_TASK_FLAGS 0`  
*Flags added to all tasks on creation.*
- `#define OS_MAX_CONSOLES 1`  
*The maximum number of console devices to support.*
- `#define OS_MODULE_FILE_EXTENSION ".so"`  
*The system-specific file extension used on loadable module files.*
- `#define OS_FS_DEV_NAME_LEN 32`
- `#define OS_FS_PHYS_NAME_LEN 64`
- `#define OS_FS_VOL_NAME_LEN 32`

### 12.35.1 Macro Definition Documentation

#### 12.35.1.1 OS\_ADD\_TASK\_FLAGS `#define OS_ADD_TASK_FLAGS 0`

Flags added to all tasks on creation.

Added to the task flags on creation

Supports adding floating point support for all tasks when the OS requires it

Definition at line 254 of file osconfig.h.

#### 12.35.1.2 OS\_BUFFER\_MSG\_DEPTH `#define OS_BUFFER_MSG_DEPTH 100`

The maximum number of `OS_printf()` output strings to buffer.

Based on the OSAL\_CONFIG\_PRINTF\_BUFFER\_DEPTH configuration option

Definition at line 187 of file osconfig.h.

#### 12.35.1.3 OS\_BUFFER\_SIZE `#define OS_BUFFER_SIZE 172`

The maximum size of output produced by a single `OS_printf()`

Based on the OSAL\_CONFIG\_PRINTF\_BUFFER\_SIZE configuration option

Definition at line 180 of file osconfig.h.

#### 12.35.1.4 OS\_FS\_DEV\_NAME\_LEN `#define OS_FS_DEV_NAME_LEN 32`

Device name length

Definition at line 281 of file osconfig.h.

#### 12.35.1.5 OS\_FS\_PHYS\_NAME\_LEN `#define OS_FS_PHYS_NAME_LEN 64`

Physical drive name length

Definition at line 282 of file osconfig.h.

#### 12.35.1.6 OS\_FS\_VOL\_NAME\_LEN `#define OS_FS_VOL_NAME_LEN 32`

Volume name length

Definition at line 283 of file osconfig.h.

**12.35.1.7 OS\_MAX\_API\_NAME** #define OS\_MAX\_API\_NAME 20

The maximum length of OSAL resource names.

Based on the OSAL\_CONFIG\_MAX\_API\_NAME configuration option

**Note**

This value must include a terminating NUL character

Definition at line 163 of file osconfig.h.

**12.35.1.8 OS\_MAX\_BIN\_SEMAPHORES** #define OS\_MAX\_BIN\_SEMAPHORES 20

The maximum number of binary semaphores to support.

Based on the OSAL\_CONFIG\_MAX\_BIN\_SEMAPHORES configuration option

Definition at line 65 of file osconfig.h.

**12.35.1.9 OS\_MAX\_CMD\_LEN** #define OS\_MAX\_CMD\_LEN 1000

The maximum size of a shell command.

This limit is only applicable if shell support is enabled.

Based on the OSAL\_CONFIG\_MAX\_CMD\_LEN configuration option

**Note**

This value must include a terminating NUL character

Definition at line 218 of file osconfig.h.

**12.35.1.10 OS\_MAX\_CONDVARS** #define OS\_MAX\_CONDVARS 4

The maximum number of condition variables to support.

Based on the OSAL\_CONFIG\_MAX\_CONDVARS configuration option

Definition at line 79 of file osconfig.h.

**12.35.1.11 OS\_MAX\_CONSOLES** #define OS\_MAX\_CONSOLES 1

The maximum number of console devices to support.

Fixed value based on current OSAL implementation, not user configurable.

Definition at line 269 of file osconfig.h.

**12.35.1.12 OS\_MAX\_COUNT\_SEMAPHORES** #define OS\_MAX\_COUNT\_SEMAPHORES 20

The maximum number of counting semaphores to support.

Based on the OSAL\_CONFIG\_MAX\_COUNT\_SEMAPHORES configuration option

Definition at line 58 of file osconfig.h.

**12.35.1.13 OS\_MAX\_FILE\_NAME** #define OS\_MAX\_FILE\_NAME 20

The maximum length of OSAL file names.

This limit applies specifically to the file name portion, not the directory portion, of a path name.

Based on the OSAL\_CONFIG\_MAX\_FILE\_NAME configuration option

**Note**

This value must include a terminating NUL character

Definition at line 142 of file osconfig.h.

**12.35.1.14 OS\_MAX\_FILE\_SYSTEMS** #define OS\_MAX\_FILE\_SYSTEMS 14  
The maximum number of file systems to support.  
Based on the OSAL\_CONFIG\_MAX\_FILE\_SYSTEMS configuration option  
Definition at line 121 of file osconfig.h.

**12.35.1.15 OS\_MAX\_MODULES** #define OS\_MAX\_MODULES 20  
The maximum number of modules to support.  
Based on the OSAL\_CONFIG\_MAX\_MODULES configuration option  
Definition at line 86 of file osconfig.h.

**12.35.1.16 OS\_MAX\_MUTEXES** #define OS\_MAX\_MUTEXES 20  
The maximum number of mutexes to support.  
Based on the OSAL\_CONFIG\_MAX\_MUTEXES configuration option  
Definition at line 72 of file osconfig.h.

**12.35.1.17 OS\_MAX\_NUM\_OPEN\_DIRS** #define OS\_MAX\_NUM\_OPEN\_DIRS 4  
The maximum number of concurrently open directories to support.  
Based on the OSAL\_CONFIG\_MAX\_NUM\_OPEN\_DIRS configuration option  
Definition at line 114 of file osconfig.h.

**12.35.1.18 OS\_MAX\_NUM\_OPEN\_FILES** #define OS\_MAX\_NUM\_OPEN\_FILES 50  
The maximum number of concurrently open files to support.  
Based on the OSAL\_CONFIG\_MAX\_NUM\_OPEN\_FILES configuration option  
Definition at line 107 of file osconfig.h.

**12.35.1.19 OS\_MAX\_PATH\_LEN** #define OS\_MAX\_PATH\_LEN 64  
The maximum length of OSAL path names.  
This limit applies to the overall length of a path name, including the file name and directory portions.  
Based on the OSAL\_CONFIG\_MAX\_PATH\_LEN configuration option

**Note**

This value must include a terminating NUL character

Definition at line 154 of file osconfig.h.

**12.35.1.20 OS\_MAX\_QUEUES** #define OS\_MAX\_QUEUES 64  
The maximum number of queues to support.  
Based on the OSAL\_CONFIG\_MAX\_QUEUES configuration option  
Definition at line 51 of file osconfig.h.

**12.35.1.21 OS\_MAX\_SYM\_LEN** #define OS\_MAX\_SYM\_LEN 64  
The maximum length of symbols.  
Based on the OSAL\_CONFIG\_MAX\_SYM\_LEN configuration option

**Note**

This value must include a terminating NUL character

Definition at line 130 of file osconfig.h.

**12.35.1.22 OS\_MAX\_TASKS #define OS\_MAX\_TASKS 64**

The maximum number of tasks to support.

Based on the OSAL\_CONFIG\_MAX\_TASKS configuration option

Definition at line 44 of file osconfig.h.

**12.35.1.23 OS\_MAX\_TIMEBASES #define OS\_MAX\_TIMEBASES 5**

The maximum number of timebases to support.

Based on the OSAL\_CONFIG\_MAX\_TIMEBASES configuration option

Definition at line 93 of file osconfig.h.

**12.35.1.24 OS\_MAX\_TIMERS #define OS\_MAX\_TIMERS 10**

The maximum number of timer callbacks to support.

Based on the OSAL\_CONFIG\_MAX\_TIMERS configuration option

Definition at line 100 of file osconfig.h.

**12.35.1.25 OS\_MODULE\_FILE\_EXTENSION #define OS\_MODULE\_FILE\_EXTENSION ".so"**

The system-specific file extension used on loadable module files.

Fixed value based on system selection, not user configurable.

Definition at line 276 of file osconfig.h.

**12.35.1.26 OS\_PRINTF\_CONSOLE\_NAME #define OS\_PRINTF\_CONSOLE\_NAME ""**

The name of the primary console device.

This is the device to which [OS\\_printf\(\)](#) output is written. The output may be configured to tag each line with this prefix for identification.

Based on the OSAL\_CONFIG\_PRINTF\_CONSOLE\_NAME configuration option

Definition at line 245 of file osconfig.h.

**12.35.1.27 OS\_QUEUE\_MAX\_DEPTH #define OS\_QUEUE\_MAX\_DEPTH 50**

The maximum depth of OSAL queues.

Based on the OSAL\_CONFIG\_QUEUE\_MAX\_DEPTH configuration option

Definition at line 225 of file osconfig.h.

**12.35.1.28 OS\_SHELL\_CMD\_INPUT\_FILE\_NAME #define OS\_SHELL\_CMD\_INPUT\_FILE\_NAME ""**

The name of the temporary file used to store shell commands.

This configuration is only applicable if shell support is enabled, and only necessary/relevant on some OS implementations.

Based on the OSAL\_CONFIG\_SHELL\_CMD\_INPUT\_FILE\_NAME configuration option

Definition at line 235 of file osconfig.h.

**12.35.1.29 OS SOCKADDR\_MAX\_LEN** #define OS SOCKADDR\_MAX\_LEN 28

The maximum size of the socket address structure.

This is part of the Socket API, and should be set large enough to hold the largest address type in use on the target system.

Based on the OSAL\_CONFIG\_SOCKADDR\_MAX\_LEN configuration option

Definition at line 173 of file osconfig.h.

**12.35.1.30 OS UTILITYTASK\_PRIORITY** #define OS UTILITYTASK\_PRIORITY 245

Priority level of the background utility task.

This task is responsible for writing buffered output of OS\_printf to the actual console device, and any other future maintenance task.

Based on the OSAL\_CONFIG\_UTILITYTASK\_PRIORITY configuration option

Definition at line 197 of file osconfig.h.

**12.35.1.31 OS UTILITYTASK\_STACK\_SIZE** #define OS UTILITYTASK\_STACK\_SIZE 2048

The stack size of the background utility task.

This task is responsible for writing buffered output of OS\_printf to the actual console device, and any other future maintenance task.

Based on the OSAL\_CONFIG\_UTILITYTASK\_STACK\_SIZE configuration option

Definition at line 207 of file osconfig.h.

**12.35.1.32 OSAL\_CONFIG\_CONSOLE\_ASYNC** #define OSAL\_CONFIG\_CONSOLE\_ASYNC

Definition at line 27 of file osconfig.h.

**12.35.1.33 OSAL\_CONFIG\_INCLUDE\_DYNAMIC\_LOADER** #define OSAL\_CONFIG\_INCLUDE\_DYNAMIC\_LOADER

Configuration file Operating System Abstraction Layer.

The specific definitions in this file may only be modified by setting the respective OSAL configuration options in the CMake build.

Any direct modifications to the generated copy will be overwritten each time CMake executes.

**Note**

This file was automatically generated by CMake from /home/runner/work/CS/CS/osal/default\_config.cmake

Definition at line 21 of file osconfig.h.

**12.35.1.34 OSAL\_CONFIG\_INCLUDE\_NETWORK** #define OSAL\_CONFIG\_INCLUDE\_NETWORK

Definition at line 22 of file osconfig.h.

**12.35.1.35 OSAL\_CONFIG\_INCLUDE\_STATIC\_LOADER** #define OSAL\_CONFIG\_INCLUDE\_STATIC\_LOADER

Definition at line 23 of file osconfig.h.

**12.36 cfe/cmake/sample\_defs/example\_mission\_cfg.h File Reference****Macros**

- #define CFE\_MISSION\_MAX\_PATH\_LEN 64
- #define CFE\_MISSION\_MAX\_FILE\_LEN 20

- #define CFE\_MISSION\_MAX\_API\_LEN 20
- #define CFE\_MISSION\_MAX\_NUM\_FILES 50
- #define CFE\_MISSION\_ES\_MAX\_APPLICATIONS 16
- #define CFE\_MISSION\_ES\_PERF\_MAX\_IDS 128
- #define CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS 17
- #define CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH 16
- #define CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_ES\_CrcType\_CRC\_16
- #define CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN (CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH + CFE\_MISSION\_MAX\_API\_LEN + 4)

#### Checksum/CRC algorithm identifiers

- #define CFE\_MISSION\_ES\_CRC\_8 CFE\_ES\_CrcType\_CRC\_8 /\* 1 \*/
- #define CFE\_MISSION\_ES\_CRC\_16 CFE\_ES\_CrcType\_CRC\_16 /\* 2 \*/
- #define CFE\_MISSION\_ES\_CRC\_32 CFE\_ES\_CrcType\_CRC\_32 /\* 3 \*/
- #define CFE\_MISSION\_ES\_MAX\_MESSAGE\_LENGTH 122
- #define CFE\_FS\_HDR\_DESC\_MAX\_LEN 32
  - Max length of description field in a standard cFE File Header.*
- #define CFE\_FS\_FILE\_CONTENT\_ID 0x63464531
  - Magic Number for cFE compliant files (= 'cFE1')*
- #define CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE 32768
- #define CFE\_MISSION\_SB\_MAX\_PIPES 64
- #define CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH 16
- #define CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN (CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH + CFE\_MISSION\_MAX\_API\_LEN + 4)
- #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI true
- #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC false
- #define CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE true
- #define CFE\_MISSION\_TIME\_AT\_TONE\_WAS true
- #define CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE false
- #define CFE\_MISSION\_TIME\_MIN\_ELAPSED 0
- #define CFE\_MISSION\_TIME\_MAX\_ELAPSED 200000
- #define CFE\_MISSION\_TIME\_DEF\_MET\_SECS 1000
- #define CFE\_MISSION\_TIME\_DEF\_MET\_SUBS 0
- #define CFE\_MISSION\_TIME\_DEF\_STCF\_SECS 1000000
- #define CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS 0
- #define CFE\_MISSION\_TIME\_DEF\_LEAPS 37
- #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS 0
- #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS 1000
- #define CFE\_MISSION\_TIME\_EPOCH\_YEAR 1980
- #define CFE\_MISSION\_TIME\_EPOCH\_DAY 1
- #define CFE\_MISSION\_TIME\_EPOCH\_HOUR 0
- #define CFE\_MISSION\_TIME\_EPOCH\_MINUTE 0
- #define CFE\_MISSION\_TIME\_EPOCH\_SECOND 0
- #define CFE\_MISSION\_TIME\_EPOCH\_MICROS 0
- #define CFE\_MISSION\_TIME\_FS\_FACTOR 789004800

##### 12.36.1 Detailed Description

This header file contains the mission configuration parameters and typedefs with mission scope.

This provides values for configurable items that affect the interface(s) of this module. This includes the CMD/T $\leftarrow$ LM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

**Note**

It is no longer necessary to provide this file directly in the defs directory, but if present, this file is still supported/usable for backward compatibility. To use this file, it should be called "cfe\_mission\_cfg.h".

Going forward, more fine-grained (module/purposes-specific) header files are included with each submodule. These may be overridden as necessary, but only if a definition within that file needs to be changed from the default. This approach will reduce the amount of duplicate/cloned definitions and better support alternative build configurations in the future.

Note that if this file is present, the fine-grained header files noted above will *not* be used.

### 12.36.2 Macro Definition Documentation

#### 12.36.2.1 CFE\_FS\_FILE\_CONTENT\_ID #define CFE\_FS\_FILE\_CONTENT\_ID 0x63464531

Magic Number for cFE compliant files (= 'cFE1')

Definition at line 313 of file example\_mission\_cfg.h.

#### 12.36.2.2 CFE\_FS\_HDR\_DESC\_MAX\_LEN #define CFE\_FS\_HDR\_DESC\_MAX\_LEN 32

Max length of description field in a standard cFE File Header.

Definition at line 311 of file example\_mission\_cfg.h.

#### 12.36.2.3 CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN #define CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN (CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH + CFE\_MISSION\_MAX\_API\_LEN + 4)

**Purpose** Maximum Length of Full CDS Name in messages

**Description:**

Indicates the maximum length (in characters) of the entire CDS name of the following form: "ApplicationName.CDSName"

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 262 of file example\_mission\_cfg.h.

#### 12.36.2.4 CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH #define CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH 16

**Purpose** Maximum Length of CDS Name

**Description:**

Indicates the maximum length (in characters) of the CDS name ('CDSName') portion of a Full CDS Name of the following form: "ApplicationName.CDSName"

This length does not need to include an extra character for NULL termination.

**Limits**

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 228 of file example\_mission\_cfg.h.

**12.36.2.5 CFE\_MISSION\_ES\_CRC\_16** #define CFE\_MISSION\_ES\_CRC\_16 CFE\_ES\_CrcType\_CRC\_16 /\* 2 \*/  
Definition at line 270 of file example\_mission\_cfg.h.

**12.36.2.6 CFE\_MISSION\_ES\_CRC\_32** #define CFE\_MISSION\_ES\_CRC\_32 CFE\_ES\_CrcType\_CRC\_32 /\* 3 \*/  
Definition at line 271 of file example\_mission\_cfg.h.

**12.36.2.7 CFE\_MISSION\_ES\_CRC\_8** #define CFE\_MISSION\_ES\_CRC\_8 CFE\_ES\_CrcType\_CRC\_8 /\* 1 \*/  
Definition at line 269 of file example\_mission\_cfg.h.

**12.36.2.8 CFE\_MISSION\_ES\_DEFAULT\_CRC** #define CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_ES\_CrcType\_CRC\_16

**Purpose** Mission Default CRC algorithm

**Description:**

Indicates the which CRC algorithm should be used as the default for verifying the contents of Critical Data Stores and when calculating Table Image data integrity values.

**Limits**

Currently only CFE\_ES\_CrcType\_CRC\_16 is supported (see brief in CFE\_ES\_CrcType\_Enum definition in [cfe\\_es\\_api\\_typedefs.h](#))

Definition at line 242 of file example\_mission\_cfg.h.

**12.36.2.9 CFE\_MISSION\_ES\_MAX\_APPLICATIONS** #define CFE\_MISSION\_ES\_MAX\_APPLICATIONS 16

**Purpose** Mission Max Apps in a message

**Description:**

Indicates the maximum number of apps in a telemetry housekeeping message

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 173 of file example\_mission\_cfg.h.

**12.36.2.10 CFE\_MISSION\_ES\_PERF\_MAX\_IDS** #define CFE\_MISSION\_ES\_PERF\_MAX\_IDS 128

**Purpose** Define Max Number of Performance IDs for messages

Description:

Defines the maximum number of perf ids allowed in command/telemetry messages

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 190 of file example\_mission\_cfg.h.

**12.36.2.11 CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS** #define CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS 17

**Purpose** Maximum number of block sizes in pool structures

Description:

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS. This definition is used as the array size with the pool stats structure, and therefore should be consistent across all CPUs in a mission, as well as with the ground station.

There is also a platform-specific limit which may be fewer than this value.

Limits:

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

Definition at line 211 of file example\_mission\_cfg.h.

**12.36.2.12 CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH** #define CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH 122

**Purpose** Maximum Event Message Length

Description:

Indicates the maximum length (in characters) of the formatted text string portion of an event message

This length does not need to include an extra character for NULL termination.

Limits

Not Applicable

Definition at line 297 of file example\_mission\_cfg.h.

**12.36.2.13 CFE\_MISSION\_MAX\_API\_LEN** #define CFE\_MISSION\_MAX\_API\_LEN 20

**Purpose** cFE Maximum length for API names within data exchange structures

**Description:**

The value of this constant dictates the size of filenames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_API\_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_API\_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_API\_LEN value.

This length must include an extra character for NULL termination.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 125 of file example\_mission\_cfg.h.

**12.36.2.14 CFE\_MISSION\_MAX\_FILE\_LEN** #define CFE\_MISSION\_MAX\_FILE\_LEN 20

**Purpose** cFE Maximum length for filenames within data exchange structures

**Description:**

The value of this constant dictates the size of filenames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_FILE\_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_FILE\_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_FILE\_LEN value.

This length must include an extra character for NULL termination.

**Limits**

All CPUs within the same SB domain (mission) and ground tools must share the same definition. Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 99 of file example\_mission\_cfg.h.

**12.36.2.15 CFE\_MISSION\_MAX\_NUM\_FILES** #define CFE\_MISSION\_MAX\_NUM\_FILES 50

**Purpose** cFE Maximum number of files in a message/data exchange

**Description:**

The value of this constant dictates the maximum number of files within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_NUM\_OPEN\_FILES but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_NUM\_OPEN\_FILES in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_NUM\_OPEN\_FILES value.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 147 of file example\_mission\_cfg.h.

**12.36.2.16 CFE\_MISSION\_MAX\_PATH\_LEN #define CFE\_MISSION\_MAX\_PATH\_LEN 64****Purpose** cFE Maximum length for pathnames within data exchange structures**Description:**

The value of this constant dictates the size of pathnames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_PATH\_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_PATH\_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_PATH\_LEN value.

This length must include an extra character for NULL termination.

**Limits**

All CPUs within the same SB domain (mission) and ground tools must share the same definition. Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 72 of file example\_mission\_cfg.h.

**12.36.2.17 CFE\_MISSION\_SB\_MAX\_PIPES #define CFE\_MISSION\_SB\_MAX\_PIPES 64****Purpose** Maximum Number of pipes that SB command/telemetry messages may hold**Description:**

Dictates the maximum number of unique Pipes the SB message definitions will hold.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 357 of file example\_mission\_cfg.h.

**12.36.2.18 CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE** #define CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE 32768

**Purpose** Maximum SB Message Size

**Description:**

The following definition dictates the maximum message size allowed on the software bus. SB checks the pkt length field in the header of all messages sent. If the pkt length field indicates the message is larger than this define, SB sends an event and rejects the send.

**Limits**

This parameter has a lower limit of 6 (CCSDS primary header size). There are no restrictions on the upper limit however, the maximum message size is system dependent and should be verified. Total message size values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 340 of file example\_mission\_cfg.h.

**12.36.2.19 CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN** #define CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_L←  
EN (CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH + CFE\_MISSION\_MAX\_API\_LEN + 4)

**Purpose** Maximum Length of Full Table Name in messages

**Description:**

Indicates the maximum length (in characters) of the entire table name within software bus messages, in "App←Name.TableName" notation.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 402 of file example\_mission\_cfg.h.

**12.36.2.20 CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH** #define CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH 16

**Purpose** Maximum Table Name Length

**Description:**

Indicates the maximum length (in characters) of the table name ('TblName') portion of a Full Table Name of the following form: "ApplicationName.TblName"

This length does not need to include an extra character for NULL termination.

**Limits**

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 382 of file example\_mission\_cfg.h.

**12.36.2.21 CFE\_MISSION\_TIME\_AT\_TONE\_WAS** #define CFE\_MISSION\_TIME\_AT\_TONE\_WAS true

**Purpose** Default Time and Tone Order

Description:

Time Services may be configured to expect the time at the tone data packet to either precede or follow the tone signal. If the time at the tone data packet follows the tone signal, then the data within the packet describes what the time "was" at the tone. If the time at the tone data packet precedes the tone signal, then the data within the packet describes what the time "will be" at the tone. One, and only one, of the following symbols must be set to true:

- CFE\_MISSION\_TIME\_AT\_TONE\_WAS
- CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE Note: If Time Services is defined as using a simulated tone signal (see [CFE\\_MISSION\\_TIME\\_CFG\\_FAKE\\_TONE](#) above), then the tone data packet must follow the tone signal.

Limits

Either CFE\_MISSION\_TIME\_AT\_TONE\_WAS or CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE must be set to true. They may not both be true and they may not both be false.

Definition at line 468 of file example\_mission\_cfg.h.

**12.36.2.22 CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE** #define CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE false

Definition at line 469 of file example\_mission\_cfg.h.

**12.36.2.23 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI** #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI true

**Purpose** Default Time Format

Description:

The following definitions select either UTC or TAI as the default (mission specific) time format. Although it is possible for an application to request time in a specific format, most callers should use [CFE\\_TIME\\_GetTime\(\)](#), which returns time in the default format. This avoids having to modify each individual caller when the default choice is changed.

Limits

if CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI is defined as true then CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC must be defined as false. if CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI is defined as false then CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC must be defined as true.

Definition at line 432 of file example\_mission\_cfg.h.

**12.36.2.24 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC** #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC false

Definition at line 433 of file example\_mission\_cfg.h.

**12.36.2.25 CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE** #define CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE true

**Purpose** Default Time Format

Description:

The following definition enables the use of a simulated time at the tone signal using a software bus message.

Limits

Not Applicable

Definition at line 445 of file example\_mission\_cfg.h.

**12.36.2.26 CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS 0

Definition at line 527 of file example\_mission\_cfg.h.

**12.36.2.27 CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS 1000

Definition at line 528 of file example\_mission\_cfg.h.

**12.36.2.28 CFE\_MISSION\_TIME\_DEF\_LEAPS** #define CFE\_MISSION\_TIME\_DEF\_LEAPS 37

Definition at line 525 of file example\_mission\_cfg.h.

**12.36.2.29 CFE\_MISSION\_TIME\_DEF\_MET\_SECS** #define CFE\_MISSION\_TIME\_DEF\_MET\_SECS 1000

**Purpose** Default Time Values

Description:

Default time values are provided to avoid problems due to time calculations performed after startup but before commands can be processed. For example, if the default time format is UTC then it is important that the sum of MET and STCF always exceed the value of Leap Seconds to prevent the UTC time calculation ( $\text{time} = \text{MET} + \text{STCF} - \text{Leap Seconds}$ ) from resulting in a negative (very large) number.

Some past missions have also created known (albeit wrong) default timestamps. For example, assume the epoch is defined as Jan 1, 1970 and further assume the default time values are set to create a timestamp of Jan 1, 2000. Even though the year 2000 timestamps are wrong, it may be of value to keep the time within some sort of bounds acceptable to the software.

Note: Sub-second units are in micro-seconds (0 to 999,999) and all values must be defined

Limits

Not Applicable

Definition at line 519 of file example\_mission\_cfg.h.

**12.36.2.30 CFE\_MISSION\_TIME\_DEF\_MET\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_MET\_SUBS 0

Definition at line 520 of file example\_mission\_cfg.h.

**12.36.2.31 CFE\_MISSION\_TIME\_DEF\_STCF\_SECS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SECS 1000000  
Definition at line 522 of file example\_mission\_cfg.h.

**12.36.2.32 CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS 0  
Definition at line 523 of file example\_mission\_cfg.h.

**12.36.2.33 CFE\_MISSION\_TIME\_EPOCH\_DAY** #define CFE\_MISSION\_TIME\_EPOCH\_DAY 1  
Definition at line 546 of file example\_mission\_cfg.h.

**12.36.2.34 CFE\_MISSION\_TIME\_EPOCH\_HOUR** #define CFE\_MISSION\_TIME\_EPOCH\_HOUR 0  
Definition at line 547 of file example\_mission\_cfg.h.

**12.36.2.35 CFE\_MISSION\_TIME\_EPOCH\_MICROS** #define CFE\_MISSION\_TIME\_EPOCH\_MICROS 0  
Definition at line 550 of file example\_mission\_cfg.h.

**12.36.2.36 CFE\_MISSION\_TIME\_EPOCH\_MINUTE** #define CFE\_MISSION\_TIME\_EPOCH\_MINUTE 0  
Definition at line 548 of file example\_mission\_cfg.h.

**12.36.2.37 CFE\_MISSION\_TIME\_EPOCH\_SECOND** #define CFE\_MISSION\_TIME\_EPOCH\_SECOND 0  
Definition at line 549 of file example\_mission\_cfg.h.

**12.36.2.38 CFE\_MISSION\_TIME\_EPOCH\_YEAR** #define CFE\_MISSION\_TIME\_EPOCH\_YEAR 1980

**Purpose** Default EPOCH Values

**Description:**

Default ground time epoch values Note: these values are used only by the [CFE\\_TIME\\_Print\(\)](#) API function

**Limits**

Year - must be within 136 years Day - Jan 1 = 1, Feb 1 = 32, etc. Hour - 0 to 23 Minute - 0 to 59 Second - 0 to 59  
Micros - 0 to 999999

Definition at line 545 of file example\_mission\_cfg.h.

**12.36.2.39 CFE\_MISSION\_TIME\_FS\_FACTOR** #define CFE\_MISSION\_TIME\_FS\_FACTOR 789004800

**Purpose** Time File System Factor

**Description:**

Define the s/c vs file system time conversion constant...

Note: this value is intended for use only by CFE TIME API functions to convert time values based on the ground system epoch (s/c time) to and from time values based on the file system epoch (fs time).

FS time = S/C time + factor S/C time = FS time - factor

**Worksheet:**

S/C epoch = Jan 1, 2005 (LRO ground system epoch) FS epoch = Jan 1, 1980 (vxWorks DOS file system epoch)

Delta = 25 years, 0 days, 0 hours, 0 minutes, 0 seconds

Leap years = 1980, 1984, 1988, 1992, 1996, 2000, 2004 (divisible by 4 – except if by 100 – unless also by 400)

1 year = 31,536,000 seconds 1 day = 86,400 seconds 1 hour = 3,600 seconds 1 minute = 60 seconds

25 years = 788,400,000 seconds 7 extra leap days = 604,800 seconds

total delta = 789,004,800 seconds

**Limits**

Not Applicable

Definition at line 588 of file example\_mission\_cfg.h.

**12.36.2.40 CFE\_MISSION\_TIME\_MAX\_ELAPSED #define CFE\_MISSION\_TIME\_MAX\_ELAPSED 200000**

Definition at line 494 of file example\_mission\_cfg.h.

**12.36.2.41 CFE\_MISSION\_TIME\_MIN\_ELAPSED #define CFE\_MISSION\_TIME\_MIN\_ELAPSED 0****Purpose** Min and Max Time Elapsed**Description:**

Based on the definition of Time and Tone Order (CFE\_MISSION\_TIME\_AT\_TONE\_WAS/WILL\_BE) either the "time at the tone" signal or data packet will follow the other. This definition sets the valid window of time for the second of the pair to lag behind the first. Time Services will invalidate both the tone and packet if the second does not arrive within this window following the first.

For example, if the data packet follows the tone, it might be valid for the data packet to arrive between zero and 100,000 micro-seconds after the tone. But, if the tone follows the packet, it might be valid only if the packet arrived between 200,000 and 700,000 micro-seconds before the tone.

Note: units are in micro-seconds

**Limits**

0 to 999,999 decimal

Definition at line 493 of file example\_mission\_cfg.h.

**12.37 cfe/cmake/sample\_defs/example\_platform\_cfg.h File Reference****Macros**

- #define CFE\_PLATFORM\_ENDIAN CCSDS\_LITTLE\_ENDIAN
- #define CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC 30000
- #define CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY 68
- #define CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING "/cf"
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING "/ram"

- #define CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS 32
- #define CFE\_PLATFORM\_ES\_MAX\_LIBRARIES 10
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES 20
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE 256
- #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072
- #define CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE 30
- #define CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS 8
- #define CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE 1000
- #define CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT 5
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE 512
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS 4096
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED 30
- #define CFE\_PLATFORM\_ES\_CDS\_SIZE (128 \* 1024)
- #define CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE (1024 \* 1024)
- #define CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN 4
- #define CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE "/cf/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE "/ram/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE "/ram/cfe\_es\_app\_info.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE "/ram/cfe\_es\_taskinfo.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE "/ram/cfe\_es\_syslog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE "/ram/cfe\_erlog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME "/ram/cfe\_es\_perf.dat"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE "/ram/cfe\_cds\_reg.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE 0
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE 1
- #define CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE 10000
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE 0
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL ~CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE 0
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL ~CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY 200
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY 20
- #define CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS 50
- #define CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES 512
- #define CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS 2
- #define CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS 17
- #define CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS 10
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 32
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 512

- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 32
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC 50
- #define CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC 1000
- #define CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY 61
- #define CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS 8
- #define CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST 32
- #define CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC 15
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE "/ram/cfe\_evs.log"
- #define CFE\_PLATFORM\_EVS\_LOG\_MAX 20
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE "/ram/cfe\_evs\_app.dat"
- #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT 0x0001
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG 0xE
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE 1
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE CFE\_EVS\_MsgFormat\_LONG
- #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256
- #define CFE\_PLATFORM\_SB\_MAX\_PIPES 64
- #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4
- #define CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES 524288
- #define CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID 0x1FFF
- #define CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME "/ram/cfe\_sb\_route.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME "/ram/cfe\_sb\_pipe.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME "/ram/cfe\_sb\_msgmap.dat"
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_SB\_SEND\_NO\_SUBS\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_EVS\_FIRST\_4\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_SB\_DUP\_SUBSCRIPTION\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_EVS\_FIRST\_4\_STOP

```
• #define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
• #define CFE_PLATFORM_SB_FILTER_MASK3 CFE_EVS_FIRST_16_STOP
• #define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
• #define CFE_PLATFORM_SB_FILTER_MASK4 CFE_EVS_FIRST_16_STOP
• #define CFE_PLATFORM_SB_FILTERED_EVENT5 0
• #define CFE_PLATFORM_SB_FILTER_MASK5 CFE_EVS_NO_FILTER
• #define CFE_PLATFORM_SB_FILTERED_EVENT6 0
• #define CFE_PLATFORM_SB_FILTER_MASK6 CFE_EVS_NO_FILTER
• #define CFE_PLATFORM_SB_FILTERED_EVENT7 0
• #define CFE_PLATFORM_SB_FILTER_MASK7 CFE_EVS_NO_FILTER
• #define CFE_PLATFORM_SB_FILTERED_EVENT8 0
• #define CFE_PLATFORM_SB_FILTER_MASK8 CFE_EVS_NO_FILTER
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01 8
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_05 64
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_06 96
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_07 128
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_08 160
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_09 256
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_10 512
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_11 1024
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_12 2048
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_13 4096
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_14 8192
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_15 16384
• #define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_16 32768
• #define CFE_PLATFORM_SB_MAX_BLOCK_SIZE (CFE_MISSION_SB_MAX_SB_MSG_SIZE + 128)
• #define CFE_PLATFORM_SB_START_TASK_PRIORITY 64
• #define CFE_PLATFORM_SB_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
• #define CFE_PLATFORM_TBL_START_TASK_PRIORITY 70
• #define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
• #define CFE_PLATFORM_TBL_BUF_MEMORY_BYTES 524288
• #define CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE 16384
• #define CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE 16384
• #define CFE_PLATFORM_TBL_MAX_NUM_TABLES 128
• #define CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES 32
• #define CFE_PLATFORM_TBL_MAX_NUM_HANDLES 256
• #define CFE_PLATFORM_TBL_MAX_SIMULTANEOUS_LOADS 4
• #define CFE_PLATFORM_TBL_MAX_NUM_VALIDATIONS 10
• #define CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE "/ram/cfe_tbl_reg.log"
• #define CFE_PLATFORM_TBL_VALID_SCID_COUNT 0
• #define CFE_PLATFORM_TBL_U32FROM4CHARS(_C1, _C2, _C3, _C4) ((uint32)(_C1) << 24 | (uint32)(_C2)
<< 16 | (uint32)(_C3) << 8 | (uint32)(_C4))
• #define CFE_PLATFORM_TBL_VALID_SCID_1 (0x42)
• #define CFE_PLATFORM_TBL_VALID_SCID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))
• #define CFE_PLATFORM_TBL_VALID_PRID_COUNT 0
• #define CFE_PLATFORM_TBL_VALID_PRID_1 (1)
• #define CFE_PLATFORM_TBL_VALID_PRID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS('a', 'b', 'c', 'd'))
• #define CFE_PLATFORM_TBL_VALID_PRID_3 0
```

- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 0
- #define CFE\_PLATFORM\_TIME\_CFG\_SERVER true
- #define CFE\_PLATFORM\_TIME\_CFG\_CLIENT false
- #define CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL true
- #define CFE\_PLATFORM\_TIME\_CFG\_SIGNAL false
- #define CFE\_PLATFORM\_TIME\_CFG\_SOURCE false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME false
- #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS 0
- #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS 500000
- #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS 27
- #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS 0
- #define CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT 20000
- #define CFE\_PLATFORM\_TIME\_CFG\_START\_FLY 2
- #define CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY 8
- #define CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY 60
- #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY 25
- #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY 25
- #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE 8192

### 12.37.1 Detailed Description

This header file contains the internal configuration parameters and typedefs with platform scope.

This provides default values for configurable items that do NOT affect the interface(s) of this module. This includes internal parameters, path names, and limit value(s) that are relevant for a specific platform.

#### Note

It is no longer necessary to provide this file directly in the defs directory, but if present, this file is still supported/usable for backward compatibility. To use this file, it should be called "cfe\_platform\_cfg.h".

Going forward, more fine-grained (module/purposes-specific) header files are included with each submodule. These may be overridden as necessary, but only if a definition within that file needs to be changed from the default. This approach will reduce the amount of duplicate/cloned definitions and better support alternative build configurations in the future.

Note that if this file is present, the fine-grained header files noted above will *not* be used.

### 12.37.2 Macro Definition Documentation

#### 12.37.2.1 CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC #define CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MS← EC 30000

**Purpose** CFE core application startup timeout

**Description:**

The upper limit for the amount of time that the cFE core applications (ES, SB, EVS, TIME, TBL) are each allotted to reach their respective "ready" states.

The CFE "main" thread starts individual tasks for each of the core applications (except FS). Each of these must perform some initialization work before the next core application can be started, so the main thread waits to ensure that the application has reached the "ready" state before starting the next application.

If any core application fails to start, then it indicates a major problem with the system and startup is aborted.

Units are in milliseconds

**Limits:**

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 84 of file example\_platform\_cfg.h.

**12.37.2.2 CFE\_PLATFORM\_ENDIAN #define CFE\_PLATFORM\_ENDIAN CCSDS\_LITTLE\_ENDIAN****Purpose** Platform Endian Indicator**Description:**

The value of this constant indicates the endianess of the target system

**Limits**

This parameter has a lower limit of 0 and an upper limit of 1.

Definition at line 60 of file example\_platform\_cfg.h.

**12.37.2.3 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 its main loop and call CFE\_ES→\_ExitApp. Once it calls Exit App, then ES can delete, restart, or reload the app the next time it scans the app table.
3. If the App is not responding, the ES App will decrement this Kill Timeout value each time it runs. If the timeout value reaches zero, ES will kill the app.

The Kill timeout value depends on the [CFE\\_PLATFORM\\_ES\\_APP\\_SCAN\\_RATE](#). If the Scan Rate is 1000, or 1 second, and this [CFE\\_PLATFORM\\_ES\\_APP\\_KILL\\_TIMEOUT](#) is set to 5, then it will take 5 seconds to kill a non-responding App. If the Scan Rate is 250, or 1/4 second, and the [CFE\\_PLATFORM\\_ES\\_APP\\_KILL\\_TIMEOUT](#) is set to 2, then it will take 1/2 second to time out.

**Limits**

There is a lower limit of 1 and an upper limit of 100 on this configuration parameter. Units are number of [CFE\\_PLATFORM\\_ES\\_APP\\_SCAN\\_RATE](#) cycles.

Definition at line 289 of file example\_platform\_cfg.h.

**12.37.2.4 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 260 of file example\_platform\_cfg.h.

**12.37.2.5 CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE←  
ZE 80000

Definition at line 831 of file example\_platform\_cfg.h.

**12.37.2.6 CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES** #define CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES←  
ES 512

**Purpose** Define Maximum Number of Registered CDS Blocks

**Description:**

Maximum number of registered CDS Blocks

**Limits**

There is a lower limit of 8. There are no restrictions on the upper limit however, the maximum number of CDS entries is system dependent and should be verified.

Definition at line 721 of file example\_platform\_cfg.h.

**12.37.2.7 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01←  
ZE\_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 815 of file example\_platform\_cfg.h.

**12.37.2.8 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 16

Definition at line 816 of file example\_platform\_cfg.h.

**12.37.2.9 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 32

Definition at line 817 of file example\_platform\_cfg.h.

**12.37.2.10 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 48

Definition at line 818 of file example\_platform\_cfg.h.

**12.37.2.11 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 64

Definition at line 819 of file example\_platform\_cfg.h.

**12.37.2.12 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96

Definition at line 820 of file example\_platform\_cfg.h.

**12.37.2.13 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128

Definition at line 821 of file example\_platform\_cfg.h.

**12.37.2.14 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160

Definition at line 822 of file example\_platform\_cfg.h.

**12.37.2.15 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256

Definition at line 823 of file example\_platform\_cfg.h.

**12.37.2.16 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512

Definition at line 824 of file example\_platform\_cfg.h.

**12.37.2.17 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024

Definition at line 825 of file example\_platform\_cfg.h.

**12.37.2.18 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048

Definition at line 826 of file example\_platform\_cfg.h.

**12.37.2.19 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096

Definition at line 827 of file example\_platform\_cfg.h.

**12.37.2.20 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192

Definition at line 828 of file example\_platform\_cfg.h.

**12.37.2.21 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384

Definition at line 829 of file example\_platform\_cfg.h.

**12.37.2.22 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 32768

Definition at line 830 of file example\_platform\_cfg.h.

**12.37.2.23 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 366 of file example\_platform\_cfg.h.

**12.37.2.24 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 448 of file example\_platform\_cfg.h.

**12.37.2.25 CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_CD←  
S\_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 522 of file example\_platform\_cfg.h.

**12.37.2.26 CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FI←  
LE "/ram/cfe\_erlog.log"

**Purpose** Default Exception and Reset (ER) Log Filename

### Description:

The value of this constant defines the filename used to store the Exception and Reset (ER) Log. This filename is used only when no filename is specified in the command to dump the ER log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

### Limits

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 494 of file example\_platform\_cfg.h.

**12.37.2.27 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 508 of file example\_platform\_cfg.h.

**12.37.2.28 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 540 of file example\_platform\_cfg.h.

**12.37.2.29 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 558 of file example\_platform\_cfg.h.

**12.37.2.30 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 708 of file example\_platform\_cfg.h.

```
12.37.2.31 CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE #define CFE_PLATFORM_ES_DEFAULT_SYSLOG_FI←  
LE "/ram/cfe_es_syslog.log"
```

**Purpose** Default System Log Filename

**Description:**

The value of this constant defines the filename used to store important information (as ASCII text strings) that might not be able to be sent in an Event Message. This filename is used only when no filename is specified in the command to dump the system log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 479 of file example\_platform\_cfg.h.

```
12.37.2.32 CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE #define CFE_PLATFORM_ES_DEFAULT_TASK_LO←  
G_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 463 of file example\_platform\_cfg.h.

```
12.37.2.33 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 187 of file example\_platform\_cfg.h.

**12.37.2.34 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 201 of file example\_platform\_cfg.h.

**12.37.2.35 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 160 of file example\_platform\_cfg.h.

**12.37.2.36 CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000

Definition at line 804 of file example\_platform\_cfg.h.

**12.37.2.37 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 241 of file example\_platform\_cfg.h.

**12.37.2.38 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 174 of file example\_platform\_cfg.h.

**12.37.2.39 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 769 of file example\_platform\_cfg.h.

**12.37.2.40 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 736 of file example\_platform\_cfg.h.

**12.37.2.41 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\_TB\_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 788 of file example\_platform\_cfg.h.

**12.37.2.42 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16  
Definition at line 789 of file example\_platform\_cfg.h.

**12.37.2.43 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 32  
Definition at line 790 of file example\_platform\_cfg.h.

**12.37.2.44 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 48  
Definition at line 791 of file example\_platform\_cfg.h.

**12.37.2.45 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 64  
Definition at line 792 of file example\_platform\_cfg.h.

**12.37.2.46 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 96  
Definition at line 793 of file example\_platform\_cfg.h.

**12.37.2.47 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 128  
Definition at line 794 of file example\_platform\_cfg.h.

**12.37.2.48 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 160  
Definition at line 795 of file example\_platform\_cfg.h.

**12.37.2.49 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 256  
Definition at line 796 of file example\_platform\_cfg.h.

**12.37.2.50 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 512  
Definition at line 797 of file example\_platform\_cfg.h.

**12.37.2.51 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 1024  
Definition at line 798 of file example\_platform\_cfg.h.

**12.37.2.52 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 2048  
Definition at line 799 of file example\_platform\_cfg.h.

**12.37.2.53 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 4096  
Definition at line 800 of file example\_platform\_cfg.h.

**12.37.2.54 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 8192  
Definition at line 801 of file example\_platform\_cfg.h.

**12.37.2.55 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 16384  
Definition at line 802 of file example\_platform\_cfg.h.

**12.37.2.56 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 32768  
Definition at line 803 of file example\_platform\_cfg.h.

**12.37.2.57 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 405 of file example\_platform\_cfg.h.

**12.37.2.58 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 128 of file example\_platform\_cfg.h.

**12.37.2.59 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 419 of file example\_platform\_cfg.h.

**12.37.2.60 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 230 of file example\_platform\_cfg.h.

**12.37.2.61 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 682 of file example\_platform\_cfg.h.

**12.37.2.62 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 653 of file example\_platform\_cfg.h.

**12.37.2.63 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 667 of file example\_platform\_cfg.h.

**12.37.2.64 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 574 of file example\_platform\_cfg.h.

**12.37.2.65 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 692 of file example\_platform\_cfg.h.

**12.37.2.66 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 594 of file example\_platform\_cfg.h.

**12.37.2.67 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 605 of file example\_platform\_cfg.h.

**12.37.2.68 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 584 of file example\_platform\_cfg.h.

**12.37.2.69 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 627 of file example\_platform\_cfg.h.

**12.37.2.70 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 638 of file example\_platform\_cfg.h.

**12.37.2.71 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 616 of file example\_platform\_cfg.h.

**12.37.2.72 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 751 of file example\_platform\_cfg.h.

**12.37.2.73 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 144 of file example\_platform\_cfg.h.

**12.37.2.74 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 325 of file example\_platform\_cfg.h.

**12.37.2.75 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 349 of file example\_platform\_cfg.h.

**12.37.2.76 CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE 512

**Purpose** ES Ram Disk Sector Size

**Description:**

Defines the ram disk sector size. The ram disk is 1 of 4 memory areas that are preserved on a processor reset.  
NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as USER\_RESERVED\_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

**Limits**

There is a lower limit of 128. There are no restrictions on the upper limit however, the maximum RAM disk sector size is system dependent and should be verified.

Definition at line 307 of file example\_platform\_cfg.h.

**12.37.2.77 CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY 68

**Purpose** Define ES Task Priority

**Description:**

Defines the cFE\_ES Task priority.

**Limits**

Not Applicable

Definition at line 101 of file example\_platform\_cfg.h.

**12.37.2.78 CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define ES Task Stack Size

**Description:**

Defines the cFE\_ES Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 116 of file example\_platform\_cfg.h.

**12.37.2.79 CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC 1000

**Purpose** Startup script timeout

**Description:**

The upper limit for the total amount of time that all apps listed in the CFE ES startup script may take to all become ready.

Unlike the "core" app timeout, this is a soft limit; if the allotted time is exceeded, it probably indicates an issue with one of the apps, but does not cause CFE ES to take any additional action other than logging the event to the syslog.  
Units are in milliseconds

**Limits:**

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 871 of file example\_platform\_cfg.h.

**12.37.2.80 CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC 50

**Purpose** Poll timer for startup sync delay

**Description:**

During startup, some tasks may need to synchronize their own initialization with the initialization of other applications in the system.

CFE ES implements an API to accomplish this, that performs a task delay (sleep) while polling the overall system state until other tasks are ready.

This value controls the amount of time that the CFE\_ES\_ApplicationSyncDelay will sleep between each check of the system state. This should be large enough to allow other tasks to run, but not so large as to noticeably delay the startup completion.

Units are in milliseconds

**Limits:**

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 853 of file example\_platform\_cfg.h.

**12.37.2.81 CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE** #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072

**Purpose** Define Size of the cFE System Log.

**Description:**

Defines the size in bytes of the cFE system log. The system log holds variable length strings that are terminated by a linefeed and null character.

**Limits**

There is a lower limit of 512. There are no restrictions on the upper limit however, the maximum system log size is system dependent and should be verified.

Definition at line 216 of file example\_platform\_cfg.h.

**12.37.2.82 CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE** #define CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE  
ZE (1024 \* 1024)

**Purpose** Define User Reserved Memory Size

**Description:**

User Reserved Memory Size. This is the size in bytes of the cFE User reserved Memory area. This is a block of memory that is available for cFE application use. The address is obtained by calling [CFE\\_PSP GetUserReservedArea](#). The User Reserved Memory is one of four memory areas that are preserved during a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as USER\_RESERVED\_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

**Limits**

There is a lower limit of 1024 and an upper limit of [UINT\\_MAX](#) (4 Gigabytes) on this configuration parameter.

Definition at line 386 of file example\_platform\_cfg.h.

**12.37.2.83 CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE** #define CFE\_PLATFORM\_ES\_VOLATILE\_STARTUPFILE  
"/ram/cfe\_es\_startup.scr"

**Purpose** ES Volatile Startup Filename

**Description:**

The value of this constant defines the path and name of the file that contains a list of modules that will be loaded and started by the cFE after the cFE finishes its startup sequence.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 433 of file example\_platform\_cfg.h.

**12.37.2.84 CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC** #define CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC  
15

**Purpose** Sustained number of event messages per second per app before squelching

**Description:**

Sustained number of events that may be emitted per app per second.

**Limits**

This number must be less than or equal to [CFE\\_PLATFORM\\_EVS\\_MAX\\_APP\\_EVENT\\_BURST](#). Values lower than 8 may cause functional and unit test failures.

Definition at line 939 of file example\_platform\_cfg.h.

```
12.37.2.85 CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE #define CFE_PLATFORM_EVS_DEFAULT_APP_DA←  
TA_FILE "/ram/cfe_evs_app.dat"
```

**Purpose** Default EVS Application Data Filename

**Description:**

The value of this constant defines the filename used to store the EVS Application Data(event counts/filtering information). This filename is used only when no filename is specified in the command to dump the event log.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 980 of file example\_platform\_cfg.h.

```
12.37.2.86 CFE_PLATFORM_EVS_DEFAULT_LOG_FILE #define CFE_PLATFORM_EVS_DEFAULT_LOG_FI←  
LE "/ram/cfe_evs.log"
```

**Purpose** Default Event Log Filename

**Description:**

The value of this constant defines the filename used to store the Event Services local event log. This filename is used only when no filename is specified in the command to dump the event log.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 953 of file example\_platform\_cfg.h.

```
12.37.2.87 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 1027 of file example\_platform\_cfg.h.

**12.37.2.88 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 1040 of file example\_platform\_cfg.h.

**12.37.2.89 CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG** #define CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FL←  
AG 0xE

**Purpose** Default EVS Event Type Filter Mask

**Description:**

Defines a state of on or off for all four event types. The term event 'type' refers to the criticality level and may be Debug, Informational, Error or Critical. Each event type has a bit position. (bit 0 = Debug, bit 1 = Info, bit 2 = Error, bit 3 = Critical). This is a global setting, meaning it applies to all applications. To filter an event type, set its bit to zero. For example, 0xE means Debug = OFF, Info = ON, Error = ON, Critical = ON

**Limits**

The valid settings are 0x0 to 0xF.

Definition at line 1011 of file example\_platform\_cfg.h.

**12.37.2.90 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 965 of file example\_platform\_cfg.h.

**12.37.2.91 CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST** #define CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST 32

**Purpose** Maximum number of event before squelching

**Description:**

Maximum number of events that may be emitted per app per second. Setting this to 0 will cause events to be unrestricted.

**Limits**

This number must be less than or equal to INT\_MAX/1000

Definition at line 927 of file example\_platform\_cfg.h.

**12.37.2.92 CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS** #define CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS 8

**Purpose** Define Maximum Number of Event Filters per Application

**Description:**

Maximum number of events that may be filtered per application.

**Limits**

There are no restrictions on the lower and upper limits however, the maximum number of event filters is system dependent and should be verified.

Definition at line 915 of file example\_platform\_cfg.h.

**12.37.2.93 CFE\_PLATFORM\_EVS\_PORT\_DEFAULT** #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT 0x0001

**Purpose** Default EVS Output Port State

**Description:**

Defines the default port state (enabled or disabled) for the four output ports defined within the Event Service. Port 1 is usually the uart output terminal. To enable a port, set the proper bit to a 1. Bit 0 is port 1, bit 1 is port2 etc.

**Limits**

The valid settings are 0x0 to 0xF.

Definition at line 994 of file example\_platform\_cfg.h.

**12.37.2.94 CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY 61

**Purpose** Define EVS Task Priority

**Description:**

Defines the cFE\_EVS Task priority.

**Limits**

Not Applicable

Definition at line 887 of file example\_platform\_cfg.h.

**12.37.2.95 CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define EVS Task Stack Size

**Description:**

Defines the cFE\_EVS Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 902 of file example\_platform\_cfg.h.

**12.37.2.96 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 1135 of file example\_platform\_cfg.h.

**12.37.2.97 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 1206 of file example\_platform\_cfg.h.

**12.37.2.98 CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT** #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4

**Purpose** Default Subscription Message Limit

**Description:**

Dictates the default Message Limit when using the [CFE\\_SB\\_Subscribe](#) API. This will limit the number of messages with a specific message ID that can be received through a subscription. This only changes the default; other message limits can be set on a per subscription basis using [CFE\\_SB\\_SubscribeEx](#).

**Limits**

This parameter has a lower limit of 4 and an upper limit of 65535.

Definition at line 1113 of file example\_platform\_cfg.h.

**12.37.2.99 CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FIL←  
ENAME "/ram/cfe\_sb\_pipe.dat"

**Purpose** Default Pipe Information Filename

**Description:**

The value of this constant defines the filename used to store the software bus pipe information. This filename is used only when no filename is specified in the command.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 1189 of file example\_platform\_cfg.h.

**12.37.2.100 CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_RO←  
UTING\_FILENAME "/ram/cfe\_sb\_route.dat"

**Purpose** Default Routing Information Filename

**Description:**

The value of this constant defines the filename used to store the software bus routing information. This filename is used only when no filename is specified in the command.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 1175 of file example\_platform\_cfg.h.

**12.37.2.101 CFE\_PLATFORM\_SB\_FILTER\_MASK1** #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 [CFE\\_EVS\\_FIRST\\_4\\_STOP](#)

Definition at line 1224 of file example\_platform\_cfg.h.

**12.37.2.102 CFE\_PLATFORM\_SB\_FILTER\_MASK2** #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_EVS\_FIRST\_4\_STOP  
Definition at line 1227 of file example\_platform\_cfg.h.

**12.37.2.103 CFE\_PLATFORM\_SB\_FILTER\_MASK3** #define CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_EVS\_FIRST\_16\_STOP  
Definition at line 1230 of file example\_platform\_cfg.h.

**12.37.2.104 CFE\_PLATFORM\_SB\_FILTER\_MASK4** #define CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_EVS\_FIRST\_16\_STOP  
Definition at line 1233 of file example\_platform\_cfg.h.

**12.37.2.105 CFE\_PLATFORM\_SB\_FILTER\_MASK5** #define CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_EVS\_NO\_FILTER  
Definition at line 1236 of file example\_platform\_cfg.h.

**12.37.2.106 CFE\_PLATFORM\_SB\_FILTER\_MASK6** #define CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_EVS\_NO\_FILTER  
Definition at line 1239 of file example\_platform\_cfg.h.

**12.37.2.107 CFE\_PLATFORM\_SB\_FILTER\_MASK7** #define CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_EVS\_NO\_FILTER  
Definition at line 1242 of file example\_platform\_cfg.h.

**12.37.2.108 CFE\_PLATFORM\_SB\_FILTER\_MASK8** #define CFE\_PLATFORM\_SB\_FILTER\_MASK8 CFE\_EVS\_NO\_FILTER  
Definition at line 1245 of file example\_platform\_cfg.h.

**12.37.2.109 CFE\_PLATFORM\_SB\_FILTERED\_EVENT1** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_SB\_SEND\_NO\_SUBS\_EID

#### Purpose

SB Event Filtering

##### Description:

This group of configuration parameters dictates what SB events will be filtered through SB. The filtering will begin after the SB task initializes and stay in effect until a cmd to SB changes it. This allows the operator to set limits on the number of event messages that are sent during system initialization. NOTE: Set all unused event values and mask values to zero

##### Limits

This filtering applies only to SB events. These parameters have a lower limit of 0 and an upper limit of 65535.

Definition at line 1223 of file example\_platform\_cfg.h.

**12.37.2.110 CFE\_PLATFORM\_SB\_FILTERED\_EVENT2** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_SB\_DUP\_SUBSCRIP\_EID  
Definition at line 1226 of file example\_platform\_cfg.h.

**12.37.2.111 CFE\_PLATFORM\_SB\_FILTERED\_EVENT3** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT3 CFE\_SB\_MSGID\_LIM\_ERR\_EID  
Definition at line 1229 of file example\_platform\_cfg.h.

**12.37.2.112 CFE\_PLATFORM\_SB\_FILTERED\_EVENT4** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT4 CFE\_SB\_Q\_FULL\_ERR\_EID  
Definition at line 1232 of file example\_platform\_cfg.h.

**12.37.2.113 CFE\_PLATFORM\_SB\_FILTERED\_EVENT5** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT5 0  
Definition at line 1235 of file example\_platform\_cfg.h.

**12.37.2.114 CFE\_PLATFORM\_SB\_FILTERED\_EVENT6** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT6 0  
Definition at line 1238 of file example\_platform\_cfg.h.

**12.37.2.115 CFE\_PLATFORM\_SB\_FILTERED\_EVENT7** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT7 0  
Definition at line 1241 of file example\_platform\_cfg.h.

**12.37.2.116 CFE\_PLATFORM\_SB\_FILTERED\_EVENT8** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT8 0  
Definition at line 1244 of file example\_platform\_cfg.h.

**12.37.2.117 CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID** #define CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID 0x1FFF

**Purpose** Highest Valid Message Id

**Description:**

The value of this constant dictates the range of valid message ID's, from 0 to CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID (inclusive).

Although this can be defined differently across platforms, each platform can only publish/subscribe to message ids within their allowable range. Typically this value is set the same across all mission platforms to avoid this complexity.

**Limits**

CFE\_SB\_INVALID\_MSG is set to the maximum representable number of type [CFE\\_SB\\_MsgId\\_t](#). CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID lower limit is 1, up to CFE\_SB\_INVALID\_MSG\_ID - 1.

When using the direct message map implementation for software bus routing, this value is used to size the map where a value of 0xFFFF results in a 16 KBytes map and 0xFFFF is 128 KBytes.

When using the hash implementation for software bus routing, a multiple of the CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS is used to size the message map. In that case the range selected here does not impact message map memory use, so it's reasonable to use up to the full range supported by the message ID implementation.

Definition at line 1161 of file example\_platform\_cfg.h.

**12.37.2.118 CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE (CFE\_MISSION\_SB\_MAX\_SB\_MESSAGE\_SIZE + 128)

Definition at line 1274 of file example\_platform\_cfg.h.

**12.37.2.119 CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT** #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16

**Purpose** Maximum Number of unique local destinations a single MsgId can have

**Description:**

Dictates the maximum number of unique local destinations a single MsgId can have.

**Limits**

This parameter has a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of destinations per packet is system dependent and should be verified. Destination number values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 1098 of file example\_platform\_cfg.h.

**12.37.2.120 CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS** #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256

**Purpose** Maximum Number of Unique Message IDs SB Routing Table can hold

**Description:**

Dictates the maximum number of unique MsgIds the SB routing table will hold. This constant has a direct effect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

**Limits**

This must be a power of two if software bus message routing hash implementation is being used. Lower than 64 will cause unit test failures, and telemetry reporting is impacted below 32. There is no hard upper limit, but impacts memory footprint. For software bus message routing search implementation the number of msg ids subscribed to impacts performance.

Definition at line 1065 of file example\_platform\_cfg.h.

**12.37.2.121 CFE\_PLATFORM\_SB\_MAX\_PIPES** #define CFE\_PLATFORM\_SB\_MAX\_PIPES 64

**Purpose** Maximum Number of Unique Pipes SB Routing Table can hold

**Description:**

Dictates the maximum number of unique Pipes the SB routing table will hold. This constant has a direct effect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

**Limits**

This parameter has a lower limit of 1. This parameter must also be less than or equal to OS\_MAX\_QUEUES.

Definition at line 1082 of file example\_platform\_cfg.h.

**12.37.2.122 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 8

**Purpose** Define SB Memory Pool Block Sizes

**Description:**

Software Bus Memory Pool Block Sizes

**Limits**

These sizes MUST be increasing and MUST be an integral multiple of 4. The number of block sizes defined cannot exceed [CFE\\_PLATFORM\\_ES\\_POOL\\_MAX\\_BUCKETS](#)

Definition at line 1258 of file example\_platform\_cfg.h.

**12.37.2.123 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 16

Definition at line 1259 of file example\_platform\_cfg.h.

**12.37.2.124 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 20

Definition at line 1260 of file example\_platform\_cfg.h.

**12.37.2.125 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 36

Definition at line 1261 of file example\_platform\_cfg.h.

**12.37.2.126 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 64

Definition at line 1262 of file example\_platform\_cfg.h.

**12.37.2.127 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 96

Definition at line 1263 of file example\_platform\_cfg.h.

**12.37.2.128 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 128

07 128

Definition at line 1264 of file example\_platform\_cfg.h.

**12.37.2.129 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 160

08 160

Definition at line 1265 of file example\_platform\_cfg.h.

**12.37.2.130 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 256

09 256

Definition at line 1266 of file example\_platform\_cfg.h.

**12.37.2.131 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 512  
10 512  
Definition at line 1267 of file example\_platform\_cfg.h.

**12.37.2.132 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 1024  
11 1024  
Definition at line 1268 of file example\_platform\_cfg.h.

**12.37.2.133 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 2048  
12 2048  
Definition at line 1269 of file example\_platform\_cfg.h.

**12.37.2.134 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 4096  
13 4096  
Definition at line 1270 of file example\_platform\_cfg.h.

**12.37.2.135 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 8192  
14 8192  
Definition at line 1271 of file example\_platform\_cfg.h.

**12.37.2.136 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 16384  
15 16384  
Definition at line 1272 of file example\_platform\_cfg.h.

**12.37.2.137 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 32768  
16 32768  
Definition at line 1273 of file example\_platform\_cfg.h.

**12.37.2.138 CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY 64  
TY 64

**Purpose** Define SB Task Priority

**Description:**

Defines the cFE\_SB Task priority.

**Limits**

Not Applicable

Definition at line 1285 of file example\_platform\_cfg.h.

**12.37.2.139 CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define SB Task Stack Size

**Description:**

Defines the cFE\_SB Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1300 of file example\_platform\_cfg.h.

**12.37.2.140 CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES** #define CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES 524288

**Purpose** Size of Table Services Table Memory Pool

**Description:**

Defines the TOTAL size of the memory pool that cFE Table Services allocates from the system. The size must be large enough to provide memory for each registered table, the inactive buffers for double buffered tables and for the shared inactive buffers for single buffered tables.

**Limits**

The cFE does not place a limit on the size of this parameter.

Definition at line 1347 of file example\_platform\_cfg.h.

**12.37.2.141 CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE** #define CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE "/ram/cfe\_tbl\_reg.log"

**Purpose** Default Filename for a Table Registry Dump

**Description:**

Defines the file name used to store the table registry when no filename is specified in the dump registry command.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 1461 of file example\_platform\_cfg.h.

**12.37.2.142 CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES 32

**Purpose** Maximum Number of Critical Tables that can be Registered

**Description:**

Defines the maximum number of critical tables supported by this processor's Table Services.

**Limits**

This number must be less than 32767. It should be recognized that this parameter determines the size of the Critical Table Registry which is maintained in the Critical Data Store. An excessively high number will waste Critical Data Store memory. Therefore, this number must not exceed the value defined in CFE\_ES\_CDS\_MAX\_CRITICAL\_TABLES.

Definition at line 1402 of file example\_platform\_cfg.h.

**12.37.2.143 CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE** #define CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE 16384

**Purpose** Maximum Size Allowed for a Double Buffered Table

**Description:**

Defines the maximum allowed size (in bytes) of a double buffered table.

**Limits**

The cFE does not place a limit on the size of this parameter but it must be less than half of [CFE\\_PLATFORM\\_TBL\\_BUF\\_MEMORY\\_BLOCK\\_SIZE](#).

Definition at line 1359 of file example\_platform\_cfg.h.

**12.37.2.144 CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256

**Purpose** Maximum Number of Table Handles

**Description:**

Defines the maximum number of Table Handles.

**Limits**

This number must be less than 32767. This number must be at least as big as the number of tables ([CFE\\_PLATFORM\\_TBL\\_MAX\\_NUM\\_TABLES](#)) and should be set higher if tables are shared between applications.

Definition at line 1415 of file example\_platform\_cfg.h.

**12.37.2.145 CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128

**Purpose** Maximum Number of Tables Allowed to be Registered

Description:

Defines the maximum number of tables supported by this processor's Table Services.

Limits

This number must be less than 32767. It should be recognized that this parameter determines the size of the Table Registry. An excessively high number will waste memory.

Definition at line 1388 of file example\_platform\_cfg.h.

**12.37.2.146 CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDA←  
TIONS 10

**Purpose** Maximum Number of Simultaneous Table Validations

Description:

Defines the maximum number of pending validations that the Table Services can handle at any one time. When a table has a validation function, a validation request is made of the application to perform that validation. This number determines how many of those requests can be outstanding at any one time.

Limits

This number must be less than 32767. An excessively high number will degrade system performance and waste memory. A number less than 20 is suggested but not required.

Definition at line 1448 of file example\_platform\_cfg.h.

**12.37.2.147 CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS** #define CFE\_PLATFORM\_TBL\_MAX\_SIMUL←  
TANEOUS\_LOADS 4

**Purpose** Maximum Number of Simultaneous Loads to Support

Description:

Defines the maximum number of single buffered tables that can be loaded simultaneously. This number is used to determine the number of shared buffers to allocate.

Limits

This number must be less than 32767. An excessively high number will degrade system performance and waste memory. A number less than 5 is suggested but not required.

Definition at line 1430 of file example\_platform\_cfg.h.

```
12.37.2.148 CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE #define CFE_PLATFORM_TBL_MAX_SNGL_TABLE←  
_SIZE 16384
```

**Purpose** Maximum Size Allowed for a Single Buffered Table

Description:

Defines the maximum allowed size (in bytes) of a single buffered table. **NOTE:** This size determines the size of all shared table buffers. Therefore, this size will be multiplied by [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) below when allocating memory for shared tables.

Limits

The cFE does not place a limit on the size of this parameter but it must be small enough to allow for [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) number of tables to fit into [CFE\\_PLATFORM\\_TBL\\_BUF\\_MEMORY\\_BYT](#)

Definition at line 1375 of file example\_platform\_cfg.h.

```
12.37.2.149 CFE_PLATFORM_TBL_START_TASK_PRIORITY #define CFE_PLATFORM_TBL_START_TASK_PRIO←  
RITY 70
```

**Purpose** Define TBL Task Priority

Description:

Defines the cFE\_TBL Task priority.

Limits

Not Applicable

Definition at line 1316 of file example\_platform\_cfg.h.

```
12.37.2.150 CFE_PLATFORM_TBL_START_TASK_STACK_SIZE #define CFE_PLATFORM_TBL_START_TASK_S←  
TACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

**Purpose** Define TBL Task Stack Size

Description:

Defines the cFE\_TBL Task Stack Size

Limits

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1331 of file example\_platform\_cfg.h.

```
12.37.2.151 CFE_PLATFORM_TBL_U32FROM4CHARS #define CFE_PLATFORM_TBL_U32FROM4CHARS ( _C1, _C2, _C3, _C4 ) ((uint32) (_C1) << 24 | (uint32) (_C2) << 16 | (uint32) (_C3) << 8 | (uint32) (_C4))
```

Definition at line 1483 of file example\_platform\_cfg.h.

```
12.37.2.152 CFE_PLATFORM_TBL_VALID_PRID_1 #define CFE_PLATFORM_TBL_VALID_PRID_1 (1)
```

**Purpose** Processor ID values used for table load validation

**Description:**

Defines the processor ID values used for validating the processor ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

**Limits**

This value can be any 32 bit unsigned integer.

Definition at line 1532 of file example\_platform\_cfg.h.

```
12.37.2.153 CFE_PLATFORM_TBL_VALID_PRID_2 #define CFE_PLATFORM_TBL_VALID_PRID_2 (CFE_PLATFORM_TBL_U32FROM4CHARS ('b', 'c', 'd'))
```

Definition at line 1533 of file example\_platform\_cfg.h.

```
12.37.2.154 CFE_PLATFORM_TBL_VALID_PRID_3 #define CFE_PLATFORM_TBL_VALID_PRID_3 0
```

Definition at line 1534 of file example\_platform\_cfg.h.

```
12.37.2.155 CFE_PLATFORM_TBL_VALID_PRID_4 #define CFE_PLATFORM_TBL_VALID_PRID_4 0
```

Definition at line 1535 of file example\_platform\_cfg.h.

```
12.37.2.156 CFE_PLATFORM_TBL_VALID_PRID_COUNT #define CFE_PLATFORM_TBL_VALID_PRID_COUNT 0
```

**Purpose** Number of Processor ID's specified for validation

**Description:**

Defines the number of specified processor ID values that are verified during table loads. If the number is zero then no validation of the processor ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of processor ID's defined below are compared to the processor ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified processor ID values.

**Limits**

This number must be greater than or equal to zero and less than or equal to 4.

Definition at line 1518 of file example\_platform\_cfg.h.

**12.37.2.157 CFE\_PLATFORM\_TBL\_VALID\_SCID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 (0x42)

**Purpose** Spacecraft ID values used for table load validation

**Description:**

Defines the spacecraft ID values used for validating the spacecraft ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

**Limits**

This value can be any 32 bit unsigned integer.

Definition at line 1498 of file example\_platform\_cfg.h.

**12.37.2.158 CFE\_PLATFORM\_TBL\_VALID\_SCID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CH  
'b', 'c', 'd'))

Definition at line 1499 of file example\_platform\_cfg.h.

**12.37.2.159 CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT 0

**Purpose** Number of Spacecraft ID's specified for validation

**Description:**

Defines the number of specified spacecraft ID values that are verified during table loads. If the number is zero then no validation of the spacecraft ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of spacecraft ID's defined below are compared to the spacecraft ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified spacecraft ID values.

**Limits**

This number must be greater than or equal to zero and less than or equal to 2.

Definition at line 1480 of file example\_platform\_cfg.h.

**12.37.2.160 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY←  
TY 25

Definition at line 1729 of file example\_platform\_cfg.h.

**12.37.2.161 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STAC←  
K\_SIZE 8192

Definition at line 1748 of file example\_platform\_cfg.h.

**12.37.2.162 CFE\_PLATFORM\_TIME\_CFG\_CLIENT** #define CFE\_PLATFORM\_TIME\_CFG\_CLIENT false

Definition at line 1555 of file example\_platform\_cfg.h.

**12.37.2.163 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 1712 of file example\_platform\_cfg.h.

**12.37.2.164 CFE\_PLATFORM\_TIME\_CFG\_SERVER** #define CFE\_PLATFORM\_TIME\_CFG\_SERVER true

**Purpose** Time Server or Time Client Selection

**Description:**

This configuration parameter selects whether the Time task functions as a time "server" or "client". A time server generates the "time at the tone" packet which is received by time clients.

**Limits**

Enable one, and only one by defining either CFE\_PLATFORM\_TIME\_CFG\_SERVER or CFE\_PLATFORM\_TIME\_CFG\_CLIENT AS true. The other must be defined as false.

Definition at line 1554 of file example\_platform\_cfg.h.

**12.37.2.165 CFE\_PLATFORM\_TIME\_CFG\_SIGNAL** #define CFE\_PLATFORM\_TIME\_CFG\_SIGNAL false

**Purpose** Include or Exclude the Primary/Redundant Tone Selection Cmd

**Description:**

Depending on the specific hardware system configuration, it may be possible to switch between a primary and redundant tone signal. If supported by hardware, this definition will enable command interfaces to select the active tone signal. Both Time Clients and Time Servers support this feature. Note: Set the CFE\_PLATFORM\_TIME\_CFG\_SIGNAL define to true to enable tone signal commands.

**Limits**

Not Applicable

Definition at line 1602 of file example\_platform\_cfg.h.

**12.37.2.166 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 1622 of file example\_platform\_cfg.h.

**12.37.2.167 CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS false**

Definition at line 1639 of file example\_platform\_cfg.h.

**12.37.2.168 CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET false**

**Purpose** Choose the External Time Source for Server only

**Description:**

If [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) is set to true, then one of the following external time source types must also be set to true. Do not set any of the external time source types to true unless [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) is set to true.

**Limits**

1. If [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) is set to true then one and only one of the following three external time sources can and must be set true: [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_MET](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_GPS](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_TIME](#)
2. Only applies if [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) is set to true.

Definition at line 1638 of file example\_platform\_cfg.h.

**12.37.2.169 CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME false**

Definition at line 1640 of file example\_platform\_cfg.h.

**12.37.2.170 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 1699 of file example\_platform\_cfg.h.

**12.37.2.171 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 1687 of file example\_platform\_cfg.h.

**12.37.2.172 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 1587 of file example\_platform\_cfg.h.

**12.37.2.173 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 1659 of file example\_platform\_cfg.h.

**12.37.2.174 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS** #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SU←  
BS 500000

Definition at line 1660 of file example\_platform\_cfg.h.

**12.37.2.175 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 1672 of file example\_platform\_cfg.h.

**12.37.2.176 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS** #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS 0

Definition at line 1673 of file example\_platform\_cfg.h.

**12.37.2.177 CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_START\_TASK\_PR←  
ORITY 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 1727 of file example\_platform\_cfg.h.

**12.37.2.178 CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define TIME Task Stack Sizes

**Description:**

Defines the cFE\_TIME Main Task Stack Size Defines the cFE\_TIME Tone Task Stack Size Defines the cFE\_TIME 1HZ Task Stack Size

**Limits**

There is a lower limit of 2048 on these configuration parameters. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1746 of file example\_platform\_cfg.h.

**12.37.2.179 CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIO\_PRIORITY 25

Definition at line 1728 of file example\_platform\_cfg.h.

**12.37.2.180 CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096

Definition at line 1747 of file example\_platform\_cfg.h.

## 12.38 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

- `#define CFE_MISSION_TIME_SENDMET_PERF_ID 9`  
*Performance ID for 1 Hz Local ISR.*
- `#define CFE_MISSION_TIME_LOCAL1HZTASK_PERF_ID 10`  
*Performance ID for Time ToneSendMET.*
- `#define CFE_MISSION_TIME_TONE1HZTASK_PERF_ID 11`  
*Performance ID for 1 Hz Local Task.*
- `#define CFE_MISSION_TIME_TONE1HZTASK_PERF_ID 11`  
*Performance ID for 1 Hz Tone Task.*

### 12.38.1 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:

### 12.38.2 Macro Definition Documentation

#### 12.38.2.1 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.

#### 12.38.2.2 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.

#### 12.38.2.3 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.

#### 12.38.2.4 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.

#### 12.38.2.5 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.

#### 12.38.2.6 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.

#### 12.38.2.7 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.

**12.38.2.8 CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID** #define CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID ↵  
ID 8

Performance ID for 1 Hz Local ISR.

Definition at line 51 of file sample\_perfids.h.

**12.38.2.9 CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID** #define CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID ↵  
F\_ID 10

Performance ID for 1 Hz Local Task.

Definition at line 54 of file sample\_perfids.h.

**12.38.2.10 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.

**12.38.2.11 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.

**12.38.2.12 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.

**12.38.2.13 CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID** #define CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID ↵  
ID 11

Performance ID for 1 Hz Tone Task.

Definition at line 55 of file sample\_perfids.h.

- 12.39 [cfe/docs/src/cfe\\_api.dox File Reference](#)
- 12.40 [cfe/docs/src/cfe\\_es.dox File Reference](#)
- 12.41 [cfe/docs/src/cfe\\_evs.dox File Reference](#)
- 12.42 [cfe/docs/src/cfe\\_frontpage.dox File Reference](#)
- 12.43 [cfe/docs/src/cfe\\_glossary.dox File Reference](#)
- 12.44 [cfe/docs/src/cfe\\_sb.dox File Reference](#)
- 12.45 [cfe/docs/src/cfe\\_tbl.dox File Reference](#)
- 12.46 [cfe/docs/src/cfe\\_time.dox File Reference](#)
- 12.47 [cfe/docs/src/cfe\\_xref.dox File Reference](#)
- 12.48 [cfe/docs/src/cfs\\_versions.dox File Reference](#)
- 12.49 [cfe/modules/core\\_api/config/default\\_cfe\\_core\\_api\\_base\\_msgids.h File Reference](#)

#### 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*

#### 12.49.1 Detailed Description

Purpose: This header file contains the Message Id's for messages used by the cFE core.

#### 12.49.2 Macro Definition Documentation

##### 12.49.2.1 CFE\_PLATFORM\_CMD\_MID\_BASE `#define CFE_PLATFORM_CMD_MID_BASE 0x1800`

Platform command message ID base offset.

Example mechanism for setting default command bits and deconflicting MIDs across multiple platforms in a mission. For any sufficiently complex mission this method is typically replaced by a centralized message ID management scheme. 0x1800 - Nominal value for default message ID implementation (V1). This sets the command field and the secondary header present field. Typical V1 command MID range is 0x1800-1FFF. Additional cpus can deconflict message IDs by incrementing this value to provide sub-allocations (0x1900 for example). 0x0080 - Command bit for MISSION\_MSGID\_V2 message ID implementation (V2). Although this can be used for the value below due to the relatively small set of MIDs in the framework it will not scale so an alternative method of deconfliction is recommended.

Definition at line 47 of file `default_cfe_core_api_base_msgids.h`.

##### 12.49.2.2 CFE\_PLATFORM\_CMD\_MID\_BASE\_GLOB `#define CFE_PLATFORM_CMD_MID_BASE_GLOB 0x1860`

"Global" command message ID base offset

0x1860 - Nominal value for message ID V1 0x00E0 - Potential value for MISSION\_MSGID\_V2, note command bit is 0x0080. Works in limited cases only, alternative method of deconfliction is recommended. See [CFE\\_PLATFORM\\_CMD\\_MID\\_BASE](#) for more information

Definition at line 70 of file default\_cfe\_core\_api\_base\_msgids.h.

**12.49.2.3 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 59 of file default\_cfe\_core\_api\_base\_msgids.h.

## 12.50 cfe/modules/core\_api/config/default\_cfe\_core\_api\_interface\_cfg.h File Reference

### Macros

- #define CFE\_MISSION\_MAX\_PATH\_LEN 64
- #define CFE\_MISSION\_MAX\_FILE\_LEN 20
- #define CFE\_MISSION\_MAX\_API\_LEN 20
- #define CFE\_MISSION\_MAX\_NUM\_FILES 50

### 12.50.1 Detailed Description

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

### 12.50.2 Macro Definition Documentation

#### 12.50.2.1 CFE\_MISSION\_MAX\_API\_LEN #define CFE\_MISSION\_MAX\_API\_LEN 20

**Purpose** cFE Maximum length for API names within data exchange structures

##### Description:

The value of this constant dictates the size of filenames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_API\_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_API\_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_API\_LEN value.

This length must include an extra character for NULL termination.

##### Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 108 of file default\_cfe\_core\_api\_interface\_cfg.h.

**12.50.2.2 CFE\_MISSION\_MAX\_FILE\_LEN #define CFE\_MISSION\_MAX\_FILE\_LEN 20**

**Purpose** cFE Maximum length for filenames within data exchange structures

**Description:**

The value of this constant dictates the size of filenames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_FILE\_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_FILE\_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_FILE\_LEN value.

This length must include an extra character for NULL termination.

**Limits**

All CPUs within the same SB domain (mission) and ground tools must share the same definition. Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 82 of file default\_cfe\_core\_api\_interface\_cfg.h.

**12.50.2.3 CFE\_MISSION\_MAX\_NUM\_FILES #define CFE\_MISSION\_MAX\_NUM\_FILES 50**

**Purpose** cFE Maximum number of files in a message/data exchange

**Description:**

The value of this constant dictates the maximum number of files within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_NUM\_OPEN\_FILES but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_NUM\_OPEN\_FILES in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_NUM\_OPEN\_FILES value.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 130 of file default\_cfe\_core\_api\_interface\_cfg.h.

**12.50.2.4 CFE\_MISSION\_MAX\_PATH\_LEN #define CFE\_MISSION\_MAX\_PATH\_LEN 64**

**Purpose** cFE Maximum length for pathnames within data exchange structures

**Description:**

The value of this constant dictates the size of pathnames within all structures used for external data exchange, such as Software bus messages and table definitions. This is typically the same as OS\_MAX\_PATH\_LEN but that is OSAL dependent – and as such it definable on a per-processor/OS basis and hence may be different across multiple processors. It is recommended to set this to the value of the largest OS\_MAX\_PATH\_LEN in use on any CPU on the mission.

This affects only the layout of command/telemetry messages and table definitions; internal allocation may use the platform-specific OS\_MAX\_PATH\_LEN value.

This length must include an extra character for NULL termination.

**Limits**

All CPUs within the same SB domain (mission) and ground tools must share the same definition. Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 55 of file default\_cfe\_core\_api\_interface\_cfg.h.

## **12.51 cfe/modules/core\_api/config/default\_cfe\_mission\_cfg.h File Reference**

```
#include "cfe_core_api_interface_cfg.h"
#include "cfe_es_mission_cfg.h"
#include "cfe_evs_mission_cfg.h"
#include "cfe_sb_mission_cfg.h"
#include "cfe_tbl_mission_cfg.h"
#include "cfe_time_mission_cfg.h"
#include "cfe_fs_mission_cfg.h"
```

### **12.51.1 Detailed Description**

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

## **12.52 cfe/modules/core\_api/config/default\_cfe\_msgids.h File Reference**

```
#include "cfe_es_msgids.h"
#include "cfe_evs_msgids.h"
#include "cfe_sb_msgids.h"
#include "cfe_tbl_msgids.h"
#include "cfe_time_msgids.h"
```

### **12.52.1 Detailed Description**

Purpose: This header file contains the Message Id's for messages used by the cFE core.

## **12.53 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"
```

### 12.53.1 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.

## 12.54 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.*

### 12.54.1 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

### 12.54.2 Function Documentation

#### 12.54.2.1 CFE\_Config\_GetIdByName() `CFE_ConfigId_t CFE_Config_GetIdByName (` `const char * Name )`

Obtain the ID value associated with a configuration name.

**Parameters**

in	<i>Name</i>	The name of the ID to look up
----	-------------	-------------------------------

**Returns**

ID associated with name

**Return values**

<i>CFE_CONFIGID_UNDEFINED</i>	if the name did not correspond to a key
-------------------------------	---

**12.54.2.2 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	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

**Returns**

Name associated with key

**12.54.2.3 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	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

**Returns**

Value associated with key

## Return values

<code>NULL</code>	if key is not defined or not set
-------------------	----------------------------------

**12.54.2.4 CFE\_Config\_GetString()** `const char* CFE_Config_GetString ( CFE_ConfigId_t ConfigId )`

Obtain a string value correlating to an CFE configuration ID.

Retrieves the string value associated with the specified key.

If no value has been set, or the key is not valid, this returns the special string "UNDEFINED"

## Note

This function does not return NULL, so it can be used directly in printf-style calls.

## Parameters

<code>in</code>	<code>ConfigId</code>	Configuration ID/Key to look up
-----------------	-----------------------	---------------------------------

## Returns

String value associated with key

**12.54.2.5 CFE\_Config\_GetValue()** `uint32 CFE_Config_GetValue ( CFE_ConfigId_t ConfigId )`

Obtain an integer value correlating to an CFE configuration ID.

Retrieves the integer value associated with the specified key.

If no value has been set, or the key is not valid, this returns 0.

## Parameters

<code>in</code>	<code>ConfigId</code>	Configuration ID/Key to look up
-----------------	-----------------------	---------------------------------

## Returns

Value associated with key

## Return values

<code>0</code>	if key is not defined or not set
----------------	----------------------------------

**12.54.2.6 CFE\_Config\_IterateAll()** `void CFE_Config_IterateAll ( void * Arg, CFE_Config_Callback_t Callback )`

Iterate all known name/ID value pairs.

**Parameters**

in	<i>Arg</i>	User-supplied opaque argument to pass to callback
in	<i>Callback</i>	User-supplied callback function to invoke for each ID

**12.55 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)

**12.55.1 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

**12.55.2 Macro Definition Documentation**

**12.55.2.1 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.

**12.55.2.2 CFE\_CONFIGID\_UNDEFINED** #define CFE\_CONFIGID\_UNDEFINED CFE\_CONFIGID\_C(CFE\_RESOURCEID\_UNDEFINED)  
Definition at line 49 of file cfe\_config\_api\_typedefs.h.

**12.55.3 Typedef Documentation**

**12.55.3.1 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.

**12.55.3.2 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`.

## 12.56 cfe/modules/core\_api/fsw/inc/cfe\_endian.h File Reference

```
#include "common_types.h"
```

### Macros

- `#define CFE_MAKE_BIG16(n) (((n)&0x00FF) << 8) | (((n)&0xFF00) >> 8))`
- `#define CFE_MAKE_BIG32(n) (((n)&0x000000FF) << 24) | (((n)&0x0000FF00) << 8) | (((n)&0x00FF0000) >> 8) | (((n)&0xFF000000) >> 24))`

### 12.56.1 Detailed Description

Purpose: Define macros to enforce big-endian/network byte order for 16 and 32 bit integers

### 12.56.2 Macro Definition Documentation

**12.56.2.1 CFE\_MAKE\_BIG16** `#define CFE_MAKE_BIG16(`  
`n ) (((n)&0x00FF) << 8) | (((n)&0xFF00) >> 8))`

Definition at line 64 of file `cfe_endian.h`.

**12.56.2.2 CFE\_MAKE\_BIG32** `#define CFE_MAKE_BIG32(`  
`n ) (((n)&0x000000FF) << 24) | (((n)&0x0000FF00) << 8) | (((n)&0x00FF0000) >> 8) |`  
`((n)&0xFF000000) >> 24))`

Definition at line 65 of file `cfe_endian.h`.

## 12.57 cfe/modules/core\_api/fsw/inc/cfe\_error.h File Reference

```
#include "osapi.h"
```

### Macros

- `#define CFE_STATUS_C(X) ((CFE_Status_t)(X))`  
*cFE Status macro for literal*
- `#define CFE_STATUS_STRING_LENGTH 11`  
*cFE Status converted to string length limit*
- `#define CFE_SEVERITY_BITMASK ((CFE_Status_t)0xc0000000)`  
*Error Severity Bitmask.*
- `#define CFE_SEVERITY_SUCCESS ((CFE_Status_t)0x00000000)`  
*Severity Success.*
- `#define CFE_SEVERITY_INFO ((CFE_Status_t)0x40000000)`  
*Severity Info.*
- `#define CFE_SEVERITY_ERROR ((CFE_Status_t)0xc0000000)`

- Severity Error.*
- #define CFE\_SERVICE\_BITMASK ((CFE\_Status\_t)0x0e000000)  
*Error Service Bitmask.*
  - #define CFE\_EVENTS\_SERVICE ((CFE\_Status\_t)0x02000000)  
*Event Service.*
  - #define CFE\_EXECUTIVE\_SERVICE ((CFE\_Status\_t)0x04000000)  
*Executive Service.*
  - #define CFE\_FILE\_SERVICE ((CFE\_Status\_t)0x06000000)  
*File Service.*
  - #define CFE\_GENERIC\_SERVICE ((CFE\_Status\_t)0x08000000)  
*Generic Service.*
  - #define CFE\_SOFTWARE\_BUS\_SERVICE ((CFE\_Status\_t)0x0a000000)  
*Software Bus Service.*
  - #define CFE\_TABLE\_SERVICE ((CFE\_Status\_t)0x0c000000)  
*Table Service.*
  - #define CFE\_TIME\_SERVICE ((CFE\_Status\_t)0x0e000000)  
*Time Service.*
  - #define CFE\_SUCCESS ((CFE\_Status\_t)0)  
*Successful execution.*
  - #define CFE\_STATUS\_NO\_COUNTER\_INCREMENT ((CFE\_Status\_t)0x48000001)  
*No Counter Increment.*
  - #define CFE\_STATUS\_WRONG\_MSG\_LENGTH ((CFE\_Status\_t)0xc8000002)  
*Wrong Message Length.*
  - #define CFE\_STATUS\_UNKNOWN\_MSG\_ID ((CFE\_Status\_t)0xc8000003)  
*Unknown Message ID.*
  - #define CFE\_STATUS\_BAD\_COMMAND\_CODE ((CFE\_Status\_t)0xc8000004)  
*Bad Command Code.*
  - #define CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL ((CFE\_Status\_t)0xc8000005)  
*External failure.*
  - #define CFE\_STATUS\_REQUEST\_ALREADY\_PENDING ((int32)0xc8000006)  
*Request already pending.*
  - #define CFE\_STATUS\_VALIDATION\_FAILURE ((int32)0xc8000007)  
*Request or input value failed basic structural validation.*
  - #define CFE\_STATUS\_RANGE\_ERROR ((int32)0xc8000008)  
*Request or input value is out of range.*
  - #define CFE\_STATUS\_INCORRECT\_STATE ((int32)0xc8000009)  
*Cannot process request at this time.*
  - #define CFE\_STATUS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc800ffff)  
*Not Implemented.*
  - #define CFE\_EVS\_UNKNOWN\_FILTER ((CFE\_Status\_t)0xc2000001)  
*Unknown Filter.*
  - #define CFE\_EVS\_APP\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000002)  
*Application Not Registered.*
  - #define CFE\_EVS\_APP\_ILLEGAL\_APP\_ID ((CFE\_Status\_t)0xc2000003)  
*Illegal Application ID.*
  - #define CFE\_EVS\_APP\_FILTER\_OVERLOAD ((CFE\_Status\_t)0xc2000004)  
*Application Filter Overload.*

- #define CFE\_EVS\_RESET\_AREA\_POINTER ((CFE\_Status\_t)0xc2000005)  
*Reset Area Pointer Failure.*
- #define CFE\_EVS\_EVT\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000006)  
*Event Not Registered.*
- #define CFE\_EVS\_FILE\_WRITE\_ERROR ((CFE\_Status\_t)0xc2000007)  
*File Write Error.*
- #define CFE\_EVS\_INVALID\_PARAMETER ((CFE\_Status\_t)0xc2000008)  
*Invalid Pointer.*
- #define CFE\_EVS\_APP\_SQUELCHED ((CFE\_Status\_t)0xc2000009)  
*Event squelched.*
- #define CFE\_EVS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc200ffff)  
*Not Implemented.*
- #define CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID ((CFE\_Status\_t)0xc4000001)  
*Resource ID is not valid.*
- #define CFE\_ES\_ERR\_NAME\_NOT\_FOUND ((CFE\_Status\_t)0xc4000002)  
*Resource Name Error.*
- #define CFE\_ES\_ERR\_APP\_CREATE ((CFE\_Status\_t)0xc4000004)  
*Application Create Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_CREATE ((CFE\_Status\_t)0xc4000005)  
*Child Task Create Error.*
- #define CFE\_ES\_ERR\_SYS\_LOG\_FULL ((CFE\_Status\_t)0xc4000006)  
*System Log Full.*
- #define CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE ((CFE\_Status\_t)0xc4000008)  
*Memory Block Size Error.*
- #define CFE\_ES\_ERR\_LOAD\_LIB ((CFE\_Status\_t)0xc4000009)  
*Load Library Error.*
- #define CFE\_ES\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc400000a)  
*Bad Argument.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER ((CFE\_Status\_t)0xc400000b)  
*Child Task Register Error.*
- #define CFE\_ES\_CDS\_ALREADY\_EXISTS ((CFE\_Status\_t)0x4400000d)  
*CDS Already Exists.*
- #define CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY ((CFE\_Status\_t)0xc400000e)  
*CDS Insufficient Memory.*
- #define CFE\_ES\_CDS\_INVALID\_NAME ((CFE\_Status\_t)0xc400000f)  
*CDS Invalid Name.*
- #define CFE\_ES\_CDS\_INVALID\_SIZE ((CFE\_Status\_t)0xc4000010)  
*CDS Invalid Size.*
- #define CFE\_ES\_CDS\_INVALID ((CFE\_Status\_t)0xc4000012)  
*CDS Invalid.*
- #define CFE\_ES\_CDS\_ACCESS\_ERROR ((CFE\_Status\_t)0xc4000013)  
*CDS Access Error.*
- #define CFE\_ES\_FILE\_IO\_ERR ((CFE\_Status\_t)0xc4000014)  
*File IO Error.*
- #define CFE\_ES\_RST\_ACCESS\_ERR ((CFE\_Status\_t)0xc4000015)  
*Reset Area Access Error.*
- #define CFE\_ES\_ERR\_APP\_REGISTER ((CFE\_Status\_t)0xc4000017)

- #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)

- `#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  
*cFE Status type for readability and eventually type safety*
- typedef char CFE\_StatusString\_t[CFE\_STATUS\_STRING\_LENGTH]  
*For the `CFE_ES_StatusToString()` function, to ensure everyone is making an array of the same length.*

## Functions

- char \* CFE\_ES\_StatusToString (CFE\_Status\_t status, CFE\_StatusString\_t \*status\_string)  
*Convert status to a string.*

### 12.57.1 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

### 12.57.2 Macro Definition Documentation

#### 12.57.2.1 CFE\_EVENTS\_SERVICE `#define CFE_EVENTS_SERVICE ((CFE_Status_t)0x02000000)`

Event Service.

Definition at line 126 of file cfe\_error.h.

#### 12.57.2.2 CFE\_EXECUTIVE\_SERVICE `#define CFE_EXECUTIVE_SERVICE ((CFE_Status_t)0x04000000)`

Executive Service.

Definition at line 127 of file cfe\_error.h.

#### 12.57.2.3 CFE\_FILE\_SERVICE `#define CFE_FILE_SERVICE ((CFE_Status_t)0x06000000)`

File Service.

Definition at line 128 of file cfe\_error.h.

#### 12.57.2.4 CFE\_GENERIC\_SERVICE `#define CFE_GENERIC_SERVICE ((CFE_Status_t)0x08000000)`

Generic Service.

Definition at line 129 of file cfe\_error.h.

#### 12.57.2.5 CFE\_SERVICE\_BITMASK `#define CFE_SERVICE_BITMASK ((CFE_Status_t)0x0e000000)`

Error Service Bitmask.

Definition at line 124 of file cfe\_error.h.

#### 12.57.2.6 CFE\_SEVERITY\_BITMASK `#define CFE_SEVERITY_BITMASK ((CFE_Status_t)0xc0000000)`

Error Severity Bitmask.

Definition at line 115 of file cfe\_error.h.

#### 12.57.2.7 CFE\_SEVERITY\_ERROR `#define CFE_SEVERITY_ERROR ((CFE_Status_t)0xc0000000)`

Severity Error.

Definition at line 119 of file cfe\_error.h.

#### 12.57.2.8 CFE\_SEVERITY\_INFO `#define CFE_SEVERITY_INFO ((CFE_Status_t)0x40000000)`

Severity Info.

Definition at line 118 of file cfe\_error.h.

**12.57.2.9 CFE\_SEVERITY\_SUCCESS** #define CFE\_SEVERITY\_SUCCESS ((CFE\_Status\_t) 0x00000000)  
Severity Success.  
Definition at line 117 of file cfe\_error.h.

**12.57.2.10 CFE\_SOFTWARE\_BUS\_SERVICE** #define CFE\_SOFTWARE\_BUS\_SERVICE ((CFE\_Status\_t) 0x0a000000)  
Software Bus Service.  
Definition at line 130 of file cfe\_error.h.

**12.57.2.11 CFE\_STATUS\_C** #define CFE\_STATUS\_C ( X ) ( (CFE\_Status\_t) (X) )  
cFE Status macro for literal  
Definition at line 48 of file cfe\_error.h.

**12.57.2.12 CFE\_STATUS\_STRING\_LENGTH** #define CFE\_STATUS\_STRING\_LENGTH 11  
cFE Status converted to string length limit  
Used for sizing CFE\_StatusString\_t intended for use in printing CFE\_Status\_t values Sized for 0x%08x and NULL  
Definition at line 56 of file cfe\_error.h.

**12.57.2.13 CFE\_TABLE\_SERVICE** #define CFE\_TABLE\_SERVICE ((CFE\_Status\_t) 0x0c000000)  
Table Service.  
Definition at line 131 of file cfe\_error.h.

**12.57.2.14 CFE\_TIME\_SERVICE** #define CFE\_TIME\_SERVICE ((CFE\_Status\_t) 0x0e000000)  
Time Service.  
Definition at line 132 of file cfe\_error.h.

### 12.57.3 Typedef Documentation

**12.57.3.1 CFE\_Status\_t** typedef int32 CFE\_Status\_t  
cFE Status type for readability and eventually type safety  
Definition at line 43 of file cfe\_error.h.

**12.57.3.2 CFE\_StatusString\_t** typedef char CFE\_StatusString\_t [CFE\_STATUS\_STRING\_LENGTH]  
For the [CFE\\_ES\\_StatusToString\(\)](#) function, to ensure everyone is making an array of the same length.  
Definition at line 62 of file cfe\_error.h.

### 12.57.4 Function Documentation

**12.57.4.1 CFE\_ES\_StatusToString()** char\* CFE\_ES\_StatusToString ( CFE\_Status\_t status, CFE\_StatusString\_t \* status\_string )  
Convert status to a string.

**Parameters**

in	<i>status</i>	Status value to convert
out	<i>status_string</i>	Buffer to store status converted to string

**Returns**

Passed in string pointer

## 12.58 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 Modeld, 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 \*AppldPtr)  
*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 \*AppldPtr, 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 Appld, 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 Appld)  
*Get Application Information given a specified App ID.*
  - `CFE_Status_t CFE_ES_GetTaskInfo` (CFE\_ES\_TaskInfo\_t \*TaskInfo, CFE\_ES\_TaskId\_t TaskId)  
*Get Task Information given a specified Task ID.*
  - int32 `CFE_ES_GetLibInfo` (CFE\_ES\_AppInfo\_t \*LibInfo, CFE\_ES\_LibId\_t LibId)  
*Get Library Information given a specified Resource ID.*
  - int32 `CFE_ES_GetModuleInfo` (CFE\_ES\_AppInfo\_t \*ModuleInfo, CFE\_Resourceld\_t Resourceld)  
*Get Information given a specified Resource ID.*
  - `CFE_Status_t CFE_ES_CreateChildTask` (CFE\_ES\_TaskId\_t \*TaskIdPtr, const char \*TaskName, CFE\_ES\_ChildTaskMainFuncPtr FunctionPtr, CFE\_ES\_StackPointer\_t StackPtr, size\_t StackSize, CFE\_ES\_TaskPriority\_Atom\_t Priority, uint32 Flags)  
*Creates a new task under an existing Application.*
  - `CFE_Status_t CFE_ES_GetTaskIDByName` (CFE\_ES\_TaskId\_t \*TaskIdPtr, const char \*TaskName)  
*Get a Task ID associated with a specified Task name.*
  - `CFE_Status_t CFE_ES_GetTaskName` (char \*TaskName, CFE\_ES\_TaskId\_t TaskId, size\_t BufferLength)  
*Get a Task name for a specified Task ID.*
  - `CFE_Status_t CFE_ES_DeleteChildTask` (CFE\_ES\_TaskId\_t TaskId)  
*Deletes a task under an existing Application.*
  - void `CFE_ES_ExitChildTask` (void)  
*Exits a child task.*
  - void `CFE_ES_BackgroundWakeUp` (void)  
*Wakes up the CFE background task.*
  - `CFE_Status_t CFE_ES_WriteToSysLog` (const char \*SpecStringPtr,...) OS\_PRINTF(1  
*Write a string to the cFE System Log.*

- `CFE_Status_t uint32 CFE_ES_CalculateCRC (const void *DataPtr, size_t DataLength, uint32 InputCRC, CFE_ES_CrcType_Enum_t TypeCRC)`  
*Calculate a CRC on a block of memory.*
- `void CFE_ES_ProcessAsyncEvent (void)`  
*Notification that an asynchronous event was detected by the underlying OS/PSP.*
- `CFE_Status_t CFE_ES_RegisterCDS (CFE_ES_CDSHandle_t *CDSHandlePtr, size_t BlockSize, const char *Name)`  
*Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)*
- `CFE_Status_t CFE_ES_GetCDSBlockIDByName (CFE_ES_CDSHandle_t *BlockIdPtr, const char *BlockName)`  
*Get a CDS Block ID associated with a specified CDS Block name.*
- `CFE_Status_t CFE_ES_GetCDSBlockName (char *BlockName, CFE_ES_CDSHandle_t BlockId, size_t BufferLength)`  
*Get a Block name for a specified Block ID.*
- `CFE_Status_t CFE_ES_CopyToCDS (CFE_ES_CDSHandle_t Handle, const void *DataToCopy)`  
*Save a block of data in the Critical Data Store (CDS)*
- `CFE_Status_t CFE_ES_RestoreFromCDS (void *RestoreToMemory, CFE_ES_CDSHandle_t Handle)`  
*Recover a block of data from the Critical Data Store (CDS)*
- `CFE_Status_t CFE_ES_PoolCreateNoSem (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size)`  
*Initializes a memory pool created by an application without using a semaphore during processing.*
- `CFE_Status_t CFE_ES_PoolCreate (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size)`  
*Initializes a memory pool created by an application while using a semaphore during processing.*
- `CFE_Status_t CFE_ES_PoolCreateEx (CFE_ES_MemHandle_t *PoolID, void *MemPtr, size_t Size, uint16 NumBlockSizes, const size_t *BlockSizes, bool UseMutex)`  
*Initializes a memory pool created by an application with application specified block sizes.*
- `int32 CFE_ES_PoolDelete (CFE_ES_MemHandle_t PoolID)`  
*Deletes a memory pool that was previously created.*
- `int32 CFE_ES_GetPoolBuf (CFE_ES_MemPoolBuf_t *BufPtr, CFE_ES_MemHandle_t Handle, size_t Size)`  
*Gets a buffer from the memory pool created by `CFE_ES_PoolCreate` or `CFE_ES_PoolCreateNoSem`.*
- `CFE_Status_t CFE_ES_GetPoolBufInfo (CFE_ES_MemHandle_t Handle, CFE_ES_MemPoolBuf_t BufPtr)`  
*Gets info on a buffer previously allocated via `CFE_ES_GetPoolBuf`.*
- `int32 CFE_ES_PutPoolBuf (CFE_ES_MemHandle_t Handle, CFE_ES_MemPoolBuf_t BufPtr)`  
*Releases a buffer from the memory pool that was previously allocated via `CFE_ES_GetPoolBuf`.*
- `CFE_Status_t CFE_ES_GetMemPoolStats (CFE_ES_MemPoolStats_t *BufPtr, CFE_ES_MemHandle_t Handle)`  
*Extracts the statistics maintained by the memory pool software.*
- `void CFE_ES_PerfLogAdd (uint32 Marker, uint32 EntryExit)`  
*Adds a new entry to the data buffer.*
- `CFE_Status_t CFE_ES_RegisterGenCounter (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`  
*Register a generic counter.*
- `CFE_Status_t CFE_ES_DeleteGenCounter (CFE_ES_CounterId_t CounterId)`  
*Delete a generic counter.*
- `CFE_Status_t CFE_ES_IncrementGenCounter (CFE_ES_CounterId_t CounterId)`  
*Increments the specified generic counter.*
- `CFE_Status_t CFE_ES_SetGenCount (CFE_ES_CounterId_t CounterId, uint32 Count)`  
*Set the specified generic counter.*
- `CFE_Status_t CFE_ES_GetGenCount (CFE_ES_CounterId_t CounterId, uint32 *Count)`  
*Get the specified generic counter count.*
- `CFE_Status_t CFE_ES_GetGenCounterIDByName (CFE_ES_CounterId_t *CounterIdPtr, const char *CounterName)`

*Get the Id associated with a generic counter name.*

- **CFE\_Status\_t CFE\_ES\_GetGenCounterName** (char \*CounterName, CFE\_ES\_CounterId\_t CounterId, size\_t BufferLength)

*Get a Counter name for a specified Counter ID.*

### 12.58.1 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:

### 12.58.2 Macro Definition Documentation

**12.58.2.1 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.

**12.58.2.2 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.

**12.58.2.3 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.

**12.58.2.4 OS\_PRINTF** #define OS\_PRINTF(  
    m,  
    n )

Definition at line 50 of file cfe\_es.h.

## 12.59 cfe/modules/core\_api/fsw/inc/cfe\_es\_api\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_es_extern_typedefs.h"
```

### Data Structures

- union **CFE\_ES\_PoolAlign**

*Pool Alignment.*

## Macros

- `#define CFE_ES_STATIC_POOL_TYPE(size)`  
*Static Pool Type.*
- `#define CFE_ES_MEMPOOLBUF_C(x) ((CFE_ES_MemPoolBuf_t)(x))`  
*Conversion macro to create buffer pointer from another type.*
- `#define CFE_ES_NO_MUTEX false`  
*Indicates that the memory pool selection will not use a semaphore.*
- `#define CFE_ES_USE_MUTEX true`  
*Indicates that the memory pool selection will use a semaphore.*

## Reset Type extensions

- `#define CFE_ES_APP_RESTART CFE_PSP_RST_TYPE_MAX`

## Conversions for ES resource IDs

- `#define CFE_ES_APPID_C(val) ((CFE_ES_AppId_t)CFE_RESOURCEID_WRAP(val))`
- `#define CFE_ES_TASKID_C(val) ((CFE_ES_TaskId_t)CFE_RESOURCEID_WRAP(val))`
- `#define CFE_ES_LIBID_C(val) ((CFE_ES_LibId_t)CFE_RESOURCEID_WRAP(val))`
- `#define CFE_ES_COUNTERID_C(val) ((CFE_ES_CounterId_t)CFE_RESOURCEID_WRAP(val))`
- `#define CFE_ES_MEMHANDLE_C(val) ((CFE_ES_MemHandle_t)CFE_RESOURCEID_WRAP(val))`
- `#define CFE_ES_CDSHANDLE_C(val) ((CFE_ES_CDSHandle_t)CFE_RESOURCEID_WRAP(val))`

## Type-specific initializers for "undefined" resource IDs

- `#define CFE_ES_APPID_UNDEFINED CFE_ES_APPID_C(CFE_RESOURCEID_UNDEFINED)`
- `#define CFE_ES_TASKID_UNDEFINED CFE_ES_TASKID_C(CFE_RESOURCEID_UNDEFINED)`
- `#define CFE_ES_LIBID_UNDEFINED CFE_ES_LIBID_C(CFE_RESOURCEID_UNDEFINED)`
- `#define CFE_ES_COUNTERID_UNDEFINED CFE_ES_COUNTERID_C(CFE_RESOURCEID_UNDEFINED)`
- `#define CFE_ES_MEMHANDLE_UNDEFINED CFE_ES_MEMHANDLE_C(CFE_RESOURCEID_UNDEFINED)`
- `#define CFE_ES_CDS_BAD_HANDLE CFE_ES_CDSHANDLE_C(CFE_RESOURCEID_UNDEFINED)`

## Task Stack Constants

- `#define CFE_ES_TASK_STACK_ALLOCATE NULL /* aka OS_TASK_STACK_ALLOCATE in proposed O↔SAL change */`  
*Indicates that the stack for the child task should be dynamically allocated.*

## Typedefs

- `typedef void(* CFE_ES_TaskEntryFuncPtr_t) (void)`  
*Required Prototype of Task Main Functions.*
- `typedef int32(* CFE_ES_LibraryEntryFuncPtr_t) (CFE_ES_LibId_t LibId)`  
*Required Prototype of Library Initialization Functions.*
- `typedef CFE_ES_TaskEntryFuncPtr_t CFE_ES_ChildTaskMainFuncPtr_t`  
*Compatible typedef for ES child task entry point.*
- `typedef void * CFE_ES_StackPointer_t`  
*Type for the stack pointer of tasks.*
- `typedef enum CFE_ES_CrcType_Enum CFE_ES_CrcType_Enum_t`  
*Checksum/CRC algorithm identifiers.*
- `typedef union CFE_ES_PoolAlign CFE_ES_PoolAlign_t`  
*Pool Alignment.*
- `typedef void * CFE_ES_MemPoolBuf_t`  
*Pointer type used for memory pool API.*

## Enumerations

- enum **CFE\_ES\_CrcType\_Enum** { **CFE\_ES\_CrcType\_CRC\_8** = 1, **CFE\_ES\_CrcType\_CRC\_16** = 2, **CFE\_ES\_CrcType\_CRC\_32** = 3 }

*Checksum/CRC algorithm identifiers.*

### 12.59.1 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:

### 12.59.2 Macro Definition Documentation

#### 12.59.2.1 **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.

#### 12.59.2.2 **CFE\_ES\_APPID\_C** #define CFE\_ES\_APPID\_C(     val ) ((**CFE\_ES\_AppId\_t**)CFE\_RESOURCEID\_WRAP(val))

Definition at line 168 of file cfe\_es\_api\_typedefs.h.

#### 12.59.2.3 **CFE\_ES\_APPID\_UNDEFINED** #define CFE\_ES\_APPID\_UNDEFINED CFE\_ES\_APPID\_C(CFE\_RESOURCEID\_UNDEFINED)

Definition at line 180 of file cfe\_es\_api\_typedefs.h.

#### 12.59.2.4 **CFE\_ES\_CDS\_BAD\_HANDLE** #define CFE\_ES\_CDS\_BAD\_HANDLE CFE\_ES\_CDSHANDLE\_C(CFE\_RESOURCEID\_UNDEFINED)

Definition at line 185 of file cfe\_es\_api\_typedefs.h.

#### 12.59.2.5 **CFE\_ES\_CDSHANDLE\_C** #define CFE\_ES\_CDSHANDLE\_C(     val ) ((**CFE\_ES\_CDSHandle\_t**)CFE\_RESOURCEID\_WRAP(val))

Definition at line 173 of file cfe\_es\_api\_typedefs.h.

#### 12.59.2.6 **CFE\_ES\_COUNTERID\_C** #define CFE\_ES\_COUNTERID\_C(     val ) ((**CFE\_ES\_CounterId\_t**)CFE\_RESOURCEID\_WRAP(val))

Definition at line 171 of file cfe\_es\_api\_typedefs.h.

#### 12.59.2.7 **CFE\_ES\_COUNTERID\_UNDEFINED** #define CFE\_ES\_COUNTERID\_UNDEFINED CFE\_ES\_COUNTERID\_C(CFE\_RESOURCEID\_UNDEFINED)

Definition at line 183 of file cfe\_es\_api\_typedefs.h.

#### 12.59.2.8 **CFE\_ES\_LIBID\_C** #define CFE\_ES\_LIBID\_C(     val ) ((**CFE\_ES\_LibId\_t**)CFE\_RESOURCEID\_WRAP(val))

Definition at line 170 of file cfe\_es\_api\_typedefs.h.

**12.59.2.9 CFE\_ES\_LIBID\_UNDEFINED** #define CFE\_ES\_LIBID\_UNDEFINED CFE\_ES\_LIBID\_C(CFE\_RESOURCEID\_UNDEFINED)  
Definition at line 182 of file cfe\_es\_api\_typedefs.h.

**12.59.2.10 CFE\_ES\_MEMHANDLE\_C** #define CFE\_ES\_MEMHANDLE\_C(  
    val) ((CFE\_ES\_MemHandle\_t)CFE\_RESOURCEID\_WRAP(val))  
Definition at line 172 of file cfe\_es\_api\_typedefs.h.

**12.59.2.11 CFE\_ES\_MEMHANDLE\_UNDEFINED** #define CFE\_ES\_MEMHANDLE\_UNDEFINED CFE\_ES\_MEMHANDLE\_C(CFE\_RESOURCEID\_UNDEFINED)  
Definition at line 184 of file cfe\_es\_api\_typedefs.h.

**12.59.2.12 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 153 of file cfe\_es\_api\_typedefs.h.

**12.59.2.13 CFE\_ES\_NO\_MUTEX** #define CFE\_ES\_NO\_MUTEX false

Indicates that the memory pool selection will not use a semaphore.

Definition at line 200 of file cfe\_es\_api\_typedefs.h.

**12.59.2.14 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 120 of file cfe\_es\_api\_typedefs.h.

**12.59.2.15 CFE\_ES\_TASK\_STACK\_ALLOCATE** #define CFE\_ES\_TASK\_STACK\_ALLOCATE NULL /\* aka OS\_TA←SK\_STACK\_ALLOCATE in proposed OSAL change \*/

Indicates that the stack for the child task should be dynamically allocated.

This value may be supplied as the Stack Pointer argument to CFE\_ES\_ChildTaskCreate() to indicate that the stack should be dynamically allocated.

Definition at line 197 of file cfe\_es\_api\_typedefs.h.

**12.59.2.16 CFE\_ES\_TASKID\_C** #define CFE\_ES\_TASKID\_C(  
    val ) (([CFE\\_ES\\_TaskId\\_t](#))CFE\_RESOURCEID\_WRAP(val))

Definition at line 169 of file cfe\_es\_api\_typedefs.h.

**12.59.2.17 CFE\_ES\_TASKID\_UNDEFINED** #define CFE\_ES\_TASKID\_UNDEFINED [CFE\\_ES\\_TASKID\\_C\(CFE\\_RESOURCEID\\_UNDEFINED\)](#)

Definition at line 181 of file cfe\_es\_api\_typedefs.h.

**12.59.2.18 CFE\_ES\_USE\_MUTEX** #define CFE\_ES\_USE\_MUTEX true

Indicates that the memory pool selection will use a semaphore.

Definition at line 201 of file cfe\_es\_api\_typedefs.h.

### 12.59.3 Typedef Documentation

**12.59.3.1 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.

**12.59.3.2 CFE\_ES\_CrcType\_Enum\_t** [typedef enum CFE\\_ES\\_CrcType\\_Enum CFE\\_ES\\_CrcType\\_Enum\\_t](#)

CHECKSUM/CRC algorithm identifiers.

Currently only CFE\_ES\_CrcType\_CRC\_16 is supported.

**12.59.3.3 CFE\_ES\_LibraryEntryFuncPtr\_t** [typedef int32 \(\\* CFE\\_ES\\_LibraryEntryFuncPtr\\_t\) \(CFE\\_ES\\_LibId\\_t LibId\)](#)

Required Prototype of Library Initialization Functions.

Definition at line 69 of file cfe\_es\_api\_typedefs.h.

**12.59.3.4 CFE\_ES\_MemPoolBuf\_t** [typedef void\\* CFE\\_ES\\_MemPoolBuf\\_t](#)

Pointer type used for memory pool API.

This is used in the Get/Put API calls to refer to a pool buffer.

This pointer is expected to be type cast to the real object type after getting a new buffer. Using void\* allows this type conversion to occur easily.

#### Note

Older versions of CFE implemented the API using a uint32\*, which required explicit type casting everywhere it was called. Although the API type is now void\* to make usage easier, the pool buffers are aligned to machine requirements - typically 64 bits.

Definition at line 141 of file cfe\_es\_api\_typedefs.h.

**12.59.3.5 CFE\_ES\_PoolAlign\_t** [typedef union CFE\\_ES\\_PoolAlign CFE\\_ES\\_PoolAlign\\_t](#)

Pool Alignment.

Union that can be used for minimum memory alignment of ES memory pools on the target. It contains the longest native data types such that the alignment of this structure should reflect the largest possible alignment requirements for any data on this processor.

**12.59.3.6 CFE\_ES\_StackPointer\_t** `typedef void* CFE_ES_StackPointer_t`

Type for the stack pointer of tasks.

This type is used in the CFE ES task API.

Definition at line 84 of file `cfe_es_api_typedefs.h`.

**12.59.3.7 CFE\_ES\_TaskEntryFuncPtr\_t** `typedef void(* CFE_ES_TaskEntryFuncPtr_t) (void)`

Required Prototype of Task Main Functions.

Definition at line 68 of file `cfe_es_api_typedefs.h`.

## 12.59.4 Enumeration Type Documentation

**12.59.4.1 CFE\_ES\_CrcType\_Enum** `enum CFE_ES_CrcType_Enum`

Checksum/CRC algorithm identifiers.

Currently only `CFE_ES_CrcType_CRC_16` is supported.

### Enumerator

<code>CFE_ES_CrcType_CRC_8</code>	CRC ( 8 bit additive - returns 32 bit total) (Not currently implemented)
<code>CFE_ES_CrcType_CRC_16</code>	CRC (16 bit additive - returns 32 bit total)
<code>CFE_ES_CrcType_CRC_32</code>	CRC (32 bit additive - returns 32 bit total) (Not currently implemented)

Definition at line 91 of file `cfe_es_api_typedefs.h`.

## 12.60 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.*

### 12.60.1 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

### 12.60.2 Macro Definition Documentation

**12.60.2.1 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.

**12.60.2.2 CFE\_EVS\_SendCrit** `#define CFE_EVS_SendCrit(`  
    `E,`  
    `... ) CFE_EVS_Send(E, CRITICAL, __VA_ARGS__)`

Definition at line 50 of file cfe\_evs.h.

**12.60.2.3 CFE\_EVS\_SendDbg** `#define CFE_EVS_SendDbg(`  
    `E,`  
    `... ) CFE_EVS_Send(E, DEBUG, __VA_ARGS__)`

Definition at line 47 of file cfe\_evs.h.

**12.60.2.4 CFE\_EVS\_SendErr** `#define CFE_EVS_SendErr(`  
    `E,`  
    `... ) CFE_EVS_Send(E, ERROR, __VA_ARGS__)`

Definition at line 49 of file cfe\_evs.h.

**12.60.2.5 CFE\_EVS\_SendInfo** `#define CFE_EVS_SendInfo(`  
    `E,`  
    `... ) CFE_EVS_Send(E, INFORMATION, __VA_ARGS__)`

Definition at line 48 of file cfe\_evs.h.

## 12.61 cfe/modules/core\_api/fsw/inc/cfe\_evs\_api\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_evs_extern_typedefs.h"
```

### Data Structures

- struct [CFE\\_EVS\\_BinFilter](#)

*Event message filter definition structure.*

### Macros

#### Common Event Filter Mask Values

*Message is sent if (previous event count) & MASK == 0*

- #define [CFE\\_EVS\\_NO\\_FILTER](#) 0x0000  
*Stops any filtering. All messages are sent.*
- #define [CFE\\_EVS\\_FIRST\\_ONE\\_STOP](#) 0xFFFF  
*Sends the first event. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_TWO\\_STOP](#) 0xFFFE  
*Sends the first 2 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_4\\_STOP](#) 0xFFFFC  
*Sends the first 4 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_8\\_STOP](#) 0xFFF8  
*Sends the first 8 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_16\\_STOP](#) 0xFFF0  
*Sends the first 16 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_32\\_STOP](#) 0xFFE0  
*Sends the first 32 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_64\\_STOP](#) 0xFFC0  
*Sends the first 64 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_EVERY\\_OTHER\\_ONE](#) 0x0001  
*Sends every other event.*
- #define [CFE\\_EVS\\_EVERY\\_OTHER\\_TWO](#) 0x0002  
*Sends two, filters one, sends two, filters one, etc.*
- #define [CFE\\_EVS\\_EVERY\\_FOURTH\\_ONE](#) 0x0003  
*Sends every fourth event message. All others are filtered.*

### Typedefs

- typedef struct [CFE\\_EVS\\_BinFilter](#) [CFE\\_EVS\\_BinFilter\\_t](#)  
*Event message filter definition structure.*

#### 12.61.1 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

#### 12.61.2 Macro Definition Documentation

**12.61.2.1 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.

**12.61.2.2 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.

**12.61.2.3 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.

**12.61.2.4 CFE\_EVS\_FIRST\_16\_STOP** #define CFE\_EVS\_FIRST\_16\_STOP 0xFFFF0  
Sends the first 16 events. All remaining messages are filtered.  
Definition at line 49 of file cfe\_evs\_api\_typedefs.h.

**12.61.2.5 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.

**12.61.2.6 CFE\_EVS\_FIRST\_4\_STOP** #define CFE\_EVS\_FIRST\_4\_STOP 0xFFFFC  
Sends the first 4 events. All remaining messages are filtered.  
Definition at line 47 of file cfe\_evs\_api\_typedefs.h.

**12.61.2.7 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.

**12.61.2.8 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.

**12.61.2.9 CFE\_EVS\_FIRST\_ONE\_STOP** #define CFE\_EVS\_FIRST\_ONE\_STOP 0xFFFFF  
Sends the first event. All remaining messages are filtered.  
Definition at line 45 of file cfe\_evs\_api\_typedefs.h.

**12.61.2.10 CFE\_EVS\_FIRST\_TWO\_STOP** #define CFE\_EVS\_FIRST\_TWO\_STOP 0xFFFFE  
Sends the first 2 events. All remaining messages are filtered.  
Definition at line 46 of file cfe\_evs\_api\_typedefs.h.

**12.61.2.11 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.

### 12.61.3 Typedef Documentation

**12.61.3.1 CFE\_EVS\_BinFilter\_t** typedef struct CFE\_EVS\_BinFilter CFE\_EVS\_BinFilter\_t  
Event message filter definition structure.

## 12.62 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\_BackgroundFileDumpsPending (const CFE\_FS\_FileWriteMetaData\_t \*Meta)**  
*Query if a background file write request is currently pending.*

### 12.62.1 Detailed Description

Purpose: cFE File Services (FS) library API header file  
 Author: S.Walling/Microtel

## 12.63 cfe/modules/core\_api/fsw/inc/cfe\_fs\_api\_typedefs.h File Reference

```
#include "common_types.h"
#include "osconfig.h"
#include "cfe_fs_extern_typedefs.h"
```

### Data Structures

- struct [CFE\\_FS\\_FileWriteMetaData](#)

*External Metadata/State object associated with background file writes.*

### TypeDefs

- typedef bool(\* [CFE\\_FS\\_FileWriteGetData\\_t](#)) (void \*Meta, uint32 RecordNum, void \*\*Buffer, size\_t \*BufSize)
- typedef void(\* [CFE\\_FS\\_FileWriteOnEvent\\_t](#)) (void \*Meta, [CFE\\_FS\\_FileWriteEvent\\_t](#) Event, int32 Status, uint32 RecordNum, size\_t BlockSize, size\_t Position)
- typedef struct [CFE\\_FS\\_FileWriteMetaData](#) [CFE\\_FS\\_FileWriteMetaData\\_t](#)

*External Metadata/State object associated with background file writes.*

### Enumerations

- enum [CFE\\_FS\\_FileCategory\\_t](#) {
 [CFE\\_FS\\_FileCategory\\_UNDEFINED](#), [CFE\\_FS\\_FileCategory\\_DYNAMIC\\_MODULE](#), [CFE\\_FS\\_FileCategory\\_BINARY\\_DATA\\_DUMP](#),
 [CFE\\_FS\\_FileCategory\\_TEXT\\_LOG](#),
 [CFE\\_FS\\_FileCategory\\_SCRIPT](#), [CFE\\_FS\\_FileCategory\\_TEMP](#), [CFE\\_FS\\_FileCategory\\_MAX](#) }
- Generalized file types/categories known to FS.*
- enum [CFE\\_FS\\_FileWriteEvent\\_t](#) {
 [CFE\\_FS\\_FileWriteEvent\\_UNDEFINED](#), [CFE\\_FS\\_FileWriteEvent\\_COMPLETE](#), [CFE\\_FS\\_FileWriteEvent\\_CREATE\\_ERROR](#),
 [CFE\\_FS\\_FileWriteEvent\\_HEADER\\_WRITE\\_ERROR](#),
 [CFE\\_FS\\_FileWriteEvent\\_RECORD\\_WRITE\\_ERROR](#), [CFE\\_FS\\_FileWriteEvent\\_MAX](#) }

### 12.63.1 Detailed Description

Purpose: cFE File Services (FS) library API header file  
 Author: S.Walling/Microtel

### 12.63.2 Typedef Documentation

**12.63.2.1 [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
--------	------	--------------------------------

**Parameters**

in	<i>RecordNum</i>	Incrementing record number counter
out	<i>Buffer</i>	Pointer to buffer data block, should be set by implementation
out	<i>BufSize</i>	Pointer to buffer data size, should be set by implementation

**Returns**

End of file status

**Return values**

<i>true</i>	if at last data record, and output file should be closed
<i>false</i>	if not at last record, more data records to write

**Note**

The implementation of this function must always set the "Buffer" and "BufSize" outputs. If no data is available, they may be set to NULL and 0, respectively.

Definition at line 97 of file cfe\_fs\_api\_typedefs.h.

**12.63.2.2 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).

**12.63.2.3 CFE\_FS\_FileWriteOnEvent\_t** `typedef void(* CFE_FS_FileWriteOnEvent_t) (void *Meta, CFE_FS_FileWriteEvent_t Event, int32 Status, uint32 RecordNum, size_t BlockSize, size_t Position)`  
Event generator routine provided by requester  
Invoked from certain points in the file write process. Implementation may invoke [CFE\\_EVS\\_SendEvent\(\)](#) appropriately to inform of progress.

**Parameters**

in,out	<i>Meta</i>	Pointer to the metadata object
in	<i>Event</i>	Generalized type of event to report (not actual event ID)
in	<i>Status</i>	Generalized status code (may be from OSAL or CFE)
in	<i>RecordNum</i>	Record number counter at which event occurred
in	<i>BlockSize</i>	Size of record being processed when event occurred (if applicable)
in	<i>Position</i>	File position/size when event occurred

Definition at line 113 of file cfe\_fs\_api\_typedefs.h.

### 12.63.3 Enumeration Type Documentation

**12.63.3.1 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.

#### 12.63.3.2 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.

## 12.64 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\\_UpdateHeader \(CFE\\_MSG\\_Message\\_t \\*MsgPtr, CFE\\_MSG\\_SequenceCount\\_t SeqCnt\)](#)

*Set/compute all dynamically-updated headers on a message.*
- [CFE\\_Status\\_t CFE\\_MSG\\_GetSize \(const CFE\\_MSG\\_Message\\_t \\*MsgPtr, CFE\\_MSG\\_Size\\_t \\*Size\)](#)

*Gets the total size of a message.*

- `CFE_Status_t CFE_MSG_SetSize (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t Size)`  
*Sets the total size of a message.*
- `CFE_Status_t CFE_MSG_GetType (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t *Type)`  
*Gets the message type.*
- `CFE_Status_t CFE_MSG_SetType (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t Type)`  
*Sets the message type.*
- `CFE_Status_t CFE_MSG_GetHeaderVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t *Version)`  
*Gets the message header version.*
- `CFE_Status_t CFE_MSG_SetHeaderVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t Version)`  
*Sets the message header version.*
- `CFE_Status_t CFE_MSG_GetHasSecondaryHeader (const CFE_MSG_Message_t *MsgPtr, bool *HasSecondary)`  
*Gets the message secondary header boolean.*
- `CFE_Status_t CFE_MSG_SetHasSecondaryHeader (CFE_MSG_Message_t *MsgPtr, bool HasSecondary)`  
*Sets the message secondary header boolean.*
- `CFE_Status_t CFE_MSG_GetApld (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t *Apld)`  
*Gets the message application ID.*
- `CFE_Status_t CFE_MSG_SetApld (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t Apld)`  
*Sets the message application ID.*
- `CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)`  
*Gets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)`  
*Sets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)`  
*Gets the message sequence count.*
- `CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)`  
*Sets the message sequence count.*
- `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)`  
*Gets the next sequence count value (rolls over if appropriate)*
- `CFE_Status_t CFE_MSG_GetEDSVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t *Version)`  
*Gets the message EDS version.*
- `CFE_Status_t CFE_MSG_SetEDSVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t Version)`  
*Sets the message EDS version.*
- `CFE_Status_t CFE_MSG_GetEndian (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t *Endian)`  
*Gets the message endian.*
- `CFE_Status_t CFE_MSG_SetEndian (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t Endian)`  
*Sets the message endian.*
- `CFE_Status_t CFE_MSG_GetPlaybackFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t *PlayFlag)`  
*Gets the message playback flag.*
- `CFE_Status_t CFE_MSG_SetPlaybackFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t PlayFlag)`  
*Sets the message playback flag.*

- `CFE_Status_t CFE_MSG_SetSubsystem` (`const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t *Subsystem`)
 

*Sets the message subsystem.*
- `CFE_Status_t CFE_MSG_SetSubsystem` (`CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t Subsystem`)
 

*Gets the message subsystem.*
- `CFE_Status_t CFE_MSG_SetSystem` (`const CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t *System`)
 

*Gets the message system.*
- `CFE_Status_t CFE_MSG_SetSystem` (`CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t System`)
 

*Sets the message system.*
- `CFE_Status_t CFE_MSG_GenerateChecksum` (`CFE_MSG_Message_t *MsgPtr`)
 

*Calculates and sets the checksum of a message.*
- `CFE_Status_t CFE_MSG_ValidateChecksum` (`const CFE_MSG_Message_t *MsgPtr, bool *isValid`)
 

*Validates the checksum of a message.*
- `CFE_Status_t CFE_MSG_SetFcnCode` (`CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t FcnCode`)
 

*Sets the function code field in a message.*
- `CFE_Status_t CFE_MSG_GetFcnCode` (`const CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t *FcnCode`)
 

*Gets the function code field from a message.*
- `CFE_Status_t CFE_MSG_GetMsgTime` (`const CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t *Time`)
 

*Gets the time field from a message.*
- `CFE_Status_t CFE_MSG_SetMsgTime` (`CFE_MSG_Message_t *MsgPtr, CFE_TIME_SysTime_t NewTime`)
 

*Sets the time field in a message.*
- `CFE_Status_t CFE_MSG_GetMsgId` (`const CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t *MsgId`)
 

*Gets the message id from a message.*
- `CFE_Status_t CFE_MSG_SetMsgId` (`CFE_MSG_Message_t *MsgPtr, CFE_SB_MsgId_t MsgId`)
 

*Sets the message id bits in a message.*
- `CFE_Status_t CFE_MSG_GetTypeFromMsgId` (`CFE_SB_MsgId_t MsgId, CFE_MSG_Type_t *Type`)
 

*Gets message type using message ID.*
- `CFE_Status_t CFE_MSG_Verify` (`const CFE_MSG_Message_t *MsgPtr, bool *VerifyStatus`)
 

*Checks message headers against expected values.*

#### 12.64.1 Detailed Description

Message access APIs

## 12.65 cfe/modules/core\_api/fsw/inc/cfe\_msg\_api\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
```

#### Macros

- `#define CFE_MSG_BAD_ARGUMENT CFE_SB_BAD_ARGUMENT`

*Error - bad argument.*
- `#define CFE_MSG_NOT_IMPLEMENTED CFE_SB_NOT_IMPLEMENTED`

*Error - not implemented.*
- `#define CFE_MSG_WRONG_MSG_TYPE CFE_SB_WRONG_MSG_TYPE`

*Error - wrong type.*

## Typedefs

- `typedef size_t CFE_MSG_Size_t`  
*Message size, note CCSDS maximum is UINT16\_MAX+7.*
- `typedef uint32 CFE_MSG_Checksum_t`  
*Message checksum (Oversized to avoid redefine)*
- `typedef uint16 CFE_MSG_FcnCode_t`  
*Message function code.*
- `typedef uint16 CFE_MSG_HeaderVersion_t`  
*Message header version.*
- `typedef uint16 CFE_MSG_Apld_t`  
*Message application ID.*
- `typedef uint16 CFE_MSG_SequenceCount_t`  
*Message sequence count.*
- `typedef uint16 CFE_MSG_EDSVersion_t`  
*Message EDS version.*
- `typedef uint16 CFE_MSG_Subsystem_t`  
*Message subsystem.*
- `typedef uint16 CFE_MSG_System_t`  
*Message system.*
- `typedef enum CFE_MSG_Type CFE_MSG_Type_t`  
*Message type.*
- `typedef enum CFE_MSG_SegmentationFlag CFE_MSG_SegmentationFlag_t`  
*Segmentation flags.*
- `typedef enum CFE_MSG_Endian CFE_MSG_Endian_t`  
*Endian flag.*
- `typedef enum CFE_MSG_PlaybackFlag CFE_MSG_PlaybackFlag_t`  
*Playback flag.*
- `typedef union CFE_MSG_Message CFE_MSG_Message_t`  
*cFS generic base message*
- `typedef struct CFE_MSG_CommandHeader CFE_MSG_CommandHeader_t`  
*cFS command header*
- `typedef struct CFE_MSG_TelemetryHeader CFE_MSG_TelemetryHeader_t`  
*cFS telemetry header*

## Enumerations

- `enum CFE_MSG_Type { CFE_MSG_Type_Invalid, CFE_MSG_Type_Cmd, CFE_MSG_Type_Tlm }`  
*Message type.*
- `enum CFE_MSG_SegmentationFlag { CFE_MSG_SegFlag_Invalid, CFE_MSG_SegFlag_Continue, CFE_MSG_SegFlag_First, CFE_MSG_SegFlag_Last, CFE_MSG_SegFlag_Unsegmented }`  
*Segmentation flags.*
- `enum CFE_MSG_Endian { CFE_MSG_Endian_Invalid, CFE_MSG_Endian_Big, CFE_MSG_Endian_Little }`  
*Endian flag.*
- `enum CFE_MSG_PlaybackFlag { CFE_MSG_PlayFlag_Invalid, CFE_MSG_PlayFlag_Original, CFE_MSG_PlayFlag_Playback }`  
*Playback flag.*

### 12.65.1 Detailed Description

Typedefs for Message API

- Separate from API so these can be adjusted for custom implementations

### 12.65.2 Macro Definition Documentation

#### 12.65.2.1 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.

#### 12.65.2.2 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.

#### 12.65.2.3 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.

### 12.65.3 Typedef Documentation

#### 12.65.3.1 CFE\_MSG\_ApId\_t `typedef uint16 CFE_MSG_ApId_t`

Message application ID.

Definition at line 50 of file cfe\_msg\_api\_typedefs.h.

#### 12.65.3.2 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.

#### 12.65.3.3 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.

#### 12.65.3.4 CFE\_MSG\_EDSVersion\_t `typedef uint16 CFE_MSG_EDSVersion_t`

Message EDS version.

Definition at line 52 of file cfe\_msg\_api\_typedefs.h.

#### 12.65.3.5 CFE\_MSG\_Endian\_t `typedef enum CFE_MSG_Endian CFE_MSG_Endian_t`

Endian flag.

**12.65.3.6 CFE\_MSG\_FcnCode\_t** `typedef uint16 CFE_MSG_FcnCode_t`

Message function code.

Definition at line 48 of file cfe\_msg\_api\_typedefs.h.

**12.65.3.7 CFE\_MSG\_HeaderVersion\_t** `typedef uint16 CFE_MSG_HeaderVersion_t`

Message header version.

Definition at line 49 of file cfe\_msg\_api\_typedefs.h.

**12.65.3.8 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.

**12.65.3.9 CFE\_MSG\_PlaybackFlag\_t** `typedef enum CFE_MSG_PlaybackFlag CFE_MSG_PlaybackFlag_t`

Playback flag.

**12.65.3.10 CFE\_MSG\_SegmentationFlag\_t** `typedef enum CFE_MSG_SegmentationFlag CFE_MSG_SegmentationFlag_t`

Segmentation flags.

**12.65.3.11 CFE\_MSG\_SequenceCount\_t** `typedef uint16 CFE_MSG_SequenceCount_t`

Message sequence count.

Definition at line 51 of file cfe\_msg\_api\_typedefs.h.

**12.65.3.12 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.

**12.65.3.13 CFE\_MSG\_Subsystem\_t** `typedef uint16 CFE_MSG_Subsystem_t`

Message subsystem.

Definition at line 53 of file cfe\_msg\_api\_typedefs.h.

**12.65.3.14 CFE\_MSG\_System\_t** `typedef uint16 CFE_MSG_System_t`

Message system.

Definition at line 54 of file cfe\_msg\_api\_typedefs.h.

**12.65.3.15 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.

**12.65.3.16 CFE\_MSG\_Type\_t** `typedef enum CFE_MSG_Type CFE_MSG_Type_t`

Message type.

## 12.65.4 Enumeration Type Documentation

### 12.65.4.1 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.

### 12.65.4.2 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.

### 12.65.4.3 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.

### 12.65.4.4 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.

## 12.66 cfe/modules/core\_api/fsw/inc/cfe\_resourceid.h File Reference

```
#include "cfe_resourceid_api_typedefs.h"
```

### Functions

- `uint32 CFE_ResourceId_GetBase (CFE_ResourceId_t ResourceId)`  
*Get the Base value (type/category) from a resource ID value.*
- `uint32 CFE_ResourceId_GetSerial (CFE_ResourceId_t ResourceId)`  
*Get the Serial Number (sequential ID) from a resource ID value.*
- `CFE_ResourceId_t CFE_ResourceId_FindNext (CFE_ResourceId_t StartId, uint32 TableSize, bool(*Check←Func)(CFE_ResourceId_t))`  
*Locate the next resource ID which does not map to an in-use table entry.*
- `int32 CFE_ResourceId_ToIndex (CFE_ResourceId_t Id, uint32 BaseValue, uint32 TableSize, uint32 *Idx)`  
*Internal routine to aid in converting an ES resource ID to an array index.*

### Resource ID test/conversion macros and inline functions

- `#define CFE_RESOURCEID_TO ULONG(id) CFE_ResourceId_ToInteger(CFE_RESOURCEID_UNWRAP(id))`  
*Convert a derived (app-specific) ID directly into an "unsigned long".*
- `#define CFE_RESOURCEID_TEST_DEFINED(id) CFE_ResourceId_IsDefined(CFE_RESOURCEID_UNWRAP(id))`  
*Determine if a derived (app-specific) ID is defined or not.*
- `#define CFE_RESOURCEID_TEST_EQUAL(id1, id2) CFE_ResourceId_Equal(CFE_RESOURCEID_UNWRAP(id1), CFE_RESOURCEID_UNWRAP(id2))`  
*Determine if two derived (app-specific) IDs are equal.*
- `static unsigned long CFE_ResourceId_ToInteger (CFE_ResourceId_t id)`  
*Convert a resource ID to an integer.*
- `static CFE_ResourceId_t CFE_ResourceId_FromInteger (unsigned long Value)`  
*Convert an integer to a resource ID.*
- `static bool CFE_ResourceId_Equal (CFE_ResourceId_t id1, CFE_ResourceId_t id2)`  
*Compare two Resource ID values for equality.*
- `static bool CFE_ResourceId_IsDefined (CFE_ResourceId_t id)`  
*Check if a resource ID value is defined.*

### 12.66.1 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)

## 12.66.2 Macro Definition Documentation

**12.66.2.1 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.

**12.66.2.2 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.

**12.66.2.3 CFE\_RESOURCEID\_TO ULONG** #define CFE\_RESOURCEID\_TO ULONG (  
 `id ) CFE_ResourceId_ToInteger(CFE_RESOURCEID_UNWRAP(id))`

Convert a derived (app-specific) ID directly into an "unsigned long".

This generic routine is implemented as a macro so it is agnostic to the actual argument type, and it will evaluate correctly so long as the argument type is based on the CFE\_RESOURCEID\_BASE\_TYPE.

There is no inverse of this macro, as it depends on the actual derived type desired. Applications needing to recreate an ID from an integer should use `CFE_Resourceld_FromInteger()` combined with a cast/conversion to the correct/intended derived type, as needed.

### Note

This evaluates as an "unsigned long" such that it can be used in printf()-style functions with the "%lx" modifier without extra casting, as this is the most typical use-case for representing an ID as an integer.

Definition at line 62 of file cfe\_resourceid.h.

## 12.66.3 Function Documentation

**12.66.3.1 CFE\_Resourceld\_Equal()** static bool CFE\_ResourceId\_Equal (  
 `CFE_ResourceId_t id1,`  
 `CFE_ResourceId_t id2 ) [inline], [static]`

Compare two Resource ID values for equality.

### Parameters

in	<code>id1</code>	Resource ID to check
in	<code>id2</code>	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().

**12.66.3.2 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	<i>StartId</i>	the last issued ID for the resource category (app, lib, etc).
in	<i>TableSize</i>	the maximum size of the target table
in	<i>CheckFunc</i>	a function to check if the given ID is available

**Returns**

Next ID value which does not map to a valid entry

**Return values**

<a href="#">CFE_RESOURCEID_UNDEFINED</a>	if no open slots or bad arguments.
--	------------------------------------

**12.66.3.3 CFE\_ResourceId\_FromInteger()** `static CFE_ResourceId_t CFE_ResourceId_FromInteger (`  
 `unsigned long Value ) [inline], [static]`

Convert an integer to a resource ID.

This is the inverse of [CFE\\_ResourceId\\_ToInteger\(\)](#), and reconstitutes the original CFE\_ResourceId\_t value from the integer representation.

This may be used, for instance, where an ID value is parsed from a text file or message using C library APIs such as scanf() or strtoul().

**See also**

[CFE\\_ResourceId\\_ToInteger\(\)](#)

**Parameters**

in	<i>Value</i>	Integer value to convert
----	--------------	--------------------------

**Returns**

ID value corresponding to integer

Definition at line 121 of file cfe\_resourceid.h.

**12.66.3.4 CFE\_ResourceId\_GetBase()** `uint32 CFE_ResourceId_GetBase ( CFE_ResourceId_t ResourceId )`

Get the Base value (type/category) from a resource ID value.

This masks out the ID serial number to obtain the base value, which is different for each resource type.

**Note**

The value is NOT shifted or otherwise adjusted.

**Parameters**

in	<i>Resource</i> <i>Id</i>	the resource ID to decode
----	------------------------------	---------------------------

**Returns**

The base value associated with that ID

**12.66.3.5 CFE\_ResourceId\_GetSerial()** `uint32 CFE_ResourceId_GetSerial ( CFE_ResourceId_t ResourceId )`

Get the Serial Number (sequential ID) from a resource ID value.

This masks out the ID base value to obtain the serial number, which is different for each entity created.

**Parameters**

in	<i>Resource</i> <i>Id</i>	the resource ID to decode
----	------------------------------	---------------------------

**Returns**

The serial number associated with that ID

**12.66.3.6 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	<i>id</i>	Resource ID to check
----	-----------	----------------------

**Returns**

True if the ID may refer to a defined entity, false if invalid/undefined.

Definition at line 151 of file cfe\_resourceid.h.

References CFE\_Resourceld\_Equal(), and CFE\_RESOURCEID\_UNDEFINED.

Here is the call graph for this function:

**12.66.3.7 CFE\_Resourceld\_ToIndex()** `int32 CFE_ResourceId_ToIndex (`

```
    CFE_ResourceId_t Id,  
    uint32 BaseValue,  
    uint32 TableSize,  
    uint32 * Idx )
```

Internal routine to aid in converting an ES resource ID to an array index.

**Parameters**

in	<i>Id</i>	The resource ID
in	<i>BaseValue</i>	The respective ID base value corresponding to the ID type
in	<i>TableSize</i>	The actual size of the internal table (MAX index value + 1)
out	<i>Idx</i>	The output index

**Returns**

Execution status, see [cFE Return Code Defines](#)

**Return values**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.
<code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code>	Resource ID is not valid.

**12.66.3.8 CFE\_Resourceld\_Tointeger()** `static unsigned long CFE_ResourceId_ToInteger (`  
 `CFE_ResourceId_t id ) [inline], [static]`

Convert a resource ID to an integer.

This is primarily intended for logging purposes, such as writing to debug console, event messages, or log files, using printf-like APIs.

For compatibility with C library APIs, this returns an "unsigned long" type and should be used with the "%lx" format specifier in a printf format string.

**Note**

No assumptions should be made about the actual integer value, such as its base/range. It may be printed, but should not be modified or tested/compared using other arithmetic ops, and should never be used as the index to an array or table. See the related function [CFE\\_ResourceId\\_ToIndex\(\)](#) for cases where a zero-based array/table index is needed.

**See also**

[CFE\\_ResourceId\\_FromInteger\(\)](#)

**Parameters**

in	<i>id</i>	Resource ID to convert
----	-----------	------------------------

**Returns**

Integer value corresponding to ID

Definition at line 102 of file cfe\_resourceid.h.

**12.67 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.*

**12.67.1 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)

**12.67.2 Macro Definition Documentation**

**12.67.2.1 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.

**12.67.2.2 CFE\_RESOURCEID\_UNDEFINED** #define CFE\_RESOURCEID\_UNDEFINED ((CFE\_ResourceId\_t)CFE\_RESOURCEID\_WRAP(0))

A resource ID value that represents an undefined/unused resource.

This constant may be used to initialize local variables of the CFE\_Resourceld\_t type to a safe value that will not alias a valid ID.

By design, this value is also the result of zeroing a CFE\_Resourceld\_t type via standard functions like memset(), such that objects initialized using this method will also be set to safe values.

Definition at line 65 of file cfe\_resourceid\_api\_typedefs.h.

## 12.68 cfe/modules/core\_api/fsw/inc/cfe\_sb.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_sb_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
```

### Macros

- #define **CFE\_BIT**(x) (1 << (x))  
*Places a one at bit positions 0 - 31.*
- #define **CFE\_SET**(i, x) ((i) |= **CFE\_BIT**(x))  
*Sets bit x of i.*
- #define **CFE\_CLR**(i, x) ((i) &= ~**CFE\_BIT**(x))  
*Clears bit x of i.*
- #define **CFE\_TST**(i, x) (((i)&**CFE\_BIT**(x)) != 0)  
*true(non zero) if bit x of i is set*

### Functions

- **CFE\_Status\_t CFE\_SB\_CreatePipe** (**CFE\_SB\_Pipeld\_t** \*PipeldPtr, **uint16** Depth, const char \*PipeName)  
*Creates a new software bus pipe.*
- **CFE\_Status\_t CFE\_SB\_DeletePipe** (**CFE\_SB\_Pipeld\_t** Pipeld)  
*Delete a software bus pipe.*
- **CFE\_Status\_t CFE\_SB\_Pipeld\_ToIndex** (**CFE\_SB\_Pipeld\_t** PipeID, **uint32** \*Idx)  
*Obtain an index value correlating to an SB Pipe ID.*
- **CFE\_Status\_t CFE\_SB\_SetPipeOpts** (**CFE\_SB\_Pipeld\_t** Pipeld, **uint8** Opts)  
*Set options on a pipe.*
- **CFE\_Status\_t CFE\_SB\_GetPipeOpts** (**CFE\_SB\_Pipeld\_t** Pipeld, **uint8** \*OptsPtr)  
*Get options on a pipe.*
- **CFE\_Status\_t CFE\_SB\_GetPipeName** (char \*PipeNameBuf, **size\_t** PipeNameSize, **CFE\_SB\_Pipeld\_t** Pipeld)  
*Get the pipe name for a given id.*
- **CFE\_Status\_t CFE\_SB\_GetPipeldByName** (**CFE\_SB\_Pipeld\_t** \*PipeldPtr, const char \*PipeName)

- **`CFE_Status_t CFE_SB_SubscribeEx (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld, CFE_SB_Qos_t Quality, uint16 MsgLim)`**

*Get pipe id by pipe name.*
- **`CFE_Status_t CFE_SB_Subscribe (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`**

*Subscribe to a message on the software bus.*
- **`CFE_Status_t CFE_SB_SubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld, uint16 MsgLim)`**

*Subscribe to a message while keeping the request local to a cpu.*
- **`CFE_Status_t CFE_SB_Unsubscribe (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`**

*Remove a subscription to a message on the software bus.*
- **`CFE_Status_t CFE_SB_UnsubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`**

*Remove a subscription to a message on the software bus on the current CPU.*
- **`CFE_Status_t CFE_SB_TransmitMsg (const CFE_MSG_Message_t *MsgPtr, bool UpdateHeader)`**

*Transmit a message.*
- **`CFE_Status_t CFE_SB_ReceiveBuffer (CFE_SB_Buffer_t **BufPtr, CFE_SB_Pipeld_t Pipeld, int32 TimeOut)`**

*Receive a message from a software bus pipe.*
- **`CFE_SB_Buffer_t * CFE_SB_AllocateMessageBuffer (size_t MsgSize)`**

*Get a buffer pointer to use for "zero copy" SB sends.*
- **`CFE_Status_t CFE_SB_ReleaseMessageBuffer (CFE_SB_Buffer_t *BufPtr)`**

*Release an unused "zero copy" buffer pointer.*
- **`CFE_Status_t CFE_SB_TransmitBuffer (CFE_SB_Buffer_t *BufPtr, bool UpdateHeader)`**

*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`.*

### 12.68.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.  
 Author: R.McGraw/SSI

## 12.68.2 Macro Definition Documentation

**12.68.2.1 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.

**12.68.2.2 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.

**12.68.2.3 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.

**12.68.2.4 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.

## 12.69 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`

- **#define CFE\_SB\_MSGID\_WRAP\_VALUE(val)**

*Subtype specified used in `CFE_SB_SingleSubscriptionTlm_t` by SBN App.*
- **#define CFE\_SB\_MSGID\_C(val) ((CFE\_SB\_MsgId\_t)CFE\_SB\_MSGID\_WRAP\_VALUE(val))**

*Translation macro to convert from MsgId integer values to opaque/abstract API values.*
- **#define CFE\_SB\_MSGID\_UNWRAP\_VALUE(mid) ((mid).Value)**

*Translation macro to convert to MsgId integer values from a literal.*
- **#define CFE\_SB\_MSGID\_RESERVED CFE\_SB\_MSGID\_WRAP\_VALUE(0)**

*Reserved value for `CFE_SB_MsgId_t` that will not match any valid MsgId.*
- **#define CFE\_SB\_INVALID\_MSG\_ID CFE\_SB\_MSGID\_C(0)**

*A literal of the `CFE_SB_MsgId_t` type representing an invalid ID.*
- **#define CFE\_SB\_PIPEID\_C(val) ((CFE\_SB\_Pipeld\_t)CFE\_RESOURCEID\_WRAP(val))**

*Cast/Convert a generic `CFE_Resourceld_t` to a `CFE_SB_Pipeld_t`.*
- **#define CFE\_SB\_INVALID\_PIPE CFE\_SB\_PIPEID\_C(CFE\_RESOURCEID\_UNDEFINED)**

*A `CFE_SB_Pipeld_t` value which is always invalid.*
- **#define CFE\_SB PIPEOPTS IGNOREMINE 0x00000001**

*Messages sent by the app that owns this pipe will not be sent to this pipe.*
- **#define CFE\_SB\_DEFAULT\_QOS ((CFE\_SB\_Qos\_t) {0})**

*Default Qos macro.*

## Typedefs

- **typedef union CFE\_SB\_Msg CFE\_SB\_Buffer\_t**

*Software Bus generic message.*

### 12.69.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.  
 Author: R.McGraw/SSI

### 12.69.2 Macro Definition Documentation

#### 12.69.2.1 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`.

#### 12.69.2.2 CFE\_SB\_INVALID\_MSG\_ID #define CFE\_SB\_INVALID\_MSG\_ID CFE\_SB\_MSGID\_C(0)

A literal of the `CFE_SB_MsgId_t` type representing an invalid ID.

This value should be used for runtime initialization of `CFE_SB_MsgId_t` values.

#### Note

This may be a compound literal in a future revision. Per C99, compound literals are lvalues, not rvalues, so this value should not be used in static/compile-time data initialization. For static data initialization purposes (rvalue), `CFE_SB_MSGID_RESERVED` should be used instead. However, in the current implementation, they are equivalent.

Definition at line 113 of file `cfe_sb_api_typedefs.h`.

**12.69.2.3 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.

**12.69.2.4 CFE\_SB\_MSGID\_C** #define CFE\_SB\_MSGID\_C(  
    val) ((CFE\_SB\_MsgId\_t)CFE\_SB\_MSGID\_WRAP\_VALUE(val))

Translation macro to convert to MsgId integer values from a literal.

This ensures that the literal is interpreted as the CFE\_SB\_MsgId\_t type, rather than the default type associated with that literal (e.g. int/unsigned int).

Note

Due to constraints in C99 this style of initializer can only be used at runtime, not for static/compile-time initializers.

See also

[CFE\\_SB\\_ValueToMsgId\(\)](#)

Definition at line 80 of file cfe\_sb\_api\_typedefs.h.

**12.69.2.5 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.

**12.69.2.6 CFE\_SB\_MSGID\_UNWRAP\_VALUE** #define CFE\_SB\_MSGID\_UNWRAP\_VALUE(  
    mid) ((mid).Value)

Translation macro to convert to MsgId integer values from opaque/abstract API values.

This conversion exists in macro form to allow compile-time evaluation for constants, and should not be used directly in application code.

For applications, use the [CFE\\_SB\\_MsgIdToValue\(\)](#) inline function instead.

See also

[CFE\\_SB\\_MsgIdToValue\(\)](#)

Definition at line 92 of file cfe\_sb\_api\_typedefs.h.

**12.69.2.7 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.

**12.69.2.8 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.

**12.69.2.9 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.

**12.69.2.10 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.

**12.69.2.11 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.

**12.69.2.12 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.

### 12.69.3 Typedef Documentation

**12.69.3.1 CFE\_SB\_Buffer\_t** typedef union [CFE\\_SB\\_Msg](#) CFE\_SB\_Buffer\_t  
 Software Bus generic message.

## 12.70 cfe/modules/core\_api/fsw/inc/cfe\_tbl.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_tbl_api_typedefs.h"
#include "cfe_sb_api_typedefs.h"
```

### Functions

- [CFE\\_Status\\_t CFE\\_TBL\\_Register](#) ([CFE\\_TBL\\_Handle\\_t](#) \*TblHandlePtr, const char \*Name, size\_t Size, uint16 TblOptionFlags, [CFE\\_TBL\\_CallbackFuncPtr\\_t](#) TblValidationFuncPtr)
   
*Register a table with cFE to obtain Table Management Services.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Share](#) ([CFE\\_TBL\\_Handle\\_t](#) \*TblHandlePtr, const char \*TblName)
   
*Obtain handle of table registered by another application.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Unregister](#) ([CFE\\_TBL\\_Handle\\_t](#) TblHandle)
   
*Unregister a table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Load](#) ([CFE\\_TBL\\_Handle\\_t](#) TblHandle, [CFE\\_TBL\\_SrcEnum\\_t](#) SrcType, const void \*SrcDataPtr)
   
*Load a specified table with data from specified source.*

- [CFE\\_Status\\_t CFE\\_TBL\\_Update \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Update contents of a specified table, if an update is pending.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Validate \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Perform steps to validate the contents of a table image.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Manage \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Perform standard operations to maintain a table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_DumpToBuffer \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Copies the contents of a Dump Only Table to a shared buffer.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Modified \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Notify cFE Table Services that table contents have been modified by the Application.*
- [CFE\\_Status\\_t CFE\\_TBL\\_GetAddress \(void \\*\\*TblPtr, CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Obtain the current address of the contents of the specified table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_ReleaseAddress \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Release previously obtained pointer to the contents of the specified table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_GetAddresses \(void \\*\\*TblPtrs\[\], uint16 NumTables, const CFE\\_TBL\\_Handle\\_t TblHandles\[\]\)](#)  
*Obtain the current addresses of an array of specified tables.*
- [CFE\\_Status\\_t CFE\\_TBL\\_ReleaseAddresses \(uint16 NumTables, const CFE\\_TBL\\_Handle\\_t TblHandles\[\]\)](#)  
*Release the addresses of an array of specified tables.*
- [CFE\\_Status\\_t CFE\\_TBL\\_GetStatus \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Obtain current status of pending actions for a table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_GetInfo \(CFE\\_TBL\\_Info\\_t \\*TblInfoPtr, const char \\*TblName\)](#)  
*Obtain characteristics/information of/about a specified table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_NotifyByMessage \(CFE\\_TBL\\_Handle\\_t TblHandle, CFE\\_SB\\_MsgId\\_t MsgId, CFE\\_MSG\\_FcnCode\\_t CommandCode, uint32 Parameter\)](#)  
*Instruct cFE Table Services to notify Application via message when table requires management.*

### 12.70.1 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:

## 12.71 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.*

### 12.71.1 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:

### 12.71.2 Macro Definition Documentation

#### 12.71.2.1 **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.

#### 12.71.2.2 **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.

### 12.71.3 Typedef Documentation

#### 12.71.3.1 **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.

#### 12.71.3.2 **CFE\_TBL\_Handle\_t** typedef int16 CFE\_TBL\_Handle\_t

Table Handle primitive.

Definition at line 87 of file cfe\_tbl\_api\_typedefs.h.

#### 12.71.3.3 **CFE\_TBL\_Info\_t** typedef struct CFE\_TBL\_Info CFE\_TBL\_Info\_t

Table Info.

#### 12.71.3.4 **CFE\_TBL\_SrcEnum\_t** typedef enum CFE\_TBL\_SrcEnum CFE\_TBL\_SrcEnum\_t

Table Source.

### 12.71.4 Enumeration Type Documentation

#### 12.71.4.1 **CFE\_TBL\_SrcEnum** enum CFE\_TBL\_SrcEnum

Table Source.

**Enumerator**

CFE_TBL_SRC_FILE	File source When this option is selected, the <code>SrcDataPtr</code> will be interpreted as a pointer to a null terminated character string. The string should specify the full path and filename of the file containing the initial data contents of the table.
CFE_TBL_SRC_ADDRESS	Address source When this option is selected, the <code>SrcDataPtr</code> will be interpreted as a pointer to a memory location that is the beginning of the initialization data for loading the table OR, in the case of a "user defined" dump only table, the address of the active table itself. The block of memory is assumed to be of the same size specified in the <a href="#">CFE_TBL_Register</a> function Size parameter.

Definition at line 90 of file `cfe_tbl_api_typedefs.h`.

## 12.72 cfe/modules/core\_api/fsw/inc/cfe\_tbl\_filedef.h File Reference

```
#include "cfe_mission_cfg.h"
#include "common_types.h"
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_fs_extern_typedefs.h"
```

### Data Structures

- struct [CFE\\_TBL\\_FileDef](#)

*Table File summary object.*

### Macros

- #define [CFE\\_TBL\\_FILEDEF](#)(ObjName, TblName, Desc, Filename) [CFE\\_TBL\\_FileDef\\_t](#) `CFE_TBL_FileDef = {#ObjName "\0", #TblName "\0", #Desc "\0", #Filename "\0", sizeof(ObjName)}`;

*Macro to assist in with table definition object declaration.*

### Typedefs

- typedef struct [CFE\\_TBL\\_FileDef](#) [CFE\\_TBL\\_FileDef\\_t](#)

*Table File summary object.*

### 12.72.1 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:

### 12.72.2 Macro Definition Documentation

```
12.72.2.1 CFE_TBL_FILEDEF #define CFE_TBL_FILEDEF (
    ObjName,
    TblName,
    Desc,
    Filename ) CFE_TBL_FileDef_t CFE_TBL_FileDef = { #ObjName "\0", #TblName "\0", #Desc
"\0", #Filename "\0", sizeof(ObjName) };
```

Macro to assist in with table definition object declaration.

See notes in the [CFE\\_TBL\\_FileDef\\_t](#) structure type about naming conventions and recommended practices for the various fields.

The CFE\_TBL\_FILEDEF macro can be used to simplify the declaration of a table image when using the elf2cfetbl utility. Note that the macro adds a NULL at the end to ensure that it is null-terminated. (C allows a struct to be statically initialized with a string exactly the length of the array, which loses the null terminator.) This means the actual length limit of the fields are the above LEN - 1.

An example of the source code and how this macro would be used is as follows:

```
#include "cfe_tbl_filedef.h"
typedef struct MyTblStruct
{
    int      Int1;
    int      Int2;
    int      Int3;
    char    Char1;
} MyTblStruct_t;
MyTblStruct_t MyTblStruct = { 0x01020304, 0x05060708, 0x090A0B0C, 0x0D };
CFE_TBL_FILEDEF(MyTblStruct, MyApp.TableName, Table Utility Test Table, MyTblDefault.bin )
```

Definition at line 149 of file cfe\_tbl\_filedef.h.

### 12.72.3 Typedef Documentation

#### 12.72.3.1 CFE\_TBL\_FileDef\_t [typedef struct CFE\\_TBL\\_FileDef CFE\\_TBL\\_FileDef\\_t](#)

Table File summary object.

The definition of the file definition metadata that can be used by external tools (e.g. elf2cfetbl) to generate CFE table data files.

## 12.73 [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.*

### 12.73.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microlab

Notes:

## 12.73.2 Macro Definition Documentation

### **12.73.2.1 CFE\_TIME\_Copy #define CFE\_TIME\_Copy(**

$m,$   
 $t$ )

## **Value:**

```
{\n    (m) ->Seconds      = (t) ->Seconds;\n    (m) ->Subseconds = (t) ->Subseconds;\n}
```

## 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.

## **12.74 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.*

#### **12.74.1 Detailed Description**

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

#### **Notes:**

## 12.74.2 Macro Definition Documentation

**12.74.2.1 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)

Required size of buffer to be passed into `CFTimeGet`.

### 12.74.3 Typedef Documentation

#### 12.74.3.1 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.

#### 12.74.3.2 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`.

### 12.74.4 Enumeration Type Documentation

#### 12.74.4.1 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

<code>CFE_TIME_A_LT_B</code>	The first specified time is considered to be before the second specified time.
<code>CFE_TIME_EQUAL</code>	The two specified times are considered to be equal.
<code>CFE_TIME_A_GT_B</code>	The first specified time is considered to be after the second specified time.

Definition at line 60 of file `cfe_time_api_typedefs.h`.

## 12.75 cfe/modules/core\_api/fsw/inc/cfe\_version.h File Reference

### Macros

- `#define CFE_BUILD_NUMBER 384`

*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.*

### 12.75.1 Detailed Description

Provide version identifiers for the cFE core. See [Version Numbers](#) for further details.

### 12.75.2 Macro Definition Documentation

**12.75.2.1 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`.

**12.75.2.2 CFE\_BUILD\_NUMBER** `#define CFE_BUILD_NUMBER 384`  
Development: Number of development git commits since CFE\_BUILD\_BASELINE.  
Definition at line 29 of file `cfe_version.h`.

**12.75.2.3 CFE\_MAJOR\_VERSION** `#define CFE_MAJOR_VERSION 6`  
Major version number.  
Definition at line 33 of file `cfe_version.h`.

**12.75.2.4 CFE\_MINOR\_VERSION** `#define CFE_MINOR_VERSION 7`  
Minor version number.  
Definition at line 34 of file `cfe_version.h`.

**12.75.2.5 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.

**12.75.2.6 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.

**12.75.2.7 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.

**12.75.2.8 CFE\_STR** #define CFE\_STR(  
x ) CFE\_STR\_HELPER(x)

Expand macro before conversion.

Definition at line 47 of file cfe\_version.h.

**12.75.2.9 CFE\_STR\_HELPER** #define CFE\_STR\_HELPER(  
x ) #x

Convert argument to string.

Definition at line 46 of file cfe\_version.h.

**12.75.2.10 CFE\_VERSION\_STRING** #define CFE\_VERSION\_STRING " cFE DEVELOPMENT BUILD " CFE\_SRC\_V←  
ERSION " (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 re-releases.

Definition at line 63 of file cfe\_version.h.

## 12.76 cfe/modules/es/config/default\_cfe\_es\_extern\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_resourceid_typedef.h"
#include "cfe_mission_cfg.h"
```

### Data Structures

- struct [CFE\\_ES\\_AppInfo](#)  
*Application Information.*
- struct [CFE\\_ES\\_TaskInfo](#)  
*Task Information.*
- struct [CFE\\_ES\\_CDSRegDumpRec](#)  
*CDS Register Dump Record.*

- struct [CFE\\_ES\\_BlockStats](#)  
*Block statistics.*
- struct [CFE\\_ES\\_MemPoolStats](#)  
*Memory Pool Statistics.*

## Macros

- #define [CFE\\_ES\\_MEMOFFSET\\_C](#)(x) (([CFE\\_ES\\_MemOffset\\_t](#))(x))  
*Memory Offset initializer wrapper.*
- #define [CFE\\_ES\\_MEMOFFSET\\_TO\\_SIZE\\_T](#)(x) ((size\_t)(x))  
*Memory Offset to integer value (size\_t) wrapper.*
- #define [CFE\\_ES\\_MEMADDRESS\\_C](#)(x) (([CFE\\_ES\\_MemAddress\\_t](#))((cpuaddr)(x)&0xFFFFFFFF))  
*Memory Address initializer wrapper.*
- #define [CFE\\_ES\\_MEMADDRESS\\_TO\\_PTR](#)(x) ((void \*)([cpuaddr](#))(x))  
*Memory Address to pointer wrapper.*

## Typedefs

- typedef uint8 [CFE\\_ES\\_LogMode\\_Enum\\_t](#)  
*Identifies handling of log messages after storage is filled.*
- typedef uint8 [CFE\\_ES\\_ExceptionAction\\_Enum\\_t](#)  
*Identifies action to take if exception occurs.*
- typedef uint8 [CFE\\_ES\\_AppType\\_Enum\\_t](#)  
*Identifies type of CFE application.*
- typedef uint32 [CFE\\_ES\\_RunStatus\\_Enum\\_t](#)  
*Run Status and Exit Status identifiers.*
- typedef uint32 [CFE\\_ES\\_SystemState\\_Enum\\_t](#)  
*The overall cFE System State.*
- typedef uint8 [CFE\\_ES\\_LogEntryType\\_Enum\\_t](#)  
*Type of entry in the Error and Reset (ER) Log.*
- typedef uint32 [CFE\\_ES\\_AppState\\_Enum\\_t](#)  
*Application Run State.*
- typedef [CFE\\_RESOURCEID\\_BASE\\_TYPE](#) [CFE\\_ES\\_AppId\\_t](#)  
*A type for Application IDs.*
- typedef [CFE\\_RESOURCEID\\_BASE\\_TYPE](#) [CFE\\_ES\\_TaskId\\_t](#)  
*A type for Task IDs.*
- typedef [CFE\\_RESOURCEID\\_BASE\\_TYPE](#) [CFE\\_ES\\_LibId\\_t](#)  
*A type for Library IDs.*
- typedef [CFE\\_RESOURCEID\\_BASE\\_TYPE](#) [CFE\\_ES\\_CounterId\\_t](#)  
*A type for Counter IDs.*
- typedef [CFE\\_RESOURCEID\\_BASE\\_TYPE](#) [CFE\\_ES\\_MemHandle\\_t](#)  
*Memory Handle type.*
- typedef [CFE\\_RESOURCEID\\_BASE\\_TYPE](#) [CFE\\_ES\\_CDSHandle\\_t](#)  
*CDS Handle type.*
- typedef uint16 [CFE\\_ES\\_TaskPriority\\_Atom\\_t](#)  
*Type used for task priority in CFE ES as including the commands/telemetry messages.*
- typedef uint32 [CFE\\_ES\\_MemOffset\\_t](#)  
*Type used for memory sizes and offsets in commands and telemetry.*

- **typedef uint32 CFE\_ES\_MemAddress\_t**  
*Type used for memory addresses in command and telemetry messages.*
- **typedef struct CFE\_ES\_AppInfo CFE\_ES\_AppInfo\_t**  
*Application Information.*
- **typedef struct CFE\_ES\_TaskInfo CFE\_ES\_TaskInfo\_t**  
*Task Information.*
- **typedef struct CFE\_ES\_CDSRegDumpRec CFE\_ES\_CDSRegDumpRec\_t**  
*CDS Register Dump Record.*
- **typedef struct CFE\_ES\_BlockStats CFE\_ES\_BlockStats\_t**  
*Block statistics.*
- **typedef struct CFE\_ES\_MemPoolStats CFE\_ES\_MemPoolStats\_t**  
*Memory Pool Statistics.*

## Enumerations

- **enum CFE\_ES\_LogMode { CFE\_ES\_LogMode\_OVERWRITE = 0, CFE\_ES\_LogMode\_DISCARD = 1 }**  
*Label definitions associated with CFE\_ES\_LogMode\_Enum\_t.*
- **enum CFE\_ES\_ExceptionAction { CFE\_ES\_ExceptionAction\_RESTART\_APP = 0, CFE\_ES\_ExceptionAction\_PROC\_RESTART = 1 }**  
*Label definitions associated with CFE\_ES\_ExceptionAction\_Enum\_t.*
- **enum CFE\_ES\_AppType { CFE\_ES\_AppType\_CORE = 1, CFE\_ES\_AppType\_EXTERNAL = 2, CFE\_ES\_AppType\_LIBRARY = 3 }**  
*Label definitions associated with CFE\_ES\_AppType\_Enum\_t.*
- **enum CFE\_ES\_RunStatus { CFE\_ES\_RunStatus\_UNDEFINED = 0, CFE\_ES\_RunStatus\_APP\_RUN = 1, CFE\_ES\_RunStatus\_APP\_EXIT = 2, CFE\_ES\_RunStatus\_APP\_ERROR = 3, CFE\_ES\_RunStatus\_SYS\_EXCEPTION = 4, CFE\_ES\_RunStatus\_SYS\_RESTART = 5, CFE\_ES\_RunStatus\_SYS\_RELOAD = 6, CFE\_ES\_RunStatus\_SYS\_DELETE = 7, CFE\_ES\_RunStatus\_CORE\_APP\_INIT\_ERROR = 8, CFE\_ES\_RunStatus\_CORE\_APP\_RUNTIME\_ERROR = 9, CFE\_ES\_RunStatus\_MAX }**  
*Label definitions associated with CFE\_ES\_RunStatus\_Enum\_t.*
- **enum CFE\_ES\_SystemState { CFE\_ES\_SystemState\_UNDEFINED = 0, CFE\_ES\_SystemState\_EARLY\_INIT = 1, CFE\_ES\_SystemState\_CORE\_STARTUP = 2, CFE\_ES\_SystemState\_CORE\_READY = 3, CFE\_ES\_SystemState\_APPS\_INIT = 4, CFE\_ES\_SystemState\_OPERATIONAL = 5, CFE\_ES\_SystemState\_SHUTDOWN = 6, CFE\_ES\_SystemState\_MAX }**  
*Label definitions associated with CFE\_ES\_SystemState\_Enum\_t.*
- **enum CFE\_ES\_LogEntryType { CFE\_ES\_LogEntryType\_CORE = 1, CFE\_ES\_LogEntryType\_APPLICATION = 2 }**  
*Label definitions associated with CFE\_ES\_LogEntryType\_Enum\_t.*
- **enum CFE\_ES\_AppState { CFE\_ES\_AppState\_UNDEFINED = 0, CFE\_ES\_AppState\_EARLY\_INIT = 1, CFE\_ES\_AppState\_LATE\_INIT = 2, CFE\_ES\_AppState\_RUNNING = 3, CFE\_ES\_AppState\_WAITING = 4, CFE\_ES\_AppState\_STOPPED = 5, CFE\_ES\_AppState\_MAX }**  
*Label definitions associated with CFE\_ES\_AppState\_Enum\_t.*

### 12.76.1 Detailed Description

Declarations and prototypes for cfe\_es\_extern\_typedefs module

## 12.76.2 Macro Definition Documentation

**12.76.2.1 CFE\_ES\_MEMADDRESS\_C** `#define CFE_ES_MEMADDRESS_C (`  
`x ) ((CFE_ES_MemAddress_t)((cpuaddr)(x)&0xFFFFFFFF))`

Memory Address initializer wrapper.

A converter macro to use when initializing a CFE\_ES\_MemAddress\_t from a pointer value of a different type.

Definition at line 417 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.2.2 CFE\_ES\_MEMADDRESS\_TO\_PTR** `#define CFE_ES_MEMADDRESS_TO_PTR (`  
`x ) ((void *) (cpuaddr)(x))`

Memory Address to pointer wrapper.

A converter macro to use when interpreting a CFE\_ES\_MemAddress\_t as a pointer value.

Definition at line 425 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.2.3 CFE\_ES\_MEMOFFSET\_C** `#define CFE_ES_MEMOFFSET_C (`  
`x ) ((CFE_ES_MemOffset_t)(x))`

Memory Offset initializer wrapper.

A converter macro to use when initializing a CFE\_ES\_MemOffset\_t from an integer value of a different type.

Definition at line 380 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.2.4 CFE\_ES\_MEMOFFSET\_TO\_SIZE\_T** `#define CFE_ES_MEMOFFSET_TO_SIZE_T (`  
`x ) ((size_t)(x))`

Memory Offset to integer value (size\_t) wrapper.

A converter macro to use when interpreting a CFE\_ES\_MemOffset\_t value as a "size\_t" type

Definition at line 388 of file default\_cfe\_es\_extern\_typedefs.h.

## 12.76.3 Typedef Documentation

**12.76.3.1 CFE\_ES\_AppId\_t** `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_AppId_t`

A type for Application IDs.

This is the type that is used for any API accepting or returning an App ID

Definition at line 312 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.2 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.

**12.76.3.3 CFE\_ES\_AppState\_Enum\_t** `typedef uint32 CFE_ES_AppState_Enum_t`

Application Run State.

The normal progression of APP states: UNDEFINED -> EARLY\_INIT -> LATE\_INIT -> RUNNING -> WAITING -> STOPPED

**Note**

These are defined in order so that relational comparisons e.g. if (STATEA < STATEB) are possible

**See also**

enum [CFE\\_ES\\_AppState](#)

Definition at line 305 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.4 CFE\_ES\_AppType\_Enum\_t** `typedef uint8 CFE_ES_AppType_Enum_t`

Identifies type of CFE application.

**See also**

enum [CFE\\_ES\\_AppType](#)

Definition at line 104 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.5 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.

**12.76.3.6 CFE\_ES\_CDSHandle\_t** `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_CDSHandle_t`

CDS Handle type.

Data type used to hold Handles of Critical Data Stores. See [CFE\\_ES\\_RegisterCDS](#)

Definition at line 348 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.7 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.

**12.76.3.8 CFE\_ES\_CounterId\_t** `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_CounterId_t`

A type for Counter IDs.

This is the type that is used for any API accepting or returning a Counter ID

Definition at line 333 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.9 CFE\_ES\_ExceptionAction\_Enum\_t** `typedef uint8 CFE_ES_ExceptionAction_Enum_t`

Identifies action to take if exception occurs.

**See also**

enum [CFE\\_ES\\_ExceptionAction](#)

Definition at line 76 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.10 CFE\_ES\_LibId\_t** `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_LibId_t`

A type for Library IDs.

This is the type that is used for any API accepting or returning a Lib ID

Definition at line 326 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.11 CFE\_ES\_LogEntryType\_Enum\_t** `typedef uint8 CFE_ES_LogEntryType_Enum_t`

Type of entry in the Error and Reset (ER) Log.

See also

enum [CFE\\_ES\\_LogEntryType](#)

Definition at line 252 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.12 CFE\_ES\_LogMode\_Enum\_t** `typedef uint8 CFE_ES_LogMode_Enum_t`

Identifies handling of log messages after storage is filled.

See also

enum [CFE\\_ES\\_LogMode](#)

Definition at line 53 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.13 CFE\_ES\_MemAddress\_t** `typedef uint32 CFE_ES_MemAddress_t`

Type used for memory addresses in command and telemetry messages.

For backward compatibility with existing CFE code this should be uint32, but if running on a 64-bit platform, addresses in telemetry will be truncated to 32 bits and therefore will not be valid.

On 64-bit platforms this can be a 64-bit address which will allow the full memory address in commands and telemetry, but this will break compatibility with existing control systems, and may also change the alignment/padding of messages. In either case this must be an unsigned type.

FSW code should access this value via the macros provided, which converts to the native "cpuaddr" type provided by OSAL. This macro provides independence between the message representation and local representation of a memory address.

Definition at line 409 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.14 CFE\_ES\_MemHandle\_t** `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_MemHandle_t`

Memory Handle type.

Data type used to hold Handles of Memory Pools created via CFE\_ES\_PoolCreate and CFE\_ES\_PoolCreateNoSem

Definition at line 341 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.15 CFE\_ES\_MemOffset\_t** `typedef uint32 CFE_ES_MemOffset_t`

Type used for memory sizes and offsets in commands and telemetry.

For backward compatibility with existing CFE code this should be uint32, but all telemetry information will be limited to 4GB in size as a result.

On 64-bit platforms this can be a 64-bit value which will allow larger memory objects, but this will break compatibility with existing control systems, and may also change the alignment/padding of messages.

In either case this must be an unsigned type.

Definition at line 372 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.16 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](#)

**12.76.3.17 CFE\_ES\_RunStatus\_Enum\_t** `typedef uint32 CFE_ES_RunStatus_Enum_t`  
Run Status and Exit Status identifiers.

See also

enum [CFE\\_ES\\_RunStatus](#)

Definition at line 172 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.18 CFE\_ES\_SystemState\_Enum\_t** `typedef uint32 CFE_ES_SystemState_Enum_t`  
The overall cFE System State.

These values are used with the [CFE\\_ES\\_WaitForSystemState](#) API call to synchronize application startup.

Note

These are defined in order so that relational comparisons e.g. if (STATEA < STATEB) are possible

See also

enum [CFE\\_ES\\_SystemState](#)

Definition at line 229 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.19 CFE\_ES\_TaskId\_t** `typedef CFE_RESOURCEID_BASE_TYPE CFE_ES_TaskId_t`  
A type for Task IDs.

This is the type that is used for any API accepting or returning a Task ID

Definition at line 319 of file default\_cfe\_es\_extern\_typedefs.h.

**12.76.3.20 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.

**12.76.3.21 CFE\_ES\_TaskPriority\_Atom\_t** `typedef uint16 CFE_ES_TaskPriority_Atom_t`  
Type used for task priority in CFE ES as including the commands/telemetry messages.

Note

the valid range is only 0-255 (same as OSAL) but a wider type is used for backward compatibility in binary formats of messages.

Definition at line 358 of file default\_cfe\_es\_extern\_typedefs.h.

## 12.76.4 Enumeration Type Documentation

### 12.76.4.1 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 257 of file default\_cfe\_es\_extern\_typedefs.h.

### 12.76.4.2 CFE\_ES\_AppType enum [CFE\\_ES\\_AppType](#)

Label definitions associated with CFE\_ES\_AppType\_Enum\_t.

#### Enumerator

CFE_ES_AppType_CORE	CFE core application.
CFE_ES_AppType_EXTERNAL	CFE external application.
CFE_ES_AppType_LIBRARY	CFE library.

Definition at line 81 of file default\_cfe\_es\_extern\_typedefs.h.

### 12.76.4.3 CFE\_ES\_ExceptionAction enum [CFE\\_ES\\_ExceptionAction](#)

Label definitions associated with CFE\_ES\_ExceptionAction\_Enum\_t.

#### Enumerator

CFE_ES_ExceptionAction_RESTART_APP	Restart application if exception occurs.
CFE_ES_ExceptionAction_PROC_RESTART	Restart processor if exception occurs.

Definition at line 58 of file default\_cfe\_es\_extern\_typedefs.h.

### 12.76.4.4 CFE\_ES\_LogEntryType enum [CFE\\_ES\\_LogEntryType](#)

Label definitions associated with CFE\_ES\_LogEntryType\_Enum\_t.

#### Enumerator

CFE_ES_LogEntryType_CORE	Log entry from a core subsystem.
CFE_ES_LogEntryType_APPLICATION	Log entry from an application.

Definition at line 234 of file default\_cfe\_es\_extern\_typedefs.h.

#### **12.76.4.5 CFE\_ES\_LogMode** enum [CFE\\_ES\\_LogMode](#)

Label definitions associated with CFE\_ES\_LogMode\_Enum\_t.

Enumerator

CFE_ES_LogMode_OVERWRITE	Overwrite Log Mode.
CFE_ES_LogMode_DISCARD	Discard Log Mode.

Definition at line 35 of file default\_cfe\_es\_extern\_typedefs.h.

#### **12.76.4.6 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 109 of file default\_cfe\_es\_extern\_typedefs.h.

#### **12.76.4.7 CFE\_ES\_SystemState** enum [CFE\\_ES\\_SystemState](#)

Label definitions associated with CFE\_ES\_SystemState\_Enum\_t.

Enumerator

CFE_ES_SystemState_UNDEFINED	reserved
CFE_ES_SystemState_EARLY_INIT	single threaded mode while setting up CFE itself
CFE_ES_SystemState_CORE_STARTUP	core apps (CFE_ES_ObjectTable) are starting (multi-threaded)
CFE_ES_SystemState_CORE_READY	core is ready, starting other external apps/libraries (if any)
CFE_ES_SystemState_APPS_INIT	startup apps have all completed their early init, but not necessarily operational yet
CFE_ES_SystemState_OPERATIONAL	normal operation mode; all apps are RUNNING
CFE_ES_SystemState_SHUTDOWN	reserved for future use, all apps would be STOPPED
CFE_ES_SystemState_MAX	Reserved value, marker for the maximum state.

Definition at line 177 of file default\_cfe\_es\_extern\_typedefs.h.

## 12.77 cfe/modules/es/config/default\_cfe\_es\_fcncodes.h File Reference

### Macros

#### Executive Services Command Codes

- #define CFE\_ES\_NOOP\_CC 0
- #define CFE\_ES\_RESET\_COUNTERS\_CC 1
- #define CFE\_ES\_RESTART\_CC 2
- #define CFE\_ES\_START\_APP\_CC 4
- #define CFE\_ES\_STOP\_APP\_CC 5
- #define CFE\_ES\_RESTART\_APP\_CC 6
- #define CFE\_ES\_RELOAD\_APP\_CC 7
- #define CFE\_ES\_QUERY\_ONE\_CC 8
- #define CFE\_ES\_QUERY\_ALL\_CC 9
- #define CFE\_ES\_CLEAR\_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

### 12.77.1 Detailed Description

Specification for the CFE Executive Services (CFE\_ES) command function codes

#### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

### 12.77.2 Macro Definition Documentation

#### 12.77.2.1 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 540 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.2 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\\_SYS](#)

Definition at line 463 of file default\_cfe\_es\_fcncodes.h.

### 12.77.2.3 CFE\_ES\_DELETE\_CDS\_CC #define CFE\_ES\_DELETE\_CDS\_CC 21

#### Name Delete Critical Data Store

##### Description

This command allows the user to delete a Critical Data Store that was created by an Application that is now no longer executing.

##### Command Mnemonic(s) \$sc\_\$cpu\_ES\_DeleteCDS

##### Command Structure

[CFE\\_ES\\_DeleteCDSCmd\\_t](#)

##### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDPC** - command execution counter will increment
- The [CFE\\_ES\\_CDS\\_DELETED\\_INFO\\_EID](#) informational event message will be generated.
- The specified CDS should no longer appear in a CDS Registry dump generated upon receipt of the [CFE\\_ES\\_DUMP\\_CDS\\_REGISTRY\\_CC](#) command

##### Error Conditions

This command may fail for the following reason(s):

- The specified CDS is the CDS portion of a Critical Table
- The specified CDS is not found in the CDS Registry
- The specified CDS is associated with an Application that is still active
- An error occurred while accessing the CDS memory (see the System Log for more details)

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- A command specific error event message is issued for all error cases

##### Criticality

This command is not critical because it is not possible to delete a CDS that is associated with an active application. However, deleting a CDS does eliminate any "history" that an application may be wishing to keep.

### See also

[CFE\\_ES\\_DUMP\\_CDS\\_REGISTRY\\_CC](#), [CFE\\_TBL\\_DELETE\\_CDS\\_CC](#)

Definition at line 909 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.4 CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC** #define CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC 23**Name** Dump Critical Data Store Registry to a File**Description**

This command allows the user to dump the Critical Data Store Registry to an onboard file.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_WriteCDS2File**Command Structure**

[CFE\\_ES\\_DumpCDSRegistryCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_ES\\_CDS\\_REG\\_DUMP\\_INF\\_EID](#) debug event message will be generated.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_CDS\\_REG\\_DUMP\\_FILE](#) configuration parameter) will be updated with the latest information.

**Error Conditions**

This command may fail for the following reason(s):

- The file name specified could not be parsed
- Error occurred while creating or writing to the dump file

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

**Criticality**

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

**See also**

[CFE\\_ES\\_DELETE\\_CDS\\_CC](#), [CFE\\_TBL\\_DELETE\\_CDS\\_CC](#)

Definition at line 990 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.5 CFE\_ES\_NOOP\_CC** #define CFE\_ES\_NOOP\_CC 0**Name** Executive Services No-Op**Description**

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Executive Services task.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_NOOP

**Command Structure**

[CFE\\_ES\\_NoopCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDPC** - command execution counter will increment
- The [CFE\\_ES\\_BUILD\\_INF\\_EID](#) informational event message will be generated
- The [CFE\\_ES\\_NOOP\\_INF\\_EID](#) informational event message will be generated

**Error Conditions**

This command may fail for the following reason(s):

- The command packet length is incorrect

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- the [CFE\\_ES\\_LEN\\_ERR\\_EID](#) error event message will be generated

**Criticality**

None

**See also**

Definition at line 73 of file default\_cfe\_es\_fncodes.h.

**12.77.2.6 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 792 of file default\_cfe\_es\_fcncodes.h.

## 12.77.2.7 CFE\_ES\_QUERY\_ALL\_CC #define CFE\_ES\_QUERY\_ALL\_CC 9

**Name** Writes all Executive Services Information on all loaded modules to a File

### Description

This command takes the information kept by Executive Services on all of the registered applications and libraries and writes it to the specified file.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_WriteAppInfo2File

### Command Structure

[CFE\\_ES\\_QueryAllCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_ES\\_ALL\\_APPS\\_EID](#) debug event message will be generated.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_APP\\_LOG\\_FILE](#) configuration parameter) will be updated with the latest information.

### Error Conditions

This command may fail for the following reason(s):

- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

### Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

### See also

[CFE\\_ES\\_QUERY\\_ONE\\_CC](#), [CFE\\_ES\\_QUERY\\_ALL\\_TASKS\\_CC](#)

Definition at line 428 of file default\_cfe\_es\_fcncodes.h.

## 12.77.2.8 CFE\_ES\_QUERY\_ALL\_TASKS\_CC #define CFE\_ES\_QUERY\_ALL\_TASKS\_CC 24

**Name** Writes a list of All Executive Services Tasks to a File

### Description

This command takes the information kept by Executive Services on all of the registered tasks and writes it to the specified file.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_WriteTaskInfo2File

### Command Structure

[CFE\\_ES\\_QueryAllTasksCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDPC** - command execution counter will increment
- The [CFE\\_ES\\_TASKINFO\\_EID](#) debug event message will be generated.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_TASK\\_LOG\\_FILE](#) configuration parameter) will be updated with the latest information.

### Error Conditions

This command may fail for the following reason(s):

- The file name specified could not be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- A command specific error event message is issued for all error cases

### Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

### See also

[CFE\\_ES\\_QUERY\\_ALL\\_CC](#), [CFE\\_ES\\_QUERY\\_ONE\\_CC](#)

Definition at line 1032 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.9 CFE\_ES\_QUERY\_ONE\_CC #define CFE\_ES\_QUERY\_ONE\_CC 8****Name** Request Executive Services Information on a specified module**Description**

This command takes the information kept by Executive Services on the specified application or library and telemeters it to the ground.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_QueryApp**Command Structure**[CFE\\_ES\\_QueryOneCmd\\_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_ES\\_ONE\\_APP\\_EID](#) debug event message will be generated.
- Receipt of the [CFE\\_ES\\_OneAppTlm\\_t](#) telemetry packet

**Error Conditions**

This command may fail for the following reason(s):

- The specified name is not recognized as an active application or library

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

**Criticality**

None

**See also**[CFE\\_ES\\_QUERY\\_ALL\\_CC](#), [CFE\\_ES\\_QUERY\\_ALL\\_TASKS\\_CC](#)

Definition at line 386 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.10 CFE\_ES\_RELOAD\_APP\_CC #define CFE\_ES\_RELOAD\_APP\_CC 7****Name** Stops, Unloads, Loads from the command specified File and Restarts an Application**Description**

This command halts and removes the specified Application from the system. Then it immediately loads the Application from the command specified file and restarts it. This command is especially useful for restarting a Command Ingest Application since once it has been stopped, no further commands can come in to restart it.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_ReloadApp

**Command Structure**

[CFE\\_ES\\_ReloadAppCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- The `CFE_ES_RELOAD_APP_DBG_EID` debug event message will be generated. NOTE: This event message only identifies that the reload process has been initiated, not that it has completed.

**Error Conditions**

This command may fail for the following reason(s):

- The specified application filename string cannot be parsed
- The specified application name is not recognized as an active application
- The specified application is one of the cFE's Core applications (ES, EVS, SB, TBL, TIME)

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_ES_CMDEC` - command error counter will increment
- A command specific error event message is issued for all error cases
- Additional information on the reason for command failure may be found in the System Log

**Criticality**

This command is not inherently dangerous, however the restarting of certain applications (e.g. - Spacecraft Attitude and Control) may have a detrimental effect on the spacecraft.

**See also**

[CFE\\_ES\\_START\\_APP\\_CC](#), [CFE\\_ES\\_STOP\\_APP\\_CC](#), [CFE\\_ES\\_RESTART\\_APP\\_CC](#)

Definition at line 350 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.11 CFE\_ES\_RESET\_COUNTERS\_CC #define CFE\_ES\_RESET\_COUNTERS\_CC 1**

**Name** Executive Services Reset Counters

**Description**

This command resets the following counters within the Executive Services housekeeping telemetry:

- Command Execution Counter
- Command Error Counter

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_ResetCtrs

**Command Structure**

[CFE\\_ES\\_ResetCountersCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_ES_CMDPC` - command execution counter and error counter will be reset to zero
- The `CFE_ES_RESET_INF_EID` informational event message will be generated

### Error Conditions

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

### Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

### See also

[CFE\\_ES\\_RESET\\_PR\\_COUNT\\_CC](#)

Definition at line 110 of file default\_cfe\_es\_fcncodes.h.

## 12.77.2.12 CFE\_ES\_RESET\_PR\_COUNT\_CC #define CFE\_ES\_RESET\_PR\_COUNT\_CC 19

**Name** Resets the Processor Reset Counter to Zero

### Description

This command allows the user to reset the Processor Reset Counter to zero. The Processor Reset Counter counts the number of Processor Resets that have occurred so as to identify when a Processor Reset should automatically be upgraded to a full Power-On Reset.

**Command Mnemonic(s)** `$sc_$cpu_ES_ResetPRCn`

### Command Structure

`CFE_ES_ResetPRCountCmd_t`

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- `$sc_$cpu_ES_ProcResetCnt` - Current number of processor resets will go to zero
- The `CFE_ES_RESET_PR_COUNT_EID` informational event message will be generated.

### Error Conditions

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

### Criticality

This command is not critical. The only impact would be that the system would have to have more processor resets before an automatic power-on reset occurred.

### See also

[CFE\\_ES\\_SET\\_MAX\\_PR\\_COUNT\\_CC](#), [CFE\\_ES\\_RESET\\_COUNTERS\\_CC](#)

Definition at line 829 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.13 CFE\_ES\_RESTART\_APP\_CC** #define CFE\_ES\_RESTART\_APP\_CC 6

**Name** Stops, Unloads, Loads using the previous File name, and Restarts an Application

**Description**

This command halts and removes the specified Application from the system. Then it immediately loads the Application from the same filename last used to start. This command is especially useful for restarting a Command Ingest Application since once it has been stopped, no further commands can come in to restart it.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_ResetApp

**Command Structure**

[CFE\\_ES\\_RestartAppCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_ES\\_RESTART\\_APP\\_DBG\\_EID](#) debug event message will be generated. NOTE: This event message only identifies that the restart process has been initiated, not that it has completed.

**Error Conditions**

This command may fail for the following reason(s):

- The original file is missing
- The specified application name is not recognized as an active application
- The specified application is one of the cFE's Core applications (ES, EVS, SB, TBL, TIME)

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases
- Additional information on the reason for command failure may be found in the System Log

**Criticality**

This command is not inherently dangerous, however the restarting of certain applications (e.g. - Spacecraft Attitude and Control) may have a detrimental effect on the spacecraft.

**See also**

[CFE\\_ES\\_START\\_APP\\_CC](#), [CFE\\_ES\\_STOP\\_APP\\_CC](#), [CFE\\_ES\\_RELOAD\\_APP\\_CC](#)

Definition at line 304 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.14 CFE\_ES\_RESTART\_CC #define CFE\_ES\_RESTART\_CC 2****Name** Executive Services Processor / Power-On Reset**Description**

This command restarts the cFE in one of two modes. The Power-On Reset will cause the cFE to restart as though the power were first applied to the processor. The Processor Reset will attempt to retain the contents of the volatile disk and the contents of the Critical Data Store. NOTE: If a requested Processor Reset should cause the Processor Reset Counter (**\$sc\_\$cpu\_ES\_ProcResetCnt**) to exceed OR EQUAL the limit **CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS** (which is reported in housekeeping telemetry as **\$sc\_<-\$cpu\_ES\_MaxProcResets**), the command is **AUTOMATICALLY** upgraded to a Power-On Reset.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_ProcessorReset, \$sc\_\$cpu\_ES\_PowerOnReset**Command Structure****CFE\_ES\_RestartCmd\_t****Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_ES\_ProcResetCnt** - processor reset counter will increment (processor reset) or reset to zero (power-on reset)
  - **\$sc\_\$cpu\_ES\_ResetType** - processor reset type will be updated
  - **\$sc\_\$cpu\_ES\_ResetSubtype** - processor reset subtype will be updated
  - New entries in the Exception Reset Log and System Log can be found
- NOTE: Verification of a Power-On Reset is shown through the loss of data nominally retained through a Processor Reset
- NOTE: Since the reset of the processor resets the command execution counter (**\$sc\_\$cpu\_ES\_CMDPC**), this counter **CANNOT** be used to verify command execution.

**Error Conditions**

This command may fail for the following reason(s):

- The **Restart Type** was not a recognized value.

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- the **CFE\_ES\_BOOT\_ERR\_EID** error event message will be generated

**Criticality**

This command is, by definition, dangerous. Significant loss of data will occur. All processes and the cFE itself will be stopped and restarted. With the Power-On reset option, all data on the volatile disk and the contents of the Critical Data Store will be lost.

**See also****CFE\_ES\_RESET\_PR\_COUNT\_CC, CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC**

Definition at line 162 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.15 CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC** #define CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC 22**Name** Telemeter Memory Pool Statistics**Description**

This command allows the user to obtain a snapshot of the statistics maintained for a specified memory pool.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_PoolStats**Command Structure**

[CFE\\_ES\\_SendMemPoolStatsCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_ES\\_TLM\\_POOL\\_STATS\\_INFO\\_EID](#) debug event message will be generated.
- The [Memory Pool Statistics Telemetry Packet](#) is produced

**Error Conditions**

This command may fail for the following reason(s):

- The specified handle is not associated with a known memory pool

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

**Criticality**

**An incorrect Memory Pool Handle value can cause a system crash.** Extreme care should be taken to ensure the memory handle value used in the command is correct.

**See also**

Definition at line 948 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.16 CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC** #define CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC 20**Name** Configure the Maximum Number of Processor Resets before a Power-On Reset**Description**

This command allows the user to specify the number of Processor Resets that are allowed before the next Processor Reset is upgraded to a Power-On Reset.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_SetMaxPRCntr

#### Command Structure

[CFE\\_ES\\_SetMaxPRCountCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- `$sc_$cpu_ES_MaxProcResets` - Current maximum number of processor resets before an automatic power-on reset will go to the command specified value.
- The [CFE\\_ES\\_SET\\_MAX\\_PR\\_COUNT\\_EID](#) informational event message will be generated.

#### Error Conditions

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

#### Criticality

If the operator were to set the Maximum Processor Reset Count to too high a value, the processor would require an inordinate number of consecutive processor resets before an automatic power-on reset would occur. This could potentially leave the spacecraft without any control for a significant amount of time if a processor reset fails to clear a problem.

#### See also

[CFE\\_ES\\_RESET\\_PR\\_COUNT\\_CC](#)

Definition at line 867 of file default\_cfe\_es\_fcncodes.h.

### 12.77.2.17 CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC #define CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC 16

**Name** Set Performance Analyzer's Filter Masks

#### Description

This command sets the Performance Analyzer's Filter Masks.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_LAFilterMask

#### Command Structure

[CFE\\_ES\\_SetPerfFilterMaskCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- `$sc_$cpu_ES_PerfFltrMask[MaskCnt]` - the current performance filter mask value(s) should reflect the commanded value
- The [CFE\\_ES\\_PERF\\_FILTMSKCMD\\_EID](#) debug event message will be generated.

## Error Conditions

This command may fail for the following reason(s):

- The Filter Mask ID number is out of range

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- A command specific error event message is issued for all error cases

## Criticality

Changing the filter masks may cause a small change in the Performance Analyzer's CPU utilization.

## See also

[CFE\\_ES\\_START\\_PERF\\_DATA\\_CC](#), [CFE\\_ES\\_STOP\\_PERF\\_DATA\\_CC](#), [CFE\\_ES\\_SET\\_PERF\\_TRIGGER\\_MASK\\_CC](#)

Definition at line 715 of file default\_cfe\_es\_fcncodes.h.

### 12.77.2.18 CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC #define CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC 17

**Name** Set Performance Analyzer's Trigger Masks

## Description

This command sets the Performance Analyzer's Trigger Masks.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_LATriggerMask

## Command Structure

[CFE\\_ES\\_SetPerfTriggerMaskCmd\\_t](#)

## Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_ES\_PerfTrigMask[MaskCnt]** - the current performance trigger mask value(s) should reflect the commanded value
- The [CFE\\_ES\\_PERF\\_TRIGMSKCMD\\_EID](#) debug event message will be generated.

## Error Conditions

This command may fail for the following reason(s):

- The Trigger Mask ID number is out of range

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- A command specific error event message is issued for all error cases

## Criticality

Changing the trigger masks may cause a small change in the Performance Analyzer's CPU utilization.

## See also

[CFE\\_ES\\_START\\_PERF\\_DATA\\_CC](#), [CFE\\_ES\\_STOP\\_PERF\\_DATA\\_CC](#), [CFE\\_ES\\_SET\\_PERF\\_FILTER\\_MASK\\_CC](#)

Definition at line 752 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.19 CFE\_ES\_START\_APP\_CC** #define CFE\_ES\_START\_APP\_CC 4**Name** Load and Start an Application**Description**

This command starts the specified application with the specified start address, stack size, etc options.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_StartApp**Command Structure**[CFE\\_ES\\_StartAppCmd\\_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_ES\\_START\\_INF\\_EID](#) informational event message will be generated

**Error Conditions**

This command may fail for the following reason(s):

- The specified application filename string cannot be parsed
- The specified application entry point is an empty string
- The specified application name is an empty string
- The specified priority is greater than 255
- The specified exception action is neither [CFE\\_ES\\_ExceptionAction\\_RESTART\\_APP](#) (0) or [CFE\\_ES\\_ExceptionAction\\_PROC](#) (1)
- The Operating System was unable to load the specified application file

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

**Criticality**

This command is not inherently dangerous although system resources could be taxed beyond their limits with the starting of erroneous or invalid applications.

**See also**[CFE\\_ES\\_STOP\\_APP\\_CC](#), [CFE\\_ES\\_RESTART\\_APP\\_CC](#), [CFE\\_ES\\_RELOAD\\_APP\\_CC](#)

Definition at line 205 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.20 CFE\_ES\_START\_PERF\_DATA\_CC** #define CFE\_ES\_START\_PERF\_DATA\_CC 14

**Name** Start Performance Analyzer

**Description**

This command causes the Performance Analyzer to begin collecting data using the specified trigger mode.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_StartLAData

**Command Structure**

[CFE\\_ES\\_StartPerfDataCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_ES\\_PerfState](#) - Current performance analyzer state will change to either WAITING FOR TRIGGER or, if conditions are appropriate fast enough, TRIGGERED.
- [\\$sc\\_\\$cpu\\_ES\\_PerfMode](#) - Performance Analyzer Mode will change to the commanded trigger mode (TRIGGER START, TRIGGER CENTER, or TRIGGER END).
- [\\$sc\\_\\$cpu\\_ES\\_PerfTrigCnt](#) - Performance Trigger Count will go to zero
- [\\$sc\\_\\$cpu\\_ES\\_PerfDataStart](#) - Data Start Index will go to zero
- [\\$sc\\_\\$cpu\\_ES\\_PerfDataEnd](#) - Data End Index will go to zero
- [\\$sc\\_\\$cpu\\_ES\\_PerfDataCnt](#) - Performance Data Counter will go to zero
- The [CFE\\_ES\\_PERF\\_STARTCMD\\_EID](#) debug event message will be generated.

**Error Conditions**

This command may fail for the following reason(s):

- A previous [CFE\\_ES\\_STOP\\_PERF\\_DATA\\_CC](#) command has not completely finished.
- An invalid trigger mode is requested.

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

**Criticality**

This command is not inherently dangerous but may cause a small increase in CPU utilization as the performance analyzer data is collected.

**See also**

[CFE\\_ES\\_STOP\\_PERF\\_DATA\\_CC](#), [CFE\\_ES\\_SET\\_PERF\\_FILTER\\_MASK\\_CC](#), [CFE\\_ES\\_SET\\_PERF\\_TRIGGER\\_MASK\\_CC](#)

Definition at line 628 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.21 CFE\_ES\_STOP\_APP\_CC** #define CFE\_ES\_STOP\_APP\_CC 5**Name** Stop and Unload Application**Description**

This command halts and removes the specified Application from the system. **NOTE:** This command should never be used on the Command Ingest application. This would prevent further commands from entering the system. If Command Ingest needs to be stopped and restarted, use [CFE\\_ES\\_RESTART\\_APP\\_CC](#) or [CFE\\_ES\\_RELOAD\\_APP\\_CC](#).

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_StopApp**Command Structure**[CFE\\_ES\\_StopAppCmd\\_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDPC** - command execution counter will increment
- The [CFE\\_ES\\_STOP\\_DBG\\_EID](#) debug event message will be generated. NOTE: This event message only identifies that the stop request has been initiated, not that it has completed.
- Once the stop has successfully completed, the list of Applications and Tasks created in response to the **\$sc\_\$cpu\_ES\_WriteAppInfo2File**, **\$sc\_\$cpu\_ES\_WriteTaskInfo2File** should no longer contain the specified application.
- **\$sc\_\$cpu\_ES\_RegTasks** - number of tasks will decrease after tasks associated with app (main task and any child tasks) are stopped
- **\$sc\_\$cpu\_ES\_RegExtApps** - external application counter will decrement after app is cleaned up

**Error Conditions**

This command may fail for the following reason(s):

- The specified application name is not recognized as an active application
- The specified application is one of the cFE's Core applications (ES, EVS, SB, TBL, TIME)

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- A command specific error event message is issued for all error cases
- Additional information on the reason for command failure may be found in the System Log

**Criticality**

This command is not inherently dangerous, however the removal of certain applications (e.g. - Spacecraft Attitude and Control) may have a detrimental effect on the spacecraft.

**See also**[CFE\\_ES\\_START\\_APP\\_CC](#), [CFE\\_ES\\_RESTART\\_APP\\_CC](#), [CFE\\_ES\\_RELOAD\\_APP\\_CC](#)

Definition at line 258 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.22 CFE\_ES\_STOP\_PERF\_DATA\_CC** #define CFE\_ES\_STOP\_PERF\_DATA\_CC 15

**Name** Stop Performance Analyzer and write data file

**Description**

This command stops the Performance Analyzer from collecting any more data, and writes all previously collected performance data to a log file.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_StopLAData

**Command Structure**

[CFE\\_ES\\_StopPerfDataCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_ES\\_PerfState](#) - Current performance analyzer state will change to IDLE.
- The [CFE\\_ES\\_PERF\\_STOPCMD\\_EID](#) debug event message will be generated to indicate that data collection has been stopped. NOTE: Performance log data is written to the file as a background job. This event indicates that the file write process is initiated, not that it has completed.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_PERF\\_DUMP\\_FILENAME](#) configuration parameter) will be updated with the latest information.

**Error Conditions**

This command may fail for the following reason(s):

- The file name specified could not be parsed
- Log data from a previous Stop Performance Analyzer command is still being written to a file.

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

NOTE: The performance analyzer data collection will still be stopped in the event of an error parsing the log file name or writing the log file.

**Criticality**

This command is not inherently dangerous. However, depending on configuration, performance data log files may be large in size and thus may fill the available storage.

**See also**

[CFE\\_ES\\_START\\_PERF\\_DATA\\_CC](#), [CFE\\_ES\\_SET\\_PERF\\_FILTER\\_MASK\\_CC](#), [CFE\\_ES\\_SET\\_PERF\\_TRIGGER\\_MASK\\_CC](#)

Definition at line 678 of file `default_cfe_es_fcncodes.h`.

**12.77.2.23 CFE\_ES\_WRITE\_ER\_LOG\_CC #define CFE\_ES\_WRITE\_ER\_LOG\_CC 13**

**Name** Writes Exception and Reset Log to a File

**Description**

This command causes the contents of the Executive Services Exception and Reset Log to be written to the specified file.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_WriteERLog2File

**Command Structure**

[CFE\\_ES\\_WriteERLogCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_ES\\_ERLOG2\\_EID](#) debug event message will be generated.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_ER\\_LOG\\_FILE](#) configuration parameter) will be updated with the latest information.

**Error Conditions**

This command may fail for the following reason(s):

- A previous request to write the ER log has not yet completed
- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_ES\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

**Criticality**

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

**See also**

[CFE\\_ES\\_CLEAR\\_SYSLOG\\_CC](#), [CFE\\_ES\\_WRITE\\_SYSLOG\\_CC](#), [CFE\\_ES\\_CLEAR\\_ER\\_LOG\\_CC](#)

Definition at line 583 of file default\_cfe\_es\_fcncodes.h.

**12.77.2.24 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\\_SYS](#)

Definition at line 506 of file default\_cfe\_es\_fcncodes.h.

## 12.78 cfe/modules/es/config/default\_cfe\_es\_interface\_cfg.h File Reference

**Macros**

- #define CFE\_MISSION\_ES\_MAX\_APPLICATIONS 16
- #define CFE\_MISSION\_ES\_PERF\_MAX\_IDS 128
- #define CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS 17
- #define CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH 16
- #define CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_ES\_CrcType\_CRC\_16
- #define CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN (CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH + CFE\_MISSION\_MAX\_API\_LEN + 4)

**Checksum/CRC algorithm identifiers**

- #define CFE\_MISSION\_ES\_CRC\_8 CFE\_ES\_CrcType\_CRC\_8 /\* 1 \*/
- #define CFE\_MISSION\_ES\_CRC\_16 CFE\_ES\_CrcType\_CRC\_16 /\* 2 \*/
- #define CFE\_MISSION\_ES\_CRC\_32 CFE\_ES\_CrcType\_CRC\_32 /\* 3 \*/

### 12.78.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.78.2 Macro Definition Documentation

**12.78.2.1 CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN** #define CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN (CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH + CFE\_MISSION\_MAX\_API\_LEN + 4)

**Purpose** Maximum Length of Full CDS Name in messages

#### Description:

Indicates the maximum length (in characters) of the entire CDS name of the following form: "ApplicationName.CDSName"

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

#### Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 138 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.2 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 104 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.3 CFE\_MISSION\_ES\_CRC\_16** #define CFE\_MISSION\_ES\_CRC\_16 CFE\_ES\_CrcType\_CRC\_16 /\* 2 \*/

Definition at line 146 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.4 CFE\_MISSION\_ES\_CRC\_32** #define CFE\_MISSION\_ES\_CRC\_32 CFE\_ES\_CrcType\_CRC\_32 /\* 3 \*/  
Definition at line 147 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.5 CFE\_MISSION\_ES\_CRC\_8** #define CFE\_MISSION\_ES\_CRC\_8 CFE\_ES\_CrcType\_CRC\_8 /\* 1 \*/  
Definition at line 145 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.6 CFE\_MISSION\_ES\_DEFAULT\_CRC** #define CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_ES\_CrcType\_CRC\_16

**Purpose** Mission Default CRC algorithm

**Description:**

Indicates the which CRC algorithm should be used as the default for verifying the contents of Critical Data Stores and when calculating Table Image data integrity values.

**Limits**

Currently only CFE\_ES\_CrcType\_CRC\_16 is supported (see brief in CFE\_ES\_CrcType\_Enum definition in [cfe\\_es\\_api\\_typedefs.h](#))

Definition at line 118 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.7 CFE\_MISSION\_ES\_MAX\_APPLICATIONS** #define CFE\_MISSION\_ES\_MAX\_APPLICATIONS 16

**Purpose** Mission Max Apps in a message

**Description:**

Indicates the maximum number of apps in a telemetry housekeeping message

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 49 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.8 CFE\_MISSION\_ES\_PERF\_MAX\_IDS** #define CFE\_MISSION\_ES\_PERF\_MAX\_IDS 128

**Purpose** Define Max Number of Performance IDs for messages

**Description:**

Defines the maximum number of perf ids allowed 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 66 of file default\_cfe\_es\_interface\_cfg.h.

**12.78.2.9 CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS #define CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS 17**

**Purpose** Maximum number of block sizes in pool structures

Description:

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS. This definition is used as the array size with the pool stats structure, and therefore should be consistent across all CPUs in a mission, as well as with the ground station.

There is also a platform-specific limit which may be fewer than this value.

Limits:

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

Definition at line 87 of file default\_cfe\_es\_interface\_cfg.h.

**12.79 cfe/modules/es/config/default\_cfe\_es\_internal\_cfg.h File Reference****Macros**

- #define CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY 68
- #define CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING "/cf"
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING "/ram"
- #define CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS 32
- #define CFE\_PLATFORM\_ES\_MAX\_LIBRARIES 10
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES 20
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE 256
- #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072
- #define CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE 30
- #define CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS 8
- #define CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE 1000
- #define CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT 5
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE 512
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS 4096
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED 30
- #define CFE\_PLATFORM\_ES\_CDS\_SIZE (128 \* 1024)
- #define CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE (1024 \* 1024)
- #define CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN 4
- #define CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE "/cf/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE "/ram/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE "/ram/cfe\_es\_app\_info.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE "/ram/cfe\_es\_taskinfo.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE "/ram/cfe\_es\_syslog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE "/ram/cfe\_erlog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME "/ram/cfe\_es\_perf.dat"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE "/ram/cfe\_cds\_reg.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE 0
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE 1
- #define CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE 10000
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE 0

- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL ~CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE 0
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL ~CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY 200
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY 20
- #define CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS 50
- #define CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES 512
- #define CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS 2
- #define CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS 17
- #define CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS 10
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 32
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 32
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC 50
- #define CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC 1000

### 12.79.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.79.2 Macro Definition Documentation

#### 12.79.2.1 CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT `#define CFE_PLATFORM_ES_APP_KILL_TIMEOUT 5`

**Purpose** Define ES Application Kill Timeout

##### Description:

ES Application Kill Timeout. This parameter controls the number of "scan periods" that ES will wait for an application to Exit after getting the signal Delete, Reload or Restart. The sequence works as follows:

1. ES will set the control request for an App to Delete/Restart/Reload and set this kill timer to the value in this parameter.
2. If the App is responding and Calls it's RunLoop function, it will drop out of its main loop and call CFE\_ES->\_ExitApp. Once it calls Exit App, then ES can delete, restart, or reload the app the next time it scans the app table.
3. If the App is not responding, the ES App will decrement this Kill Timeout value each time it runs. If the timeout value reaches zero, ES will kill the app.

The Kill timeout value depends on the [CFE\\_PLATFORM\\_ES\\_APP\\_SCAN\\_RATE](#). If the Scan Rate is 1000, or 1 second, and this [CFE\\_PLATFORM\\_ES\\_APP\\_KILL\\_TIMEOUT](#) is set to 5, then it will take 5 seconds to kill a non-responding App. If the Scan Rate is 250, or 1/4 second, and the [CFE\\_PLATFORM\\_ES\\_APP\\_KILL\\_TIMEOUT](#) is set to 2, then it will take 1/2 second to time out.

##### Limits

There is a lower limit of 1 and an upper limit of 100 on this configuration parameter. Units are number of [CFE\\_PLATFORM\\_ES\\_APP\\_SCAN\\_RATE](#) cycles.

Definition at line 232 of file default\_cfe\_es\_internal\_cfg.h.

#### 12.79.2.2 CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE `#define CFE_PLATFORM_ES_APP_SCAN_RATE 1000`

**Purpose** Define ES Application Control Scan Rate

##### Description:

ES Application Control Scan Rate. This parameter controls the speed that ES scans the Application Table looking for App Delete/Restart/Reload requests. All Applications are deleted, restarted, or reloaded by the ES Application. ES will periodically scan for control requests to process. The scan rate is controlled by this parameter, which is given in milliseconds. A value of 1000 means that ES will scan the Application Table once per second. Be careful not to set the value of this too low, because ES will use more CPU cycles scanning the table.

**Limits**

There is a lower limit of 100 and an upper limit of 20000 on this configuration parameter. millisecond units.

Definition at line 203 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.3 CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SI←  
ZE 80000

Definition at line 774 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.4 CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES** #define CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRI←  
ES 512

**Purpose** Define Maximum Number of Registered CDS Blocks

**Description:**

Maximum number of registered CDS Blocks

**Limits**

There is a lower limit of 8. There are no restrictions on the upper limit however, the maximum number of CDS entries is system dependent and should be verified.

Definition at line 664 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.5 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SI←  
ZE\_01 8

**Purpose** Define ES Critical Data Store Memory Pool Block Sizes

**Description:**

Intermediate ES Critical Data Store Memory Pool Block Sizes

**Limits**

These sizes MUST be increasing and MUST be an integral multiple of 4.

Definition at line 758 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.6 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SI←  
ZE\_02 16

Definition at line 759 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.7 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SI←  
ZE\_03 32

Definition at line 760 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.8 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 48

Definition at line 761 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.9 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 64

Definition at line 762 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.10 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96

Definition at line 763 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.11 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128

Definition at line 764 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.12 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160

Definition at line 765 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.13 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256

Definition at line 766 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.14 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512

Definition at line 767 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.15 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024

Definition at line 768 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.16 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048

Definition at line 769 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.17 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096

Definition at line 770 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.18 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192

Definition at line 771 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.19 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384

Definition at line 772 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.20 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 32768

Definition at line 773 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.21 CFE\_PLATFORM\_ES\_CDS\_SIZE** #define CFE\_PLATFORM\_ES\_CDS\_SIZE (128 \* 1024)

**Purpose** Define Critical Data Store Size

**Description:**

Defines the Critical Data Store (CDS) area size in bytes size. The CDS is one of four memory areas that are preserved during a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as USER\_RESERVED\_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

**Limits**

There is a lower limit of 8192 and an upper limit of `UINT_MAX` (4 Gigabytes) on this configuration parameter.

Definition at line 309 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.22 CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE "/ram/cfe\_es\_app\_info.log"

**Purpose** Default Application Information Filename

**Description:**

The value of this constant defines the filename used to store information pertaining to all of the Applications that are registered with Executive Services. This filename is used only when no filename is specified in the command to query all system apps.

**Limits**

The length of each string, including the NULL terminator cannot exceed the `OS_MAX_PATH_LEN` value.

Definition at line 391 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.23 CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE #define CFE_PLATFORM_ES_DEFAULT_CD←  
S_REG_DUMP_FILE "/ram/cfe_cds_reg.log"
```

**Purpose** Default Critical Data Store Registry Filename

**Description:**

The value of this constant defines the filename used to store the Critical Data Store Registry. This filename is used only when no filename is specified in the command to stop performance data collecting.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 465 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.24 CFE_PLATFORM_ES_DEFAULT_ER_LOG_FILE #define CFE_PLATFORM_ES_DEFAULT_ER_LOG_FI←  
LE "/ram/cfe_erlog.log"
```

**Purpose** Default Exception and Reset (ER) Log Filename

**Description:**

The value of this constant defines the filename used to store the Exception and Reset (ER) Log. This filename is used only when no filename is specified in the command to dump the ER log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 437 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.25 CFE_PLATFORM_ES_DEFAULT_PERF_DUMP_FILENAME #define CFE_PLATFORM_ES_DEFAULT_←  
PERF_DUMP_FILENAME "/ram/cfe_es_perf.dat"
```

**Purpose** Default Performance Data Filename

**Description:**

The value of this constant defines the filename used to store the Performance Data. This filename is used only when no filename is specified in the command to stop performance data collecting.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 451 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.26 CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE** #define CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE 0

**Purpose** Define Default System Log Mode following Power On Reset

**Description:**

Defines the default mode for the operation of the ES System log following a power on reset. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest message in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. This constant may hold a value of either 0 or 1 depending on the desired default. Overwrite Mode = 0, Discard Mode = 1.

**Limits**

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 483 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.27 CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE** #define CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE 1

**Purpose** Define Default System Log Mode following Processor Reset

**Description:**

Defines the default mode for the operation of the ES System log following a processor reset. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest message in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. This constant may hold a value of either 0 or 1 depending on the desired default. Overwrite Mode = 0, Discard Mode = 1.

**Limits**

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 501 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.28 CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE 8192

**Purpose** Define Default Stack Size for an Application

**Description:**

This parameter defines a default stack size. This parameter is used by the cFE Core Applications.

**Limits**

There is a lower limit of 2048. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 651 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.29 CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE #define CFE_PLATFORM_ES_DEFAULT_SYSLOG_FI←  
LE "/ram/cfe_es_syslog.log"
```

**Purpose** Default System Log Filename

**Description:**

The value of this constant defines the filename used to store important information (as ASCII text strings) that might not be able to be sent in an Event Message. This filename is used only when no filename is specified in the command to dump the system log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 422 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.30 CFE_PLATFORM_ES_DEFAULT_TASK_LOG_FILE #define CFE_PLATFORM_ES_DEFAULT_TASK_LO←  
G_FILE "/ram/cfe_es_taskinfo.log"
```

**Purpose** Default Application Information Filename

**Description:**

The value of this constant defines the filename used to store information pertaining to all of the Applications that are registered with Executive Services. This filename is used only when no filename is specified in the the command to query all system tasks.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 406 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.31 CFE_PLATFORM_ES_ER_LOG_ENTRIES #define CFE_PLATFORM_ES_ER_LOG_ENTRIES 20
```

**Purpose** Define Max Number of ER (Exception and Reset) log entries

**Description:**

Defines the maximum number of ER (Exception and Reset) log entries

**Limits**

There is a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of log entries is system dependent and should be verified.

Definition at line 130 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.32 CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE** #define CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE 256

**Purpose** Maximum size of CPU Context in ES Error Log

**Description:**

This should be large enough to accommodate the CPU context information supplied by the PSP on the given platform.

**Limits:**

Must be greater than zero and a multiple of sizeof(uint32). Limited only by the available memory and the number of entries in the error log. Any context information beyond this size will be truncated.

Definition at line 144 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.33 CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS** #define CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS 32

**Purpose** Define Max Number of Applications

**Description:**

Defines the maximum number of applications that can be loaded into the system. This number does not include child tasks.

**Limits**

There is a lower limit of 6. The lower limit corresponds to the cFE internal applications. There are no restrictions on the upper limit however, the maximum number of applications is system dependent and should be verified. AppIDs that are checked against this configuration are defined by a 32 bit data word.

Definition at line 103 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.34 CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000

Definition at line 747 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.35 CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS** #define CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS 8

**Purpose** Define Max Number of Generic Counters

**Description:**

Defines the maximum number of Generic Counters that can be registered.

**Limits**

This parameter has a lower limit of 1 and an upper limit of 65535.

Definition at line 184 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.36 CFE\_PLATFORM\_ES\_MAX\_LIBRARIES** #define CFE\_PLATFORM\_ES\_MAX\_LIBRARIES 10

**Purpose** Define Max Number of Shared libraries

**Description:**

Defines the maximum number of cFE Shared libraries that can be loaded into the system.

**Limits**

There is a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of libraries is system dependent and should be verified.

Definition at line 117 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.37 CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS** #define CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS 10

**Purpose** Maximum number of memory pools

**Description:**

The upper limit for the number of memory pools that can concurrently exist within the system.

The CFE\_SB and CFE\_TBL core subsystems each define a memory pool.  
Individual applications may also create memory pools, so this value should be set sufficiently high enough to support the applications being used on this platform.

**Limits:**

Must be at least 2 to support CFE core - SB and TBL pools. No specific upper limit.

Definition at line 712 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.38 CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS** #define CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS 2

**Purpose** Define Number of Processor Resets Before a Power On Reset

**Description:**

Number of Processor Resets before a Power On Reset is called. If set to 2, then 2 processor resets will occur, and the 3rd processor reset will be a power on reset instead.

**Limits**

There is a lower limit of 0. There are no restrictions on the upper limit however, the maximum number of processor resets may be system dependent and should be verified.

Definition at line 679 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.39 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 8

**Purpose** Define Default ES Memory Pool Block Sizes

**Description:**

Default Intermediate ES Memory Pool Block Sizes. If an application is using the CFE\_ES Memory Pool APIs (CFE\_ES\_PoolCreate, CFE\_ES\_PoolCreateNoSem, CFE\_ES\_GetPoolBuf and CFE\_ES\_PutPoolBuf) but finds these sizes inappropriate for their use, they may wish to use the CFE\_ES\_PoolCreateEx API to specify their own intermediate block sizes

**Limits**

These sizes MUST be increasing and MUST be an integral multiple of 4. Also, CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE must be larger than CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE and both CFE\_PLATFORM\_TB\_MAX\_SNGL\_TABLE\_SIZE and CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE. Note that if Table Services have been removed from the CFE, the table size limits are still enforced although the table size definitions may be reduced.

Definition at line 731 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.40 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16  
Definition at line 732 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.41 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 32  
Definition at line 733 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.42 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 48  
Definition at line 734 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.43 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 64  
Definition at line 735 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.44 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 96  
Definition at line 736 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.45 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 128  
Definition at line 737 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.46 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 160  
Definition at line 738 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.47 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 256  
Definition at line 739 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.48 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 512  
Definition at line 740 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.49 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 1024  
Definition at line 741 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.50 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 2048  
Definition at line 742 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.51 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 4096  
Definition at line 743 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.52 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 8192  
Definition at line 744 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.53 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 16384  
Definition at line 745 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.54 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 32768  
Definition at line 746 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.55 CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN** #define CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN 4

#### Purpose

Define Memory Pool Alignment Size

##### Description:

Ensures that buffers obtained from a memory pool are aligned to a certain minimum block size. Note the allocator will always align to the minimum required by the CPU architecture. This may be set greater than the CPU requirement as desired for optimal performance.

For some architectures/applications it may be beneficial to set this to the cache line size of the target CPU, or to use special SIMD instructions that require a more stringent memory alignment.

##### Limits

This must always be a power of 2, as it is used as a binary address mask.

Definition at line 348 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.56 CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING** #define CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING "/cf"

**Purpose** Default virtual path for persistent storage

**Description:**

This configures the default location in the virtual file system for persistent/non-volatile storage. Files such as the startup script, app/library dynamic modules, and configuration tables are expected to be stored in this directory.

Definition at line 71 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.57 CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE** #define CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE "/cf/cfe\_es\_startup.scr"

**Purpose** ES Nonvolatile Startup Filename

**Description:**

The value of this constant defines the path and name of the file that contains a list of modules that will be loaded and started by the cFE after the cFE finishes its startup sequence.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 362 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.58 CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE** #define CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE 30

**Purpose** Define Number of entries in the ES Object table

**Description:**

Defines the number of entries in the ES Object table. This table controls the core cFE startup.

**Limits**

There is a lower limit of 15. There are no restrictions on the upper limit however, the maximum object table size is system dependent and should be verified.

Definition at line 173 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.59 CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY** #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY 20

**Purpose** Define Performance Analyzer Child Task Delay

**Description:**

This parameter defines the delay time (in milliseconds) between performance data file writes performed by the Executive Services Performance Analyzer Child Task.

**Limits**

It is recommended this parameter be greater than or equal to 20ms. This parameter is limited by the maximum value allowed by the data type. In this case, the data type is an unsigned 32-bit integer, so the valid range is 0 to 0xFFFFFFFF.

Definition at line 625 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.60 CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY** #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY 200

**Purpose** Define Performance Analyzer Child Task Priority

**Description:**

This parameter defines the priority of the child task spawned by the Executive Services to write performance data to a file. Lower numbers are higher priority, with 1 being the highest priority in the case of a child task.

**Limits**

Valid range for a child task is 1 to 255 however, the priority cannot be higher (lower number) than the ES parent application priority.

Definition at line 596 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.61 CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE 4096

**Purpose** Define Performance Analyzer Child Task Stack Size

**Description:**

This parameter defines the stack size of the child task spawned by the Executive Services to write performance data to a file.

**Limits**

It is recommended this parameter be greater than or equal to 4KB. This parameter is limited by the maximum value allowed by the data type. In this case, the data type is an unsigned 32-bit integer, so the valid range is 0 to 0xFFFFFFFF.

Definition at line 610 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.62 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 517 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.63 CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS** #define CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS 50

**Purpose** Define Performance Analyzer Child Task Number of Entries Between Delay

**Description:**

This parameter defines the number of performance analyzer entries the Performance Analyzer Child Task will write to the file between delays.

Definition at line 635 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.64 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 537 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.65 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 548 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.66 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 527 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.67 CFE_PLATFORM_ES_PERF_TRIGMASK_ALL #define CFE_PLATFORM_ES_PERF_TRIGMASK_A←  
LL ~CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
```

**Purpose** Define Filter Trigger Setting for Enabling All Performance Entries

**Description:**

Defines the trigger mask for enabling all performance data entries. The value is a bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 570 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.68 CFE_PLATFORM_ES_PERF_TRIGMASK_INIT #define CFE_PLATFORM_ES_PERF_TRIGMASK_IN←  
IT CFE_PLATFORM_ES_PERF_TRIGMASK_NONE
```

**Purpose** Define Default Filter Trigger Setting for Performance Data Buffer

**Description:**

Defines the default trigger mask for the performance data buffer. The value is a 32-bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 581 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.69 CFE_PLATFORM_ES_PERF_TRIGMASK_NONE #define CFE_PLATFORM_ES_PERF_TRIGMASK_NO←  
NE 0
```

**Purpose** Define Default Filter Trigger Setting for Disabling All Performance Entries

**Description:**

Defines the default trigger mask for disabling all performance data entries. The value is a bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 559 of file default\_cfe\_es\_internal\_cfg.h.

```
12.79.2.70 CFE_PLATFORM_ES_POOL_MAX_BUCKETS #define CFE_PLATFORM_ES_POOL_MAX_BUCKETS 17
```

**Purpose** Maximum number of block sizes in pool structures

**Description:**

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS.

**Limits:**

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

The ES and CDS block size lists must correlate with this value

Definition at line 694 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.71 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 87 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.72 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 268 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.73 CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED 30

**Purpose** Percentage of Ram Disk Reserved for Decompressing Apps

**Description:**

The **CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED** parameter is used to make sure that the Volatile ( RAM ) Disk has a defined amount of free space during a processor reset. The cFE uses the Volatile disk to decompress cFE applications during system startup. If this Volatile disk happens to get filled with logs and misc files, then a processor reset may not work, because there will be no room to decompress cFE apps. To solve that problem, this parameter sets the "Low Water Mark" for disk space on a Processor reset. It should be set to allow the largest cFE Application to be decompressed. During a Processor reset, if there is not sufficient space left on the disk, it will be re-formatted in order to clear up some space.

This feature can be turned OFF by setting the parameter to 0.

**Limits**

There is a lower limit of 0 and an upper limit of 75 on this configuration parameter. Units are percentage. A setting of zero will turn this feature off.

Definition at line 292 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.74 CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE 512

**Purpose** ES Ram Disk Sector Size

**Description:**

Defines the ram disk sector size. The ram disk is 1 of 4 memory areas that are preserved on a processor reset.  
NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as USER\_RESERVED\_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

**Limits**

There is a lower limit of 128. There are no restrictions on the upper limit however, the maximum RAM disk sector size is system dependent and should be verified.

Definition at line 250 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.75 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 44 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.76 CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define ES Task Stack Size

**Description:**

Defines the cFE\_ES Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 59 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.77 CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC 1000

**Purpose** Startup script timeout

**Description:**

The upper limit for the total amount of time that all apps listed in the CFE ES startup script may take to all become ready.

Unlike the "core" app timeout, this is a soft limit; if the allotted time is exceeded, it probably indicates an issue with one of the apps, but does not cause CFE ES to take any additional action other than logging the event to the syslog.

Units are in milliseconds

**Limits:**

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 814 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.78 CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC 50

**Purpose** Poll timer for startup sync delay

**Description:**

During startup, some tasks may need to synchronize their own initialization with the initialization of other applications in the system.

CFE ES implements an API to accomplish this, that performs a task delay (sleep) while polling the overall system state until other tasks are ready.

This value controls the amount of time that the CFE\_ES\_ApplicationSyncDelay will sleep between each check of the system state. This should be large enough to allow other tasks to run, but not so large as to noticeably delay the startup completion.

Units are in milliseconds

**Limits:**

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 796 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.79 CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE** #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072

**Purpose** Define Size of the cFE System Log.

**Description:**

Defines the size in bytes of the cFE system log. The system log holds variable length strings that are terminated by a linefeed and null character.

**Limits**

There is a lower limit of 512. There are no restrictions on the upper limit however, the maximum system log size is system dependent and should be verified.

Definition at line 159 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.80 CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE** #define CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE (1024 \* 1024)

**Purpose** Define User Reserved Memory Size

**Description:**

User Reserved Memory Size. This is the size in bytes of the cFE User reserved Memory area. This is a block of memory that is available for cFE application use. The address is obtained by calling [CFE\\_PSP\\_GetUserReservedArea](#). The User Reserved Memory is one of four memory areas that are preserved during a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as USER\_RESERVED\_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

**Limits**

There is a lower limit of 1024 and an upper limit of [UINT\\_MAX](#) (4 Gigabytes) on this configuration parameter.

Definition at line 329 of file default\_cfe\_es\_internal\_cfg.h.

**12.79.2.81 CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE** #define CFE\_PLATFORM\_ES\_VOLATILE\_STARTUPFILE "/ram/cfe\_es\_startup.scr"

**Purpose** ES Volatile Startup Filename

**Description:**

The value of this constant defines the path and name of the file that contains a list of modules that will be loaded and started by the cFE after the cFE finishes its startup sequence.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 376 of file default\_cfe\_es\_internal\_cfg.h.

## 12.80 cfe/modules/es/config/default\_cfe\_es\_mission\_cfg.h File Reference

```
#include "cfe_es_interface_cfg.h"
```

### 12.80.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.81 cfe/modules/es/config/default\_cfe\_es\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_es_msgdefs.h"
#include "cfe_es_msgstruct.h"
```

### 12.81.1 Detailed Description

Specification for the CFE Executive Services (CFE\_ES) command and telemetry message data types.  
This is a compatibility header for the "cfe\_es\_msg.h" file that has traditionally provided the message definitions for cFS apps.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.82 cfe/modules/es/config/default\_cfe\_es\_msgdefs.h File Reference

```
#include "cfe_es_fcncodes.h"
```

### 12.82.1 Detailed Description

Specification for the CFE Executive Services (CFE\_ES) command and telemetry message constant definitions.  
For CFE\_ES this is only the function/command code definitions

## 12.83 cfe/modules/es/config/default\_cfe\_es\_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_es_topicids.h"
```

### Macros

- #define CFE\_ES\_CMD\_MID CFE\_PLATFORM\_CMD\_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\_ES\_HK\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_ES\_HK\_TLM\_MSG /\* 0x0800 \*/
- #define CFE\_ES\_APP\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_ES\_APP\_TLM\_MSG /\* 0x080B \*/
- #define CFE\_ES\_MEMSTATS\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_ES\_MEMSTATS\_TLM\_MSG /\* 0x0810 \*/

### 12.83.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Message IDs

### 12.83.2 Macro Definition Documentation

#### 12.83.2.1 CFE\_ES\_APP\_TLM\_MID #define CFE\_ES\_APP\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_ES\_APP\_TLM\_MS

/\* 0x080B \*/

Definition at line 39 of file default\_cfe\_es\_msgids.h.

**12.83.2.2 CFE\_ES\_CMD\_MID** #define CFE\_ES\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_ES\_CMD\_MSG  
/\* 0x1806 \*/  
Definition at line 32 of file default\_cfe\_es\_msgids.h.

**12.83.2.3 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 38 of file default\_cfe\_es\_msgids.h.

**12.83.2.4 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 40 of file default\_cfe\_es\_msgids.h.

**12.83.2.5 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 33 of file default\_cfe\_es\_msgids.h.

## 12.84 cfe/modules/es/config/default\_cfe\_es\_msgstruct.h File Reference

```
#include "cfe_es_msgdefs.h"
#include "cfe_msg_hdr.h"
#include "cfe_mission_cfg.h"
```

### Data Structures

- struct **CFE\_ES\_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](#)

## 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 CFE_ES_NoArgsCmd_t CFE_ES_SendHkCmd_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**

### 12.84.1 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:

### 12.84.2 Typedef Documentation

**12.84.2.1 CFE\_ES\_AppNameCmd\_Payload\_t** `typedef struct CFE_ES_AppNameCmd_Payload CFE_ES_AppNameCmd_Payload_t`  
Generic application name command payload.

For command details, see [CFE\\_ES\\_STOP\\_APP\\_CC](#), [CFE\\_ES\\_RESTART\\_APP\\_CC](#), [CFE\\_ES\\_QUERY\\_ONE\\_CC](#)

**12.84.2.2 CFE\_ES\_AppNameCmd\_t** `typedef struct CFE_ES_AppNameCmd CFE_ES_AppNameCmd_t`  
Generic application name command.

**12.84.2.3 CFE\_ES\_AppReloadCmd\_Payload\_t** `typedef struct CFE_ES_AppReloadCmd_Payload CFE_ES_AppReloadCmd_Payload`  
Reload Application Command Payload.

For command details, see [CFE\\_ES\\_RELOAD\\_APP\\_CC](#)

**12.84.2.4 CFE\_ES\_ClearERLogCmd\_t** `typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearERLogCmd_t`  
Definition at line 69 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.5 CFE\_ES\_ClearSysLogCmd\_t** `typedef CFE_ES_NoArgsCmd_t CFE_ES_ClearSysLogCmd_t`  
Definition at line 68 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.6 CFE\_ES\_DeleteCDSCmd\_Payload\_t** `typedef struct CFE_ES_DeleteCDSCmd_Payload CFE_ES_DeleteCDSCmd_Payload`  
Delete Critical Data Store Command Payload.

For command details, see [CFE\\_ES\\_DELETE\\_CDS\\_CC](#)

**12.84.2.7 CFE\_ES\_DeleteCDSCmd\_t** `typedef struct CFE_ES_DeleteCDSCmd CFE_ES_DeleteCDSCmd_t`  
Delete Critical Data Store Command.

**12.84.2.8 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](#)

**12.84.2.9 CFE\_ES\_DumpCDSRegistryCmd\_t** `typedef struct CFE_ES_DumpCDSRegistryCmd CFE_ES_DumpCDSRegistryCmd_t`  
Dump CDS Registry Command.

**12.84.2.10 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](#)

**12.84.2.11 CFE\_ES\_FileNameCmd\_t** `typedef struct CFE_ES_FileNameCmd CFE_ES_FileNameCmd_t`  
Generic file name command.

**12.84.2.12 CFE\_ES\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_ES_HousekeepingTlm_Payload CFE_ES_HousekeepingTlm_Payload_t`

**Name** Executive Services Housekeeping Packet

**12.84.2.13 CFE\_ES\_HousekeepingTlm\_t** `typedef struct CFE_ES_HousekeepingTlm CFE_ES_HousekeepingTlm_t`

**12.84.2.14 CFE\_ES\_MemStatsTlm\_t** `typedef struct CFE_ES_MemStatsTlm CFE_ES_MemStatsTlm_t`

**12.84.2.15 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](#))

**12.84.2.16 CFE\_ES\_NoopCmd\_t** `typedef CFE_ES_NoArgsCmd_t CFE_ES_NoopCmd_t`  
Definition at line 66 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.17 CFE\_ES\_OneAppTlm\_Payload\_t** `typedef struct CFE_ES_OneAppTlm_Payload CFE_ES_OneAppTlm_Payload_t`

**Name** Single Application Information Packet

**12.84.2.18 CFE\_ES\_OneAppTlm\_t** `typedef struct CFE_ES_OneAppTlm CFE_ES_OneAppTlm_t`

**12.84.2.19 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](#)

**12.84.2.20 CFE\_ES\_OverWriteSysLogCmd\_t** `typedef struct CFE_ES_OverWriteSysLogCmd CFE_ES_OverWriteSysLogCmd_t`  
Overwrite/Discard System Log Configuration Command Payload.

**12.84.2.21 CFE\_ES\_PoolStatsTlm\_Payload\_t** `typedef struct CFE_ES_PoolStatsTlm_Payload CFE_ES_PoolStatsTlm_Payload_t`

**Name** Memory Pool Statistics Packet

**12.84.2.22 CFE\_ES\_QueryAllCmd\_t** `typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllCmd_t`  
Definition at line 121 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.23 CFE\_ES\_QueryAllTasksCmd\_t** `typedef CFE_ES_FileNameCmd_t CFE_ES_QueryAllTasksCmd_t`  
Definition at line 122 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.24 CFE\_ES\_QueryOneCmd\_t** `typedef CFE_ES_AppNameCmd_t CFE_ES_QueryOneCmd_t`  
Definition at line 205 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.25 CFE\_ES\_ReloadAppCmd\_t** `typedef struct CFE_ES_ReloadAppCmd CFE_ES_ReloadAppCmd_t`  
Reload Application Command.

**12.84.2.26 CFE\_ES\_ResetCountersCmd\_t** `typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetCountersCmd_t`  
Definition at line 67 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.27 CFE\_ES\_ResetPRCountCmd\_t** `typedef CFE_ES_NoArgsCmd_t CFE_ES_ResetPRCountCmd_t`  
Definition at line 70 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.28 CFE\_ES\_RestartAppCmd\_t** `typedef CFE_ES_AppNameCmd_t CFE_ES_RestartAppCmd_t`  
Definition at line 204 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.29 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](#)

**12.84.2.30 CFE\_ES\_RestartCmd\_t** `typedef struct CFE_ES_RestartCmd CFE_ES_RestartCmd_t`  
Restart cFE Command.

**12.84.2.31 CFE\_ES\_SendHkCmd\_t** `typedef CFE_ES_NoArgsCmd_t CFE_ES_SendHkCmd_t`  
Definition at line 71 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.32 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](#)

**12.84.2.33 CFE\_ES\_SendMemPoolStatsCmd\_t** `typedef struct CFE_ES_SendMemPoolStatsCmd CFE_ES_SendMemPoolStatsCmd_t`  
Send Memory Pool Statistics Command.

**12.84.2.34 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](#)

**12.84.2.35 CFE\_ES\_SetMaxPRCountCmd\_t** `typedef struct CFE_ES_SetMaxPRCountCmd CFE_ES_SetMaxPRCountCmd_t`  
Set Maximum Processor Reset Count Command.

**12.84.2.36 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](#)

**12.84.2.37 CFE\_ES\_SetPerfFilterMaskCmd\_t** `typedef struct CFE_ES_SetPerfFilterMaskCmd CFE_ES_SetPerfFilterMaskCmd_t`  
Set Performance Analyzer Filter Mask Command.

**12.84.2.38 CFE\_ES\_SetPerfTriggerMaskCmd\_t** `typedef struct CFE_ES_SetPerfTriggerMaskCmd CFE_ES_SetPerfTriggerMask`  
Set Performance Analyzer Trigger Mask Command.

**12.84.2.39 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](#)

**12.84.2.40 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](#)

**12.84.2.41 CFE\_ES\_StartAppCmd\_t** `typedef struct CFE_ES_StartApp CFE_ES_StartAppCmd_t`  
Start Application Command.

**12.84.2.42 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](#)

**12.84.2.43 CFE\_ES\_StartPerfDataCmd\_t** `typedef struct CFE_ES_StartPerfDataCmd CFE_ES_StartPerfDataCmd_t`  
Start Performance Analyzer Command.

**12.84.2.44 CFE\_ES\_StopAppCmd\_t** `typedef CFE_ES_AppNameCmd_t CFE_ES_StopAppCmd_t`  
Definition at line 203 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.45 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](#)

**12.84.2.46 CFE\_ES\_StopPerfDataCmd\_t** `typedef struct CFE_ES_StopPerfDataCmd CFE_ES_StopPerfDataCmd_t`  
Stop Performance Analyzer Command.

**12.84.2.47 CFE\_ES\_WriteERLogCmd\_t** `typedef CFE_ES_FileNameCmd_t CFE_ES_WriteERLogCmd_t`  
Definition at line 124 of file default\_cfe\_es\_msgstruct.h.

**12.84.2.48 CFE\_ES\_WriteSysLogCmd\_t** `typedef CFE_ES_FileNameCmd_t CFE_ES_WriteSysLogCmd_t`  
Definition at line 123 of file default\_cfe\_es\_msgstruct.h.

## 12.85 cfe/modules/es/config/default\_cfe\_es\_platform\_cfg.h File Reference

```
#include "cfe_es_mission_cfg.h"
#include "cfe_es_internal_cfg.h"
```

### 12.85.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Platform Configuration Header File

This is a compatibility header for the "platform\_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.86 cfe/modules/es/config/default\_cfe\_es\_topicids.h File Reference

### Macros

- #define CFE\_MISSION\_ES\_CMD\_MSG 6
- #define CFE\_MISSION\_ES\_SEND\_HK\_MSG 8
- #define CFE\_MISSION\_ES\_HK\_TLM\_MSG 0
- #define CFE\_MISSION\_ES\_APP\_TLM\_MSG 11
- #define CFE\_MISSION\_ES\_MEMSTATS\_TLM\_MSG 16

### 12.86.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Topic IDs

### 12.86.2 Macro Definition Documentation

**12.86.2.1 CFE\_MISSION\_ES\_APP\_TLM\_MSG** #define CFE\_MISSION\_ES\_APP\_TLM\_MSG 11  
Definition at line 48 of file default\_cfe\_es\_topicids.h.

**12.86.2.2 CFE\_MISSION\_ES\_CMD\_MSG** #define CFE\_MISSION\_ES\_CMD\_MSG 6

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE ES command messages

**Limits**

Not Applicable

Definition at line 35 of file default\_cfe\_es\_topicids.h.

**12.86.2.3 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 ES telemetry messages

**Limits**

Not Applicable

Definition at line 47 of file default\_cfe\_es\_topicids.h.

**12.86.2.4 CFE\_MISSION\_ES\_MEMSTATS\_TLM\_MSG** #define CFE\_MISSION\_ES\_MEMSTATS\_TLM\_MSG 16  
Definition at line 49 of file default\_cfe\_es\_topicids.h.

**12.86.2.5 CFE\_MISSION\_ES\_SEND\_HK\_MSG** #define CFE\_MISSION\_ES\_SEND\_HK\_MSG 8  
Definition at line 36 of file default\_cfe\_es\_topicids.h.

## 12.87 cfe/modules/es/fsw/inc/cfe\_es\_eventids.h File Reference

### Macros

#### ES event IDs

- #define **CFE\_ES\_INIT\_INF\_EID** 1  
*ES Initialization Event ID.*
- #define **CFE\_ES\_INITSTATS\_INF\_EID** 2  
*ES Initialization Statistics Information Event ID.*
- #define **CFE\_ES\_NOOP\_INF\_EID** 3  
*ES No-op Command Success Event ID.*
- #define **CFE\_ES\_RESET\_INF\_EID** 4  
*ES Reset Counters Command Success Event ID.*
- #define **CFE\_ES\_START\_INF\_EID** 6  
*ES Start Application Command Success Event ID.*
- #define **CFE\_ES\_STOP\_DBG\_EID** 7  
*ES Stop Application Command Request Success Event ID.*
- #define **CFE\_ES\_STOP\_INF\_EID** 8  
*ES Stop Application Completed Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_DBG\_EID** 9  
*ES Restart Application Command Request Success Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_INF\_EID** 10  
*ES Restart Application Completed Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_DBG\_EID** 11  
*ES Reload Application Command Request Success Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_INF\_EID** 12  
*ES Reload Application Complete Event ID.*
- #define **CFE\_ES\_EXIT\_APP\_INF\_EID** 13  
*ES Nominal Exit Application Complete Event ID.*
- #define **CFE\_ES\_ERREXIT\_APP\_INF\_EID** 14  
*ES Error Exit Application Complete Event ID.*
- #define **CFE\_ES\_ONE\_APP\_EID** 15  
*ES Query One Application Command Success Event ID.*
- #define **CFE\_ES\_ALL\_APPS\_EID** 16  
*ES Query All Applications Command Success Event ID.*
- #define **CFE\_ES\_SYSLOG1\_INF\_EID** 17  
*ES Clear System Log Command Success Event ID.*
- #define **CFE\_ES\_SYSLOG2\_EID** 18  
*ES Write System Log Command Success Event ID.*
- #define **CFE\_ES\_ERLOG1\_INF\_EID** 19  
*ES Clear Exception Reset Log Command Success Event ID.*

- #define `CFE_ES_ERLOG2_EID` 20  
*ES Write Exception Reset Log Complete Event ID.*
- #define `CFE_ES_MID_ERR_EID` 21  
*ES Invalid Message ID Received Event ID.*
- #define `CFE_ES_CC1_ERR_EID` 22  
*ES Invalid Command Code Received Event ID.*
- #define `CFE_ES_LEN_ERR_EID` 23  
*ES Invalid Command Length Event ID.*
- #define `CFE_ES_BOOT_ERR_EID` 24  
*ES Restart Command Invalid Restart Type Event ID.*
- #define `CFE_ES_START_ERR_EID` 26  
*ES Start Application Command Application Creation Failed Event ID.*
- #define `CFE_ES_START_INVALID_FILENAME_ERR_EID` 27  
*ES Start Application Command Invalid Filename Event ID.*
- #define `CFE_ES_START_INVALID_ENTRY_POINT_ERR_EID` 28  
*ES Start Application Command Entry Point NULL Event ID.*
- #define `CFE_ES_START_NULL_APP_NAME_ERR_EID` 29  
*ES Start Application Command App Name NULL Event ID.*
- #define `CFE_ES_START_PRIORITY_ERR_EID` 31  
*ES Start Application Command Priority Too Large Event ID.*
- #define `CFE_ES_START_EXC_ACTION_ERR_EID` 32  
*ES Start Application Command Exception Action Invalid Event ID.*
- #define `CFE_ES_ERREXIT_APP_ERR_EID` 33  
*ES Error Exit Application Cleanup Failed Event ID.*
- #define `CFE_ES_STOP_ERR1_EID` 35  
*ES Stop Application Command Request Failed Event ID.*
- #define `CFE_ES_STOP_ERR2_EID` 36  
*ES Stop Application Command Get ApplID 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 ApplID 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 ApplID 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

- #define `CFE_ES_OSCREATE_ERR_EID` 51  
*ES Query One Application Data Command Get AppID By Name Failed Event ID.*
- #define `CFE_ES_WRHDR_ERR_EID` 52  
*ES Query All Application Data Command File Creation Failed Event ID.*
- #define `CFE_ES_TASKWR_ERR_EID` 53  
*ES Query All Application Data Command File Write Header 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.*

### 12.87.1 Detailed Description

cFE Executive Services Event IDs

### 12.87.2 Macro Definition Documentation

**12.87.2.1 CFE\_ES\_ALL\_APPS\_EID** #define CFE\_ES\_ALL\_APPS\_EID 16  
ES Query All Applications Command Success Event ID.

Type: DEBUG

Cause:

[ES Query All Applications Command](#) success.  
Definition at line 206 of file cfe\_es\_eventids.h.

**12.87.2.2 CFE\_ES\_BOOT\_ERR\_EID** #define CFE\_ES\_BOOT\_ERR\_EID 24  
ES Restart Command Invalid Restart Type Event ID.

Type: ERROR

Cause:

[ES cFE Restart Command](#) failure due to invalid restart type.

Definition at line 294 of file cfe\_es\_eventids.h.

**12.87.2.3 CFE\_ES\_BUILD\_INF\_EID** #define CFE\_ES\_BUILD\_INF\_EID 92  
cFS Build Information Event ID

Type: INFORMATION

Cause:

ES Initialization complete and response to [ES NO-OP Command](#).

The Build field identifies the build date, time, hostname and user identifier of the build host machine for the current running binary. The first string is the build date/time, and the second string is formatted as "user@hostname"

This additionally reports the configuration name that was selected by the user, which may affect various platform/mission limits.

By default, if not specified/overridden, the default values of these variables will be: BUILDDATE ==> the output of "date +%Y%m%d%H%M" HOSTNAME ==> the output of "hostname" USER ==> the output of "whoami"

The values can be overridden by setting an environment variable with the names above to the value desired for the field when running "make".

Definition at line 1047 of file cfe\_es\_eventids.h.

**12.87.2.4 CFE\_ES\_CC1\_ERR\_EID** #define CFE\_ES\_CC1\_ERR\_EID 22  
ES Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE\\_ES\\_CMD\\_MID](#) received on the ES message pipe.

Definition at line 272 of file cfe\_es\_eventids.h.

**12.87.2.5 CFE\_ES\_CDS\_DELETE\_ERR\_EID** #define CFE\_ES\_CDS\_DELETE\_ERR\_EID 76  
ES Delete CDS Command Delete Failed Event ID.

Type: ERROR

Cause:

[ES Delete CDS Command](#) failed while deleting, see reported status code or system log for details.

Definition at line 834 of file cfe\_es\_eventids.h.

**12.87.2.6 CFE\_ES\_CDS\_DELETE\_TBL\_ERR\_EID** #define CFE\_ES\_CDS\_DELETE\_TBL\_ERR\_EID 79  
ES Delete CDS Command For Critical Table Event ID.

Type: ERROR

Cause:

[Delete CDS Command](#) failure due to the specified CDS name being a critical table. Critical Table images can only be deleted via a Table Services command, [CFE\\_TBL\\_DELETE\\_CDS\\_CC](#).

Definition at line 871 of file cfe\_es\_eventids.h.

**12.87.2.7 CFE\_ES\_CDS\_DELETED\_INFO\_EID** #define CFE\_ES\_CDS\_DELETED\_INFO\_EID 78  
ES Delete CDS Command Success Event ID.

Type: INFORMATION

Cause:

[ES Delete CDS Command](#) success.

Definition at line 857 of file cfe\_es\_eventids.h.

**12.87.2.8 CFE\_ES\_CDS\_DUMP\_ERR\_EID** #define CFE\_ES\_CDS\_DUMP\_ERR\_EID 84  
ES Write Critical Data Store Registry Command Record Write Failed Event ID.

Type: ERROR

Cause:

[ES Write Critical Data Store Registry Command](#) failed to write CDS record.

Definition at line 929 of file cfe\_es\_eventids.h.

**12.87.2.9 CFE\_ES\_CDS\_NAME\_ERR\_EID** #define CFE\_ES\_CDS\_NAME\_ERR\_EID 77  
ES Delete CDS Command Lookup CDS Failed Event ID.

Type: ERROR

Cause:

[ES Delete CDS Command](#) failed due to the specified CDS name not found in the CDS Registry.

Definition at line 846 of file cfe\_es\_eventids.h.

**12.87.2.10 CFE\_ES\_CDS\_OWNER\_ACTIVE\_EID** #define CFE\_ES\_CDS\_OWNER\_ACTIVE\_EID 80  
ES Delete CDS Command With Active Owner Event ID.

Type: ERROR

Cause:

[ES Delete CDS Command](#) failure due to the specifies CDS name is registered to an active application.  
Definition at line 883 of file cfe\_es\_eventids.h.

**12.87.2.11 CFE\_ES\_CDS\_REG\_DUMP\_INF\_EID** #define CFE\_ES\_CDS\_REG\_DUMP\_INF\_EID 83  
ES Write Critical Data Store Registry Command Success Event ID.

Type: DEBUG

Cause:

[ES Write Critical Data Store Registry Command](#) success.  
Definition at line 917 of file cfe\_es\_eventids.h.

**12.87.2.12 CFE\_ES\_CDS\_REGISTER\_ERR\_EID** #define CFE\_ES\_CDS\_REGISTER\_ERR\_EID 69  
ES Register CDS API Failed Event ID.

Type: ERROR

Cause:

[CFE\\_ES\\_RegisterCDS](#) API failure, see reported status code or system log for details.  
Definition at line 766 of file cfe\_es\_eventids.h.

**12.87.2.13 CFE\_ES\_CREATING\_CDS\_DUMP\_ERR\_EID** #define CFE\_ES\_CREATING\_CDS\_DUMP\_ERR\_EID 86  
ES Write Critical Data Store Registry Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[ES Write Critical Data Store Registry Command](#) failed to parse filename or open/create the file. OVERLOADED  
Definition at line 953 of file cfe\_es\_eventids.h.

**12.87.2.14 CFE\_ES\_ERLOG1\_INF\_EID** #define CFE\_ES\_ERLOG1\_INF\_EID 19  
ES Clear Exception Reset Log Command Success Event ID.

Type: INFORMATION

Cause:

[ES Clear Exception Reset Log Command](#) success.  
Definition at line 239 of file cfe\_es\_eventids.h.

**12.87.2.15 CFE\_ES\_ERLOG2\_EID** #define CFE\_ES\_ERLOG2\_EID 20  
ES Write Exception Reset Log Complete Event ID.

Type: DEBUG

Cause:

Request to write the Exception Reset log successfully completed.  
Definition at line 250 of file cfe\_es\_eventids.h.

**12.87.2.16 CFE\_ES\_ERLOG2\_ERR\_EID** #define CFE\_ES\_ERLOG2\_ERR\_EID 56  
ES Write Exception Reset Log Command Request or File Creation Failed Event ID.

Type: ERROR

Cause:

[ES Write Exception Reset Log Command](#) request failed or file creation failed. OVERLOADED  
Definition at line 626 of file cfe\_es\_eventids.h.

**12.87.2.17 CFE\_ES\_ERLOG\_PENDING\_ERR\_EID** #define CFE\_ES\_ERLOG\_PENDING\_ERR\_EID 93  
ES Write Exception Reset Log Command Already In Progress Event ID.

Type: ERROR

Cause:

[ES Write Exception Reset Log Command](#) failure due to a write already being in progress.  
Definition at line 1059 of file cfe\_es\_eventids.h.

**12.87.2.18 CFE\_ES\_ERR\_SYSLOGMODE\_EID** #define CFE\_ES\_ERR\_SYSLOGMODE\_EID 71  
ES Set System Log Overwrite Mode Command Failed Event ID.

Type: ERROR

Cause:

[ES Set System Log Overwrite Mode Command](#) failed due to invalid mode requested.  
Definition at line 789 of file cfe\_es\_eventids.h.

**12.87.2.19 CFE\_ES\_ERREXIT\_APP\_ERR\_EID** #define CFE\_ES\_ERREXIT\_APP\_ERR\_EID 33  
ES Error Exit Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Error request to exit an application failed during application cleanup. Application and related resources will be in undefined state.  
Definition at line 379 of file cfe\_es\_eventids.h.

**12.87.2.20 CFE\_ES\_ERREXIT\_APP\_INF\_EID** #define CFE\_ES\_ERREXIT\_APP\_INF\_EID 14  
ES Error Exit Application Complete Event ID.

Type: INFORMATION

Cause:

Error request to exit an application successfully completed. This event indicates the Application exited due to an error condition. The details of the error that occurred should be given by the Application through an event message, System Log entry, or both.

Definition at line 184 of file cfe\_es\_eventids.h.

**12.87.2.21 CFE\_ES\_EXIT\_APP\_ERR\_EID** #define CFE\_ES\_EXIT\_APP\_ERR\_EID 46  
ES Exit Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Nominal request to exit an application failed during application cleanup. Application and related resources will be in undefined state.  
Definition at line 522 of file cfe\_es\_eventids.h.

**12.87.2.22 CFE\_ES\_EXIT\_APP\_INF\_EID** #define CFE\_ES\_EXIT\_APP\_INF\_EID 13  
ES Nominal Exit Application Complete Event ID.

Type: INFORMATION

Cause:

Nominal request to exit an application successfully completed. This event indicates the Application exited due to a nominal exit condition.

Definition at line 170 of file cfe\_es\_eventids.h.

**12.87.2.23 CFE\_ES\_FILEWRITE\_ERR\_EID** #define CFE\_ES\_FILEWRITE\_ERR\_EID 74  
ES File Write Failed Event ID.

Type: ERROR

Cause:

ES File Write failure writing data to file. OVERLOADED

Definition at line 822 of file cfe\_es\_eventids.h.

**12.87.2.24 CFE\_ES\_INIT\_INF\_EID** #define CFE\_ES\_INIT\_INF\_EID 1  
ES Initialization Event ID.

Type: INFORMATION

Cause:

Executive Services Task initialization complete.

Definition at line 42 of file cfe\_es\_eventids.h.

**12.87.2.25 CFE\_ES\_INITSTATS\_INF\_EID** #define CFE\_ES\_INITSTATS\_INF\_EID 2  
ES Initialization Statistics Information Event ID.

Type: INFORMATION

Cause:

Executive Services Task initialization complete.

Definition at line 53 of file cfe\_es\_eventids.h.

**12.87.2.26 CFE\_ES\_INVALID\_POOL\_HANDLE\_ERR\_EID** #define CFE\_ES\_INVALID\_POOL\_HANDLE\_ERR\_EID 82  
ES Telemeter Memory Statistics Command Invalid Handle Event ID.

Type: ERROR

Cause:

[ES Telemeter Memory Statistics Command](#) failure due to an invalid memory handle.  
Definition at line 906 of file cfe\_es\_eventids.h.

**12.87.2.27 CFE\_ES\_LEN\_ERR\_EID** #define CFE\_ES\_LEN\_ERR\_EID 23  
ES Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE\\_ES\\_CMD\\_MID](#) received on the ES message pipe.  
Definition at line 283 of file cfe\_es\_eventids.h.

**12.87.2.28 CFE\_ES\_MID\_ERR\_EID** #define CFE\_ES\_MID\_ERR\_EID 21  
ES Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the ES message pipe.  
Definition at line 261 of file cfe\_es\_eventids.h.

**12.87.2.29 CFE\_ES\_NOOP\_INF\_EID** #define CFE\_ES\_NOOP\_INF\_EID 3  
ES No-op Command Success Event ID.

Type: INFORMATION

Cause:

[ES No-op Command](#) success.  
Definition at line 64 of file cfe\_es\_eventids.h.

**12.87.2.30 CFE\_ES\_ONE\_APP\_EID** #define CFE\_ES\_ONE\_APP\_EID 15  
ES Query One Application Command Success Event ID.

Type: DEBUG

Cause:

[ES Query One Application Command](#) success.  
Definition at line 195 of file cfe\_es\_eventids.h.

**12.87.2.31 CFE\_ES\_ONE\_APPID\_ERR\_EID** #define CFE\_ES\_ONE\_APPID\_ERR\_EID 50  
ES Query One Application Data Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Query One Application Data Command](#) failed to get application ID from application name. Message will not be sent.  
Definition at line 569 of file cfe\_es\_eventids.h.

**12.87.2.32 CFE\_ES\_ONE\_ERR\_EID** #define CFE\_ES\_ONE\_ERR\_EID 49  
ES Query One Application Data Command Transmit Message Failed Event ID.

Type: ERROR

Cause:

[ES Query One Application Data Command](#) failed during message transmission.  
Definition at line 557 of file cfe\_es\_eventids.h.

**12.87.2.33 CFE\_ES\_OSCREATE\_ERR\_EID** #define CFE\_ES\_OSCREATE\_ERR\_EID 51  
ES Query All Application Data Command File Creation Failed Event ID.

Type: ERROR

Cause:

[ES Query All Application Data Command](#) failed to create file.  
Definition at line 580 of file cfe\_es\_eventids.h.

**12.87.2.34 CFE\_ES\_PCR\_ERR1\_EID** #define CFE\_ES\_PCR\_ERR1\_EID 47  
ES Process Control Invalid Exception State Event ID.

Type: ERROR

Cause:

Invalid Exception state encountered when processing requests for application state changes. Exceptions are processed immediately, so this state should never occur during routine processing.

Definition at line 534 of file cfe\_es\_eventids.h.

**12.87.2.35 CFE\_ES\_PCR\_ERR2\_EID** #define CFE\_ES\_PCR\_ERR2\_EID 48  
ES Process Control Unknown State Event ID.

Type: ERROR

Cause:

Unknown state encountered when processing requests for application state changes.

Definition at line 545 of file cfe\_es\_eventids.h.

**12.87.2.36 CFE\_ES\_PERF\_DATAWRITTEN\_EID** #define CFE\_ES\_PERF\_DATAWRITTEN\_EID 68  
Performance Log Write Success Event ID.

Type: DEBUG

Cause:

Request to write the performance log successfully completed.

Definition at line 755 of file cfe\_es\_eventids.h.

**12.87.2.37 CFE\_ES\_PERF\_FILTMSKCMD\_EID** #define CFE\_ES\_PERF\_FILTMSKCMD\_EID 63  
ES Set Performance Analyzer Filter Mask Command Success Event ID.

Type: DEBUG

Cause:

[ES Set Performance Analyzer Filter Mask Command](#) success.

Definition at line 697 of file cfe\_es\_eventids.h.

**12.87.2.38 CFE\_ES\_PERF\_FILTMSKERR\_EID** #define CFE\_ES\_PERF\_FILTMSKERR\_EID 64  
ES Set Performance Analyzer Filter Mask Command Invalid Index Event ID.

Type: ERROR

Cause:

[ES Set Performance Analyzer Filter Mask Command](#) failed filter index range check.  
Definition at line 709 of file cfe\_es\_eventids.h.

**12.87.2.39 CFE\_ES\_PERF\_LOG\_ERR\_EID** #define CFE\_ES\_PERF\_LOG\_ERR\_EID 67  
ES Stop Performance Analyzer Data Collection Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[ES Stop Performance Analyzer Data Collection Command](#) failed either parsing the file name or during open/creation of the file. OVERLOADED  
Definition at line 744 of file cfe\_es\_eventids.h.

**12.87.2.40 CFE\_ES\_PERF\_STARTCMD\_EID** #define CFE\_ES\_PERF\_STARTCMD\_EID 57  
ES Start Performance Analyzer Data Collection Command Success Event ID.

Type: DEBUG

Cause:

[ES Start Performance Analyzer Data Collection Command](#) success.  
Definition at line 637 of file cfe\_es\_eventids.h.

**12.87.2.41 CFE\_ES\_PERF\_STARTCMD\_ERR\_EID** #define CFE\_ES\_PERF\_STARTCMD\_ERR\_EID 58  
ES Start Performance Analyzer Data Collection Command Idle Check Failed Event ID.

Type: ERROR

Cause:

[ES Start Performance Analyzer Data Collection Command](#) failed due to already being started.  
Definition at line 649 of file cfe\_es\_eventids.h.

**12.87.2.42 CFE\_ES\_PERF\_STARTCMD\_TRIG\_ERR\_EID** #define CFE\_ES\_PERF\_STARTCMD\_TRIG\_ERR\_EID 59  
ES Start Performance Analyzer Data Collection Command Invalid Trigger Event ID.

Type: ERROR

Cause:

[ES Start Performance Analyzer Data Collection Command](#) failed due to invalid trigger mode.  
Definition at line 661 of file cfe\_es\_eventids.h.

**12.87.2.43 CFE\_ES\_PERF\_STOPCMD\_EID** #define CFE\_ES\_PERF\_STOPCMD\_EID 60  
ES Stop Performance Analyzer Data Collection Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Stop Performance Analyzer Data Collection Command](#) success. Note this event signifies the request to stop and write the performance data has been successfully submitted. The successful completion will generate a [CFE\\_ES\\_PERF\\_DATAWRITTEN\\_EID](#) event.  
Definition at line 674 of file cfe\_es\_eventids.h.

**12.87.2.44 CFE\_ES\_PERF\_STOPCMD\_ERR2\_EID** #define CFE\_ES\_PERF\_STOPCMD\_ERR2\_EID 62  
ES Stop Performance Analyzer Data Collection Command Request Idle Check Failed Event ID.

Type: ERROR

Cause:

[ES Stop Performance Analyzer Data Collection Command](#) failed due to a write already in progress.  
Definition at line 686 of file cfe\_es\_eventids.h.

**12.87.2.45 CFE\_ES\_PERF\_TRIGMSKCMD\_EID** #define CFE\_ES\_PERF\_TRIGMSKCMD\_EID 65  
ES Set Performance Analyzer Trigger Mask Command Success Event ID.

Type: DEBUG

Cause:

[ES Set Performance Analyzer Trigger Mask Command](#) success.  
Definition at line 720 of file cfe\_es\_eventids.h.

**12.87.2.46 CFE\_ES\_PERF\_TRIGMSKERR\_EID** #define CFE\_ES\_PERF\_TRIGMSKERR\_EID 66  
ES Set Performance Analyzer Trigger Mask Command Invalid Mask Event ID.

Type: ERROR

Cause:

[ES Set Performance Analyzer Trigger Mask Command](#) failed the mask range check.  
Definition at line 732 of file cfe\_es\_eventids.h.

**12.87.2.47 CFE\_ES\_RELOAD\_APP\_DBG\_EID** #define CFE\_ES\_RELOAD\_APP\_DBG\_EID 11  
ES Reload Application Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Reload Application Command](#) success. Note this event signifies the request to reload the application has been successfully submitted. The successful completion will generate a [CFE\\_ES\\_RELOAD\\_APP\\_INF\\_EID](#) event.  
Definition at line 147 of file cfe\_es\_eventids.h.

**12.87.2.48 CFE\_ES\_RELOAD\_APP\_ERR1\_EID** #define CFE\_ES\_RELOAD\_APP\_ERR1\_EID 42  
ES Reload Application Command Request Failed Event ID.

Type: ERROR

Cause:

[ES Reload Application Command](#) request failed.  
Definition at line 473 of file cfe\_es\_eventids.h.

**12.87.2.49 CFE\_ES\_RELOAD\_APP\_ERR2\_EID** #define CFE\_ES\_RELOAD\_APP\_ERR2\_EID 43  
ES Reload Application Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Reload Application Command](#) failed to get application ID from application name. The application will not be reloaded.  
Definition at line 485 of file cfe\_es\_eventids.h.

**12.87.2.50 CFE\_ES\_RELOAD\_APP\_ERR3\_EID** #define CFE\_ES\_RELOAD\_APP\_ERR3\_EID 44  
ES Reload Application Startup Failed Event ID.

Type: ERROR

Cause:

Request to reload an application failed during application startup. The application will not be reloaded.  
Definition at line 497 of file cfe\_es\_eventids.h.

**12.87.2.51 CFE\_ES\_RELOAD\_APP\_ERR4\_EID** #define CFE\_ES\_RELOAD\_APP\_ERR4\_EID 45  
ES Reload Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Request to reload an application failed during application cleanup. The application will not be reloaded and will be in an undefined state along with its associated resources.  
Definition at line 510 of file cfe\_es\_eventids.h.

**12.87.2.52 CFE\_ES\_RELOAD\_APP\_INF\_EID** #define CFE\_ES\_RELOAD\_APP\_INF\_EID 12  
ES Reload Application Complete Event ID.

Type: INFORMATION

Cause:

Request to reload an application successfully completed.  
Definition at line 158 of file cfe\_es\_eventids.h.

**12.87.2.53 CFE\_ES\_RESET\_INF\_EID** #define CFE\_ES\_RESET\_INF\_EID 4  
ES Reset Counters Command Success Event ID.

Type: INFORMATION

Cause:

[ES Reset Counters Command](#) success.  
Definition at line 75 of file cfe\_es\_eventids.h.

**12.87.2.54 CFE\_ES\_RESET\_PR\_COUNT\_EID** #define CFE\_ES\_RESET\_PR\_COUNT\_EID 72  
ES Set Processor Reset Counter to Zero Command Success Event ID.

Type: INFORMATION

Cause:

[ES Set Processor Reset Counter to Zero Command](#) success.  
Definition at line 800 of file cfe\_es\_eventids.h.

**12.87.2.55 CFE\_ES\_RESTART\_APP\_DBG\_EID** #define CFE\_ES\_RESTART\_APP\_DBG\_EID 9  
ES Restart Application Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Restart Application Command](#) success. Note this event signifies the request to restart the application has been successfully submitted. The successful completion will generate a [CFE\\_ES\\_RESTART\\_APP\\_INF\\_EID](#) event.  
Definition at line 123 of file cfe\_es\_eventids.h.

**12.87.2.56 CFE\_ES\_RESTART\_APP\_ERR1\_EID** #define CFE\_ES\_RESTART\_APP\_ERR1\_EID 38  
ES Restart Application Command Request Failed Event ID.

Type: ERROR

Cause:

[ES Restart Application Command](#) request failed.  
Definition at line 425 of file cfe\_es\_eventids.h.

**12.87.2.57 CFE\_ES\_RESTART\_APP\_ERR2\_EID** #define CFE\_ES\_RESTART\_APP\_ERR2\_EID 39  
ES Restart Application Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Restart Application Command](#) failed to get application ID from application name. The application will not be restarted.  
Definition at line 437 of file cfe\_es\_eventids.h.

**12.87.2.58 CFE\_ES\_RESTART\_APP\_ERR3\_EID** #define CFE\_ES\_RESTART\_APP\_ERR3\_EID 40  
ES Restart Application Startup Failed Event ID.

Type: ERROR

Cause:

Request to restart an application failed during application startup. The application will not be restarted.  
Definition at line 449 of file cfe\_es\_eventids.h.

**12.87.2.59 CFE\_ES\_RESTART\_APP\_ERR4\_EID** #define CFE\_ES\_RESTART\_APP\_ERR4\_EID 41  
ES Restart Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Request to restart an application failed during application cleanup. The application will not be restarted and will be in an undefined state along with its associated resources.  
Definition at line 462 of file cfe\_es\_eventids.h.

**12.87.2.60 CFE\_ES\_RESTART\_APP\_INF\_EID** #define CFE\_ES\_RESTART\_APP\_INF\_EID 10  
ES Restart Application Completed Event ID.

Type: INFORMATION

Cause:

Request to restart an application successfully completed.  
Definition at line 134 of file cfe\_es\_eventids.h.

**12.87.2.61 CFE\_ES\_SET\_MAX\_PR\_COUNT\_EID** #define CFE\_ES\_SET\_MAX\_PR\_COUNT\_EID 73  
ES Set Maximum Processor Reset Limit Command Success Event ID.

Type: INFORMATION

Cause:

[ES Set Maximum Processor Reset Limit Command](#) success.  
Definition at line 811 of file cfe\_es\_eventids.h.

**12.87.2.62 CFE\_ES\_START\_ERR\_EID** #define CFE\_ES\_START\_ERR\_EID 26  
ES Start Application Command Application Creation Failed Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure during application creation after successful parameter validation.  
Definition at line 306 of file cfe\_es\_eventids.h.

**12.87.2.63 CFE\_ES\_START\_EXC\_ACTION\_ERR\_EID** #define CFE\_ES\_START\_EXC\_ACTION\_ERR\_EID 32  
ES Start Application Command Exception Action Invalid Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to invalid application exception action.  
Definition at line 367 of file cfe\_es\_eventids.h.

**12.87.2.64 CFE\_ES\_START\_INF\_EID** #define CFE\_ES\_START\_INF\_EID 6  
ES Start Application Command Success Event ID.

Type: INFORMATION

Cause:

[ES Start Application Command](#) success.  
Definition at line 86 of file cfe\_es\_eventids.h.

**12.87.2.65 CFE\_ES\_START\_INVALID\_ENTRY\_POINT\_ERR\_EID** #define CFE\_ES\_START\_INVALID\_ENTRY\_POINT\_ERR\_EID 28  
ES Start Application Command Entry Point NULL Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to a NULL Application Entry Point.  
Definition at line 330 of file cfe\_es\_eventids.h.

**12.87.2.66 CFE\_ES\_START\_INVALID\_FILENAME\_ERR\_EID** #define CFE\_ES\_START\_INVALID\_FILENAME\_ERR\_EID 27

ES Start Application Command Invalid Filename Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to invalid filename.

Definition at line 318 of file cfe\_es\_eventids.h.

**12.87.2.67 CFE\_ES\_START\_NULL\_APP\_NAME\_ERR\_EID** #define CFE\_ES\_START\_NULL\_APP\_NAME\_ERR\_EID 29

ES Start Application Command App Name NULL Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to NULL Application Name.

Definition at line 342 of file cfe\_es\_eventids.h.

**12.87.2.68 CFE\_ES\_START\_PRIORITY\_ERR\_EID** #define CFE\_ES\_START\_PRIORITY\_ERR\_EID 31

ES Start Application Command Priority Too Large Event ID.

Type: ERROR

Cause:

[ES Start Application Command](#) failure due to a requested application priority greater than the maximum priority allowed for tasks as defined by the OS Abstraction Layer (OS\_MAX\_PRIORITY).

Definition at line 355 of file cfe\_es\_eventids.h.

**12.87.2.69 CFE\_ES\_STOP\_DBG\_EID** #define CFE\_ES\_STOP\_DBG\_EID 7

ES Stop Application Command Request Success Event ID.

Type: DEBUG

Cause:

[ES Stop Application Command](#) success. Note this event signifies the request to delete the application has been successfully submitted. The successful completion will generate a [CFE\\_ES\\_STOP\\_INF\\_EID](#) event.

Definition at line 99 of file cfe\_es\_eventids.h.

**12.87.2.70 CFE\_ES\_STOP\_ERR1\_EID** #define CFE\_ES\_STOP\_ERR1\_EID 35  
ES Stop Application Command Request Failed Event ID.

Type: ERROR

Cause:

[ES Stop Application Command](#) request failed.  
Definition at line 390 of file cfe\_es\_eventids.h.

**12.87.2.71 CFE\_ES\_STOP\_ERR2\_EID** #define CFE\_ES\_STOP\_ERR2\_EID 36  
ES Stop Application Command Get AppID By Name Failed Event ID.

Type: ERROR

Cause:

[ES Stop Application Command](#) failed to get application ID from application name. The application will not be deleted.  
Definition at line 402 of file cfe\_es\_eventids.h.

**12.87.2.72 CFE\_ES\_STOP\_ERR3\_EID** #define CFE\_ES\_STOP\_ERR3\_EID 37  
ES Stop Application Cleanup Failed Event ID.

Type: ERROR

Cause:

Request to delete an application failed during application cleanup. Application and related resources will be in undefined state.  
Definition at line 414 of file cfe\_es\_eventids.h.

**12.87.2.73 CFE\_ES\_STOP\_INF\_EID** #define CFE\_ES\_STOP\_INF\_EID 8  
ES Stop Application Completed Event ID.

Type: INFORMATION

Cause:

Request to delete an application successfully completed.  
Definition at line 110 of file cfe\_es\_eventids.h.

**12.87.2.74 CFE\_ES\_SYSLOG1\_INF\_EID** #define CFE\_ES\_SYSLOG1\_INF\_EID 17  
ES Clear System Log Command Success Event ID.

Type: INFORMATION

Cause:

[ES Clear System Log Command](#) success.  
Definition at line 217 of file cfe\_es\_eventids.h.

**12.87.2.75 CFE\_ES\_SYSLOG2\_EID** #define CFE\_ES\_SYSLOG2\_EID 18  
ES Write System Log Command Success Event ID.

Type: DEBUG

Cause:

[ES Write System Log Command](#) success.  
Definition at line 228 of file cfe\_es\_eventids.h.

**12.87.2.76 CFE\_ES\_SYSLOG2\_ERR\_EID** #define CFE\_ES\_SYSLOG2\_ERR\_EID 55  
ES Write System Log Command Filename Parse or File Creation Failed Event ID.

Type: ERROR

Cause:

[ES Write System Log Command](#) failed parsing file name or creating the file. OVERLOADED  
Definition at line 614 of file cfe\_es\_eventids.h.

**12.87.2.77 CFE\_ES\_SYSLOGMODE\_EID** #define CFE\_ES\_SYSLOGMODE\_EID 70  
ES Set System Log Overwrite Mode Command Success Event ID.

Type: DEBUG

Cause:

[ES Set System Log Overwrite Mode Command](#) success.  
Definition at line 777 of file cfe\_es\_eventids.h.

**12.87.2.78 CFE\_ES\_TASKINFO\_EID** #define CFE\_ES\_TASKINFO\_EID 87  
ES Write All Task Data Command Success Event ID.

Type: DEBUG

Cause:

[ES Write All Task Data Command](#) success.  
Definition at line 964 of file cfe\_es\_eventids.h.

**12.87.2.79 CFE\_ES\_TASKINFO\_OSCREATE\_ERR\_EID** #define CFE\_ES\_TASKINFO\_OSCREATE\_ERR\_EID 88  
ES Write All Task Data Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[ES Write All Task Data Command](#) failed to parse the filename or open/create the file.  
Definition at line 976 of file cfe\_es\_eventids.h.

**12.87.2.80 CFE\_ES\_TASKINFO\_WR\_ERR\_EID** #define CFE\_ES\_TASKINFO\_WR\_ERR\_EID 90  
ES Write All Task Data Command Write Data Failed Event ID.

Type: ERROR

Cause:

[ES Write All Task Data Command](#) failed to write task data to file.  
Definition at line 1000 of file cfe\_es\_eventids.h.

**12.87.2.81 CFE\_ES\_TASKINFO\_WRHDR\_ERR\_EID** #define CFE\_ES\_TASKINFO\_WRHDR\_ERR\_EID 89  
ES Write All Task Data Command Write Header Failed Event ID.

Type: ERROR

Cause:

[ES Write All Task Data Command](#) failed to write file header.  
Definition at line 988 of file cfe\_es\_eventids.h.

**12.87.2.82 CFE\_ES\_TASKWR\_ERR\_EID** #define CFE\_ES\_TASKWR\_ERR\_EID 53  
ES Query All Application Data Command File Write App Data Failed Event ID.

Type: ERROR

Cause:

[ES Query All Application Data Command](#) failed to write file application data.  
Definition at line 602 of file cfe\_es\_eventids.h.

**12.87.2.83 CFE\_ES\_TLM\_POOL\_STATS\_INFO\_EID** #define CFE\_ES\_TLM\_POOL\_STATS\_INFO\_EID 81  
ES Telemeter Memory Statistics Command Success Event ID.

Type: DEBUG

Cause:

[ES Telemeter Memory Statistics Command](#) success.  
Definition at line 894 of file cfe\_es\_eventids.h.

**12.87.2.84 CFE\_ES\_VERSION\_INF\_EID** #define CFE\_ES\_VERSION\_INF\_EID 91  
cFS Version Information Event ID

Type: INFORMATION

Cause:

ES Initialization complete and response to [ES NO-OP Command](#).  
A separate version info event will be generated for every module which is statically linked into the CFE core executable (e.g. OSAL, PSP, MSG, SBR, etc).  
The version information reported in this event is derived from the source revision control system at build time, as opposed to manually-assigned semantic version numbers. It is intended to uniquely identify the actual source code that is currently running, to the extent this is possible.  
The Mission version information also identifies the build configuration name, if available.  
Definition at line 1021 of file cfe\_es\_eventids.h.

**12.87.2.85 CFE\_ES\_WRHDR\_ERR\_EID** #define CFE\_ES\_WRHDR\_ERR\_EID 52  
ES Query All Application Data Command File Write Header Failed Event ID.

Type: ERROR

Cause:

[ES Query All Application Data Command](#) failed to write file header.  
Definition at line 591 of file cfe\_es\_eventids.h.

**12.87.2.86 CFE\_ES\_WRITE\_CFE\_HDR\_ERR\_EID** #define CFE\_ES\_WRITE\_CFE\_HDR\_ERR\_EID 85  
 ES Write Critical Data Store Registry Command Header Write Failed Event ID.

Type: ERROR

Cause:

[ES Write Critical Data Store Registry Command](#) failed to write header.  
 Definition at line 941 of file cfe\_es\_eventids.h.

## 12.88 cfe/modules/evs/config/default\_cfe\_evs\_extern\_typedefs.h File Reference

```
#include "common_types.h"
```

### Typedefs

- **typedef uint8 CFE\_EVS\_MsgFormat\_Enum\_t**  
*Identifies format of log messages.*
- **typedef uint8 CFE\_EVS\_LogMode\_Enum\_t**  
*Identifies handling of log messages after storage is filled.*
- **typedef uint16 CFE\_EVS\_EventType\_Enum\_t**  
*Identifies type of event message.*
- **typedef uint8 CFE\_EVS\_EventFilter\_Enum\_t**  
*Identifies event filter schemes.*
- **typedef uint8 CFE\_EVS\_EventOutput\_Enum\_t**  
*Identifies event output port.*

### Enumerations

- enum **CFE\_EVS\_MsgFormat** { **CFE\_EVS\_MsgFormat\_SHORT** = 0, **CFE\_EVS\_MsgFormat\_LONG** = 1 }  
*Label definitions associated with CFE\_EVS\_MsgFormat\_Enum\_t.*
- enum **CFE\_EVS\_LogMode** { **CFE\_EVS\_LogMode\_OVERWRITE** = 0, **CFE\_EVS\_LogMode\_DISCARD** = 1 }  
*Label definitions associated with CFE\_EVS\_LogMode\_Enum\_t.*
- enum **CFE\_EVS\_EventType** { **CFE\_EVS\_EventType\_DEBUG** = 1, **CFE\_EVS\_EventType\_INFORMATION** = 2, **CFE\_EVS\_EventType\_ERROR** = 3, **CFE\_EVS\_EventType\_CRITICAL** = 4 }  
*Label definitions associated with CFE\_EVS\_EventType\_Enum\_t.*
- enum **CFE\_EVS\_EventFilter** { **CFE\_EVS\_EventFilter\_BINARY** = 0 }  
*Label definitions associated with CFE\_EVS\_EventFilter\_Enum\_t.*
- enum **CFE\_EVS\_EventOutput** { **CFE\_EVS\_EventOutput\_PORT1** = 1, **CFE\_EVS\_EventOutput\_PORT2** = 2, **CFE\_EVS\_EventOutput\_PORT3** = 3, **CFE\_EVS\_EventOutput\_PORT4** = 4 }  
*Label definitions associated with CFE\_EVS\_EventOutput\_Enum\_t.*

### 12.88.1 Detailed Description

Declarations and prototypes for cfe\_evs\_extern\_typedefs module

### 12.88.2 Typedef Documentation

**12.88.2.1 CFE\_EVS\_EventFilter\_Enum\_t** `typedef uint8 CFE_EVS_EventFilter_Enum_t`  
Identifies event filter schemes.

See also

enum [CFE\\_EVS\\_EventFilter](#)

Definition at line 125 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.88.2.2 CFE\_EVS\_EventOutput\_Enum\_t** `typedef uint8 CFE_EVS_EventOutput_Enum_t`  
Identifies event output port.

See also

enum [CFE\\_EVS\\_EventOutput](#)

Definition at line 158 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.88.2.3 CFE\_EVS\_EventType\_Enum\_t** `typedef uint16 CFE_EVS_EventType_Enum_t`  
Identifies type of event message.

See also

enum [CFE\\_EVS\\_EventType](#)

Definition at line 107 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.88.2.4 CFE\_EVS\_LogMode\_Enum\_t** `typedef uint8 CFE_EVS_LogMode_Enum_t`  
Identifies handling of log messages after storage is filled.

See also

enum [CFE\\_EVS\\_LogMode](#)

Definition at line 74 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.88.2.5 CFE\_EVS\_MsgFormat\_Enum\_t** `typedef uint8 CFE_EVS_MsgFormat_Enum_t`  
Identifies format of log messages.

See also

enum [CFE\\_EVS\\_MsgFormat](#)

Definition at line 51 of file default\_cfe\_evs\_extern\_typedefs.h.

## 12.88.3 Enumeration Type Documentation

**12.88.3.1 CFE\_EVS\_EventFilter** `enum CFE_EVS_EventFilter`  
Label definitions associated with CFE\_EVS\_EventFilter\_Enum\_t.

Enumerator

<code>CFE_EVS_EventFilter_BINARY</code>	Binary event filter.
---	----------------------

Definition at line 112 of file default\_cfe\_evs\_extern\_typedefs.h.

#### **12.88.3.2 CFE\_EVS\_EventOutput enum [CFE\\_EVS\\_EventOutput](#)**

Label definitions associated with CFE\_EVS\_EventOutput\_Enum\_t.

Enumerator

CFE_EVS_EventOutput_PORT1	Output Port 1.
CFE_EVS_EventOutput_PORT2	Output Port 2.
CFE_EVS_EventOutput_PORT3	Output Port 3.
CFE_EVS_EventOutput_PORT4	Output Port 4.

Definition at line 130 of file default\_cfe\_evs\_extern\_typedefs.h.

#### **12.88.3.3 CFE\_EVS\_EventType enum [CFE\\_EVS\\_EventType](#)**

Label definitions associated with CFE\_EVS\_EventType\_Enum\_t.

Enumerator

CFE_EVS_EventType_DEBUG	Events that are intended only for debugging, not nominal operations.
CFE_EVS_EventType_INFORMATION	Events that identify a state change or action that is not an error.
CFE_EVS_EventType_ERROR	Events that identify an error but are not catastrophic (e.g. - bad command).
CFE_EVS_EventType_CRITICAL	Events that identify errors that are unrecoverable autonomously.

Definition at line 79 of file default\_cfe\_evs\_extern\_typedefs.h.

#### **12.88.3.4 CFE\_EVS\_LogMode enum [CFE\\_EVS\\_LogMode](#)**

Label definitions associated with CFE\_EVS\_LogMode\_Enum\_t.

Enumerator

CFE_EVS_LogMode_OVERWRITE	Overwrite Log Mode.
CFE_EVS_LogMode_DISCARD	Discard Log Mode.

Definition at line 56 of file default\_cfe\_evs\_extern\_typedefs.h.

#### **12.88.3.5 CFE\_EVS\_MsgFormat enum [CFE\\_EVS\\_MsgFormat](#)**

Label definitions associated with CFE\_EVS\_MsgFormat\_Enum\_t.

Enumerator

CFE_EVS_MsgFormat_SHORT	Short Format Messages.
CFE_EVS_MsgFormat_LONG	Long Format Messages.

Definition at line 33 of file default\_cfe\_evs\_extern\_typedefs.h.

## 12.89 cfe/modules/evs/config/default\_cfe\_evs\_fcncodes.h File Reference

### Macros

#### Event Services Command Codes

- #define CFE\_EVS\_NOOP\_CC 0
- #define CFE\_EVS\_RESET\_COUNTERS\_CC 1
- #define CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC 2
- #define CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC 3
- #define CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC 4
- #define CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC 5
- #define CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC 6
- #define CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC 7
- #define CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC 8
- #define CFE\_EVS\_RESET\_APP\_COUNTER\_CC 9
- #define CFE\_EVS\_SET\_FILTER\_CC 10
- #define CFE\_EVS\_ENABLE\_PORTS\_CC 11
- #define CFE\_EVS\_DISABLE\_PORTS\_CC 12
- #define CFE\_EVS\_RESET\_FILTER\_CC 13
- #define CFE\_EVS\_RESET\_ALL\_FILTERS\_CC 14
- #define CFE\_EVS\_ADD\_EVENT\_FILTER\_CC 15
- #define CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC 16
- #define CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC 17
- #define CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC 18
- #define CFE\_EVS\_SET\_LOG\_MODE\_CC 19
- #define CFE\_EVS\_CLEAR\_LOG\_CC 20

#### 12.89.1 Detailed Description

Specification for the CFE Event Services (CFE\_EVS) command function codes

##### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

#### 12.89.2 Macro Definition Documentation

##### 12.89.2.1 CFE\_EVS\_ADD\_EVENT\_FILTER\_CC #define CFE\_EVS\_ADD\_EVENT\_FILTER\_CC 15

**Name** Add Application Event Filter

##### Description

This command adds the given filter for the given application identifier and event identifier. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_AddEvtFltr

##### Command Structure

CFE\_EVS\_AddEventFilterCmd\_t

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of [CFE\\_EVS\\_ADDFILTER\\_EID](#) debug event message

#### Error Conditions

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services
- Specified event ID is already added to the application event filter
- Maximum number of event IDs already added to filter

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

#### Criticality

None.

#### See also

[CFE\\_EVS\\_SET\\_FILTER\\_CC](#), [CFE\\_EVS\\_RESET\\_FILTER\\_CC](#), [CFE\\_EVS\\_RESET\\_ALL\\_FILTERS\\_CC](#),  
[CFE\\_EVS\\_DELETE\\_EVENT\\_FILTER\\_CC](#)

Definition at line 693 of file default\_cfe\_evs\_fcncodes.h.

### 12.89.2.2 CFE\_EVS\_CLEAR\_LOG\_CC #define CFE\_EVS\_CLEAR\_LOG\_CC 20

#### Name

Clear Event Log

#### Description

This command clears the contents of the local event log.

#### Command Mnemonic(s)

[\\$sc\\_\\$cpu\\_EVS\\_ClrLog](#)

#### Command Structure

[CFE\\_EVS\\_ClearLogCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_EVS\_LOGFULL** - The LogFullFlag in the Housekeeping telemetry will be cleared
- **\$sc\_\$cpu\_EVS\_LOGOVERFLOWC** - The LogOverflowCounter in the Housekeeping telemetry will be reset to 0

### Error Conditions

There are no error conditions for this command. If the Event Services receives the command, the log is cleared.

### Criticality

Clearing the local event log is not particularly hazardous, as the result may be making available space to record valuable event data. However, inappropriately clearing the local event log could result in a loss of critical information. Note: the event log is a back-up log to the on-board recorder.

### See also

[CFE\\_EVS\\_WRITE\\_LOG\\_DATA\\_FILE\\_CC](#), [CFE\\_EVS\\_SET\\_LOG\\_MODE\\_CC](#)

Definition at line 873 of file default\_cfe\_evs\_fcncodes.h.

## 12.89.2.3 CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC #define CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC 16

### Name

Delete Application Event Filter

### Description

This command removes the given filter for the given application identifier and event identifier. Note: In order for this command to take effect, applications must be registered for Event Service.

### Command Mnemonic(s)

\$sc\_\$cpu\_EVS\_DelEvtFltr

### Command Structure

[CFE\\_EVS\\_DeleteEventFilterCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of [CFE\\_EVS\\_DELFILTER\\_EID](#) debug event message

### Error Conditions

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services
- Specified event ID is not found in the application event filter

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

### Criticality

None.

### See also

[CFE\\_EVS\\_SET\\_FILTER\\_CC](#), [CFE\\_EVS\\_RESET\\_FILTER\\_CC](#), [CFE\\_EVS\\_RESET\\_ALL\\_FILTERS\\_CC](#),  
[CFE\\_EVS\\_ADD\\_EVENT\\_FILTER\\_CC](#)

Definition at line 728 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.4 CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC** #define CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC 6

**Name** Disable Application Event Type

**Description**

This command disables the command specified event type for the command specified application, preventing the application from sending event messages of the command specified event type through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, critical, and error. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_DisAppEvtType, \$sc\_\$cpu\_EVS\_DisAppEvtTypeMask

**Command Structure**

**CFE\_EVS\_DisableAppEventTypeCmd\_t** The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be disabled (or filtered) for the specified application. A zero in a bit position means the filtering state is unchanged for the specified application.

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of **CFE\_EVS\_DISAPPENTTYPE\_EID** debug event message
- The clearing of the Event Type Active Flag in The Event Type Active Flag in EVS App Data File

**Error Conditions**

This command may fail for the following reason(s):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set
- Application name is not valid or not registered with event services

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

**Criticality**

Disabling an application's event type is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately disabling an application's event type could result in a loss of critical information and missed behavior for the ground system.

**See also**

[CFE\\_EVS\\_ENABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_ENABLE\\_APP\\_EVENT\\_TYPE\\_CC](#),  
[CFE\\_EVS\\_ENABLE\\_APP\\_EVENTS\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENTS\\_CC](#)

Definition at line 353 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.5 CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC** #define CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC 8**Name** Disable Event Services for an Application**Description**

This command disables the command specified application from sending events through Event Service. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_DisAppEvGen**Command Structure**`CFE_EVS_DisableAppEventsCmd_t`**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_EVS_CMDPC` - command execution counter will increment
- The generation of `CFE_EVS_DISAPPEVT_EID` debug event message

**Error Conditions**

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_EVS_CMDEC` - command error counter will increment
- An Error specific event message

**Criticality**

Disabling an application's events is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately disabling an application's events could result in a loss of critical information and missed behavior for the ground system.

**See also**

`CFE_EVS_ENABLE_EVENT_TYPE_CC`, `CFE_EVS_DISABLE_EVENT_TYPE_CC`, `CFE_EVS_ENABLE_APP_EVENT_TYPE_CC`,  
`CFE_EVS_DISABLE_APP_EVENT_TYPE_CC`, `CFE_EVS_ENABLE_APP_EVENTS_CC`

Definition at line 431 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.6 CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC** #define CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC 3**Name** Disable Event Type

### Description

This command disables the command specified Event Type preventing event messages of this type to be sent through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, error and critical. This command is a global disable of a particular event type, it applies to all applications.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_DisEventType, \$sc\_\$cpu\_EVS\_DisEventTypeMask

### Command Structure

**CFE\_EVS\_DisableEventTypeCmd\_t** The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be disabled (or filtered). A zero in a bit position means the filtering state is unchanged.

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of **CFE\_EVS\_DISEVTTYPE\_EID** debug message

### Error Conditions

This command may fail for the following reason(s):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

### Criticality

Disabling an event type is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately disabling an event type could result in a loss of critical information and missed behavior for the ground system.

### See also

[CFE\\_EVS\\_ENABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_ENABLE\\_APP\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENT\\_TYPE\\_CC](#),  
[CFE\\_EVS\\_ENABLE\\_APP\\_EVENTS\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENTS\\_CC](#)

Definition at line 201 of file default\_cfe\_evs\_fcncodes.h.

### 12.89.2.7 CFE\_EVS\_DISABLE\_PORTS\_CC #define CFE\_EVS\_DISABLE\_PORTS\_CC 12

**Name** Disable Event Services Output Ports

### Description

This command disables the specified port from outputting event messages.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_DisPort, \$sc\_\$cpu\_EVS\_DisPortMask

#### Command Structure

[CFE\\_EVS\\_DisablePortsCmd\\_t](#) The following bit positions apply to structure member named 'BitMask'. Bit 0 - Port 1 Bit 1 - Port 2 Bit 2 - Port 3 Bit 3 - Port 4 A one in a bit position means the port will be disabled. A zero in a bit position means the port state is unchanged.

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDPC](#) - command execution counter will increment
- The generation of [CFE\\_EVS\\_DISPORT\\_EID](#) debug event message

#### Error Conditions

This command may fail for the following reason(s):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDEC](#) - command error counter will increment
- An Error specific event message

#### Criticality

None.

#### See also

[CFE\\_EVS\\_ENABLE\\_PORTS\\_CC](#)

Definition at line 587 of file default\_cfe\_evs\_fcncodes.h.

### 12.89.2.8 CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC #define CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC 5

**Name** Enable Application Event Type

#### Description

This command enables the command specified event type for the command specified application, allowing the application to send event messages of the command specified event type through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, critical, and error. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_EnaAppEvtType, \$sc\_\$cpu\_EVS\_EnaAppEvtTypeMask

#### Command Structure

[CFE\\_EVS\\_EnableAppEventTypeCmd\\_t](#) The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be enabled (or unfiltered) for the specified application. A zero in a bit position means the filtering state is unchanged for the specified application.

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_EVS_CMDPC` - command execution counter will increment
- The generation of `CFE_EVS_ENAAPPLEVTTYPE_EID` debug event message

### Error Conditions

This command may fail for the following reason(s):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set
- Application name is not valid or not registered with event services

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_EVS_CMDEC` - command error counter will increment
- An Error specific event message

### Criticality

Enabling an application event type is not particularly hazardous, as the result may be turning on necessary event messages and communication to the ground system. However, inappropriately enabling an application's event type could result in flooding of the ground system.

### See also

[CFE\\_EVS\\_ENABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENT\\_TYPE\\_CC](#),  
[CFE\\_EVS\\_ENABLE\\_APP\\_EVENTS\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENTS\\_CC](#)

Definition at line 300 of file default\_cfe\_evs\_fcncodes.h.

#### 12.89.2.9 CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC #define CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC 7

**Name** Enable Event Services for an Application

### Description

This command enables the command specified application to send events through the Event Service. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_EnaAppEvGen

### Command Structure

[CFE\\_EVS\\_EnableAppEventsCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_EVS_CMDPC` - command execution counter will increment
- The generation of `CFE_EVS_ENAAPPLEVT_EID` debug event message
- The setting of the Active Flag in The Active Flag in EVS App Data File

## Error Conditions

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

## Criticality

Enabling an application events is not particularly hazardous, as the result may be turning on necessary event messages and communication to the ground system. However, inappropriately enabling an application's events could result in flooding of the ground system.

## See also

[CFE\\_EVS\\_ENABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_ENABLE\\_APP\\_EVENT\\_TYPE\\_CC](#),  
[CFE\\_EVS\\_DISABLE\\_APP\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENTS\\_CC](#)

Definition at line 392 of file default\_cfe\_evs\_fcncodes.h.

### 12.89.2.10 CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC #define CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC 2

**Name** Enable Event Type

#### Description

This command enables the command specified Event Type allowing event messages of this type to be sent through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, error and critical. This command is a global enable of a particular event type, it applies to all applications.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_EnaEventType, \$sc\_\$cpu\_EVS\_EnaEventTypeMask

#### Command Structure

[CFE\\_EVS\\_EnableEventTypeCmd\\_t](#) The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be enabled (or unfiltered). A zero in a bit position means the filtering state is unchanged.

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of [CFE\\_EVS\\_ENAEVTTYPE\\_EID](#) debug message

### Error Conditions

This command may fail for the following reason(s):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_EVS_CMDEC` - command error counter will increment
- An Error specific event message

### Criticality

Enabling an event type is not particularly hazardous, as the result may be turning on necessary event messages and communication to the ground system. However, inappropriately enabling an event type could result in flooding of the system.

### See also

[CFE\\_EVS\\_DISABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_ENABLE\\_APP\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_ENABLE\\_APP\\_EVENTS\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENTS\\_CC](#)

Definition at line 152 of file default\_cfe\_evs\_fcncodes.h.

### 12.89.2.11 CFE\_EVS\_ENABLE\_PORTS\_CC #define CFE\_EVS\_ENABLE\_PORTS\_CC 11

**Name** Enable Event Services Output Ports

#### Description

This command enables the command specified port to output event messages

**Command Mnemonic(s)** `$sc_$cpu_EVS_EnaPort`, `$sc_$cpu_EVS_EnaPortMask`

#### Command Structure

[CFE\\_EVS\\_EnablePortsCmd\\_t](#) The following bit positions apply to structure member named 'BitMask'. Bit 0 - Port 1 Bit 1 - Port 2 Bit 2 - Port 3 Bit 3 - Port 4 A one in a bit position means the port will be enabled. A zero in a bit position means the port state is unchanged.

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_EVS_CMDPC` - command execution counter will increment
- The generation of [CFE\\_EVS\\_ENAPORT\\_EID](#) debug event message

### Error Conditions

This command may fail for the following reason(s):

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_EVS_CMDEC` - command error counter will increment
- An Error specific event message

**Criticality**

None.

**See also**

[CFE\\_EVS\\_DISABLE\\_PORTS\\_CC](#)

Definition at line 548 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.12 CFE\_EVS\_NOOP\_CC #define CFE\_EVS\_NOOP\_CC 0**

**Name** Event Services No-Op

**Description**

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Event Services task.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_NOOP

**Command Structure**

[CFE\\_EVS\\_NoopCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The [CFE\\_EVS\\_NOOP\\_EID](#) informational event message will be generated

**Error Conditions**

There are no error conditions for this command. If the Event Services receives the command, the event is sent (although it may be filtered by EVS itself) and the counter is incremented unconditionally.

**Criticality**

None

**See also**

Definition at line 65 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.13 CFE\_EVS\_RESET\_ALL\_FILTERS\_CC #define CFE\_EVS\_RESET\_ALL\_FILTERS\_CC 14**

**Name** Reset All Event Filters for an Application

## Description

This command resets all of the command specified applications event filters. Note: In order for this command to take effect, applications must be registered for Event Service.

## Command Mnemonic(s) \$sc\_\$cpu\_EVS\_RstAllFltrs

### Command Structure

[CFE\\_EVS\\_ResetAllFiltersCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDPC](#) - command execution counter will increment
- The generation of [CFE\\_EVS\\_RSTALLFILTER\\_EID](#) debug event message

### Error Conditions

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDEC](#) - command error counter will increment
- An Error specific event message

### Criticality

None.

### See also

[CFE\\_EVS\\_SET\\_FILTER\\_CC](#), [CFE\\_EVS\\_RESET\\_FILTER\\_CC](#), [CFE\\_EVS\\_ADD\\_EVENT\\_FILTER\\_CC](#),  
[CFE\\_EVS\\_DELETE\\_EVENT\\_FILTER\\_CC](#)

Definition at line 657 of file default\_cfe\_evs\_fcncodes.h.

### 12.89.2.14 CFE\_EVS\_RESET\_APP\_COUNTER\_CC #define CFE\_EVS\_RESET\_APP\_COUNTER\_CC 9

## Name Reset Application Event Counters

### Description

This command sets the command specified application's event counter to zero. Note: In order for this command to take effect, applications must be registered for Event Service.

## Command Mnemonic(s) \$sc\_\$cpu\_EVS\_RstAppCtrs

### Command Structure

[CFE\\_EVS\\_ResetAppCounterCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of [CFE\\_EVS\\_RSTEVTCNT\\_EID](#) debug event message

### Error Conditions

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

### Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter value that is reset by this command.

### See also

[CFE\\_EVS\\_RESET\\_COUNTERS\\_CC](#)

Definition at line 467 of file default\_cfe\_evs\_fncodes.h.

## 12.89.2.15 CFE\_EVS\_RESET\_COUNTERS\_CC #define CFE\_EVS\_RESET\_COUNTERS\_CC 1

**Name** Event Services Reset Counters

### Description

This command resets the following counters within the Event Services housekeeping telemetry:

- Command Execution Counter (\$sc\_\$cpu\_EVS\_CMDPC)
- Command Error Counter (\$sc\_\$cpu\_EVS\_CMDEC)

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_ResetCtrs

### Command Structure

[CFE\\_EVS\\_ResetCountersCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will be reset to 0
- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will be reset to 0
- The [CFE\\_EVS\\_RSTCNT\\_EID](#) debug event message will be generated

### Error Conditions

There are no error conditions for this command. If the Event Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

### Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

### See also

[CFE\\_EVS\\_RESET\\_APP\\_COUNTER\\_CC](#)

Definition at line 104 of file default\_cfe\_evs\_fcncodes.h.

## 12.89.2.16 CFE\_EVS\_RESET\_FILTER\_CC #define CFE\_EVS\_RESET\_FILTER\_CC 13

**Name** Reset an Event Filter for an Application

### Description

This command resets the command specified application's event filter for the command specified event ID. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_RstBinFltrCtr

### Command Structure

[CFE\\_EVS\\_ResetFilterCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of [CFE\\_EVS\\_RSTFILTER\\_EID](#) debug event message

### Error Conditions

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services
- Specified event ID is not found in the application event filter

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

### Criticality

None.

### See also

[CFE\\_EVS\\_SET\\_FILTER\\_CC](#), [CFE\\_EVS\\_RESET\\_ALL\\_FILTERS\\_CC](#), [CFE\\_EVS\\_ADD\\_EVENT\\_FILTER\\_CC](#),  
[CFE\\_EVS\\_DELETE\\_EVENT\\_FILTER\\_CC](#)

Definition at line 623 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.17 CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC** #define CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC 4

**Name** Set Event Format Mode

**Description**

This command sets the event format mode to the command specified value. The event format mode may be either short or long. A short event format detaches the Event Data from the event message and only includes the following information in the event packet: Processor ID, Application ID, Event ID, and Event Type. Refer to section 5.3.3.4 for a description of the Event Service event packet contents. Event Data is defined to be data describing an Event that is supplied to the cFE Event Service. ASCII text strings are used as the primary format for Event Data because heritage ground systems use string compares as the basis for their automated alert systems. Two systems, ANSR and SERS were looked at for interface definitions. The short event format is used to accommodate experiences with limited telemetry bandwidth. The long event format includes all event information included within the short format along with the Event Data.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_SetEvtFmt

**Command Structure**

[CFE\\_EVS\\_SetEventFormatModeCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDPC](#) - command execution counter will increment
- The generation of [CFE\\_EVS\\_SETEVTFMTMOD\\_EID](#) debug message

**Error Conditions**

This command may fail for the following reason(s):

- Invalid MsgFormat mode selection

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDEC](#) - command error counter will increment
- An Error specific event message

**Criticality**

Setting the event format mode is not particularly hazardous, as the result may be saving necessary bandwidth. However, inappropriately setting the event format mode could result in a loss of information and missed behavior for the ground system

**See also**

Definition at line 248 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.18 CFE\_EVS\_SET\_FILTER\_CC** #define CFE\_EVS\_SET\_FILTER\_CC 10

**Name** Set Application Event Filter

**Description**

This command sets the command specified application's event filter mask to the command specified value for the command specified event. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_SetBinFltrMask

**Command Structure**

[CFE\\_EVS\\_SetFilterCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDPC](#) - command execution counter will increment
- The generation of [CFE\\_EVS\\_SETFILTERMSK\\_EID](#) debug event message

**Error Conditions**

This command may fail for the following reason(s):

- Application name is not valid or not registered with event services
- Specified event ID is not found in the application event filter

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDEC](#) - command error counter will increment
- An Error specific event message

**Criticality**

Setting an application event filter mask is not particularly hazardous, as the result may be shutting off unnecessary event messages and possible event flooding of the system. However, inappropriately setting an application's event filter mask could result in a loss of critical information and missed behavior for the ground system or flooding of the ground system.

**See also**

[CFE\\_EVS\\_RESET\\_FILTER\\_CC](#), [CFE\\_EVS\\_RESET\\_ALL\\_FILTERS\\_CC](#), [CFE\\_EVS\\_ADD\\_EVENT\\_FILTER\\_CC](#),  
[CFE\\_EVS\\_DELETE\\_EVENT\\_FILTER\\_CC](#)

Definition at line 509 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.19 CFE\_EVS\_SET\_LOG\_MODE\_CC** #define CFE\_EVS\_SET\_LOG\_MODE\_CC 19**Name** Set Logging Mode**Description**

This command sets the logging mode to the command specified value.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_SetLogMode**Command Structure**

[CFE\\_EVS\\_SetLogModeCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDPC](#) - command execution counter will increment
- The generation of [CFE\\_EVS\\_LOGMODE\\_EID](#) debug event message

**Error Conditions**

This command may fail for the following reason(s):

- Invalid LogMode selected - must be either [CFE\\_EVS\\_LogMode\\_OVERWRITE](#) or [CFE\\_EVS\\_LogMode\\_DISCARD](#)

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDEC](#) - command error counter will increment
- An Error specific event message

**Criticality**

Setting the event logging mode is not particularly hazardous, as the result may be saving valuable event data. However, inappropriately setting the log mode could result in a loss of critical information. Note: the event log is a back-up log to the on-board recorder.

**See also**

[CFE\\_EVS\\_WRITE\\_LOG\\_DATA\\_FILE\\_CC](#), [CFE\\_EVS\\_CLEAR\\_LOG\\_CC](#)

Definition at line 838 of file default\_cfe\_evs\_fcncodes.h.

**12.89.2.20 CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC** #define CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC 17**Name** Write Event Services Application Information to File**Description**

This command writes all application data to a file for all applications that have registered with the EVS. The application data includes the Application ID, Active Flag, Event Count, Event Types Active Flag, and Filter Data.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_WriteAppData2File

#### Command Structure

[CFE\\_EVS\\_WriteAppDataFileCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of [CFE\\_EVS\\_WRDAT\\_EID](#) debug event message
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_EVS\\_DEFAULT\\_APP\\_DATA\\_FILE](#) configuration parameter) will be updated with the latest information.

#### Error Conditions

This command may fail for the following reason(s):

- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDEC** - command error counter will increment
- An Error specific event message

#### Criticality

Writing a file is not particularly hazardous, but if proper file management is not taken, then the file system can fill up if this command is used repeatedly.

#### See also

[CFE\\_EVS\\_WRITE\\_LOG\\_DATA\\_FILE\\_CC](#), [CFE\\_EVS\\_SET\\_LOG\\_MODE\\_CC](#)

Definition at line 767 of file default\_cfe\_evs\_fcncodes.h.

### 12.89.2.21 CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC #define CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC 18

**Name** Write Event Log to File

#### Description

This command requests the Event Service to generate a file containing the contents of the local event log.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_WriteLog2File

#### Command Structure

[CFE\\_EVS\\_WriteLogFileCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_EVS\_CMDPC** - command execution counter will increment
- The generation of [CFE\\_EVS\\_WRLOG\\_EID](#) debug event message

## Error Conditions

This command may fail for the following reason(s):

- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_EVS_CMDEC` - command error counter will increment
- An Error specific event message

## Criticality

Writing a file is not particularly hazardous, but if proper file management is not taken, then the file system can fill up if this command is used repeatedly.

## See also

[CFE\\_EVS\\_WRITE\\_APP\\_DATA\\_FILE\\_CC](#), [CFE\\_EVS\\_SET\\_LOG\\_MODE\\_CC](#), [CFE\\_EVS\\_CLEAR\\_LOG\\_CC](#)

Definition at line 802 of file default\_cfe\_evs\_fcncodes.h.

## 12.90 cfe/modules/evs/config/default\_cfe\_evs\_interface\_cfg.h File Reference

### Macros

- `#define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122`

#### 12.90.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

#### 12.90.2 Macro Definition Documentation

##### 12.90.2.1 CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH `#define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122`

**Purpose** Maximum Event Message Length

**Description:**

Indicates the maximum length (in characters) of the formatted text string portion of an event message

This length does not need to include an extra character for NULL termination.

### Limits

Not Applicable

Definition at line 47 of file default\_cfe\_evs\_interface\_cfg.h.

## 12.91 cfe/modules/evs/config/default\_cfe\_evs\_internal\_cfg.h File Reference

### Macros

- #define CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY 61
- #define CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS 8
- #define CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST 32
- #define CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC 15
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE "/ram/cfe\_evs.log"
- #define CFE\_PLATFORM\_EVS\_LOG\_MAX 20
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE "/ram/cfe\_evs\_app.dat"
- #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT 0x0001
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG 0xE
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE 1
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE CFE\_EVS\_MsgFormat\_LONG

### 12.91.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.91.2 Macro Definition Documentation

#### 12.91.2.1 CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC #define CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC EC 15

**Purpose** Sustained number of event messages per second per app before squelching

#### Description:

Sustained number of events that may be emitted per app per second.

#### Limits

This number must be less than or equal to CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST. Values lower than 8 may cause functional and unit test failures.

Definition at line 96 of file default\_cfe\_evs\_internal\_cfg.h.

#### 12.91.2.2 CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE #define CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATAFILE "/ram/cfe\_evs\_app.dat"

**Purpose** Default EVS Application Data Filename

**Description:**

The value of this constant defines the filename used to store the EVS Application Data(event counts/filtering information). This filename is used only when no filename is specified in the command to dump the event log.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 137 of file default\_cfe\_evs\_internal\_cfg.h.

**12.91.2.3 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE** `#define CFE_PLATFORM_EVS_DEFAULT_LOG_FILE "/ram/cfe-evs.log"`

**Purpose** Default Event Log Filename**Description:**

The value of this constant defines the filename used to store the Event Services local event log. This filename is used only when no filename is specified in the command to dump the event log.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 110 of file default\_cfe\_evs\_internal\_cfg.h.

**12.91.2.4 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE** `#define CFE_PLATFORM_EVS_DEFAULT_LOG_MODE 1`

**Purpose** Default EVS Local Event Log Mode**Description:**

Defines a state of overwrite(0) or discard(1) for the operation of the EVS local event log. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest event in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. Overwrite Mode = 0, Discard Mode = 1.

**Limits**

The valid settings are 0 or 1

Definition at line 184 of file default\_cfe\_evs\_internal\_cfg.h.

**12.91.2.5 CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE** `#define CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE CFE_EVS_MsgFormat_LONG`

**Purpose** Default EVS Message Format Mode**Description:**

Defines the default message format (long or short) for event messages being sent to the ground. Choose between [CFE\\_EVS\\_MsgFormat\\_LONG](#) or [CFE\\_EVS\\_MsgFormat\\_SHORT](#).

**Limits**

The valid settings are [CFE\\_EVS\\_MsgFormat\\_LONG](#) or [CFE\\_EVS\\_MsgFormat\\_SHORT](#)

Definition at line 197 of file default\_cfe\_evs\_internal\_cfg.h.

```
12.91.2.6 CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG #define CFE_PLATFORM_EVS_DEFAULT_TYPE_FL←  
AG 0xE
```

**Purpose** Default EVS Event Type Filter Mask

**Description:**

Defines a state of on or off for all four event types. The term event 'type' refers to the criticality level and may be Debug, Informational, Error or Critical. Each event type has a bit position. (bit 0 = Debug, bit 1 = Info, bit 2 = Error, bit 3 = Critical). This is a global setting, meaning it applies to all applications. To filter an event type, set its bit to zero. For example, 0xE means Debug = OFF, Info = ON, Error = ON, Critical = ON

**Limits**

The valid settings are 0x0 to 0xF.

Definition at line 168 of file default\_cfe\_evs\_internal\_cfg.h.

```
12.91.2.7 CFE_PLATFORM_EVS_LOG_MAX #define CFE_PLATFORM_EVS_LOG_MAX 20
```

**Purpose** Maximum Number of Events in EVS Local Event Log

**Description:**

Dictates the EVS local event log capacity. Units are the number of events.

**Limits**

There are no restrictions on the lower and upper limits however, the maximum log size is system dependent and should be verified.

Definition at line 122 of file default\_cfe\_evs\_internal\_cfg.h.

```
12.91.2.8 CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST #define CFE_PLATFORM_EVS_MAX_APP_EVENT_B←  
URST 32
```

**Purpose** Maximum number of event before squelching

**Description:**

Maximum number of events that may be emitted per app per second. Setting this to 0 will cause events to be unrestricted.

**Limits**

This number must be less than or equal to INT\_MAX/1000

Definition at line 84 of file default\_cfe\_evs\_internal\_cfg.h.

**12.91.2.9 CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS** #define CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS 8

**Purpose** Define Maximum Number of Event Filters per Application

**Description:**

Maximum number of events that may be filtered per application.

**Limits**

There are no restrictions on the lower and upper limits however, the maximum number of event filters is system dependent and should be verified.

Definition at line 72 of file default\_cfe\_evs\_internal\_cfg.h.

**12.91.2.10 CFE\_PLATFORM\_EVS\_PORT\_DEFAULT** #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT 0x0001

**Purpose** Default EVS Output Port State

**Description:**

Defines the default port state (enabled or disabled) for the four output ports defined within the Event Service. Port 1 is usually the uart output terminal. To enable a port, set the proper bit to a 1. Bit 0 is port 1, bit 1 is port2 etc.

**Limits**

The valid settings are 0x0 to 0xF.

Definition at line 151 of file default\_cfe\_evs\_internal\_cfg.h.

**12.91.2.11 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 44 of file default\_cfe\_evs\_internal\_cfg.h.

**12.91.2.12 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 59 of file default\_cfe\_evs\_internal\_cfg.h.

## 12.92 cfe/modules/evs/config/default\_cfe\_evs\_mission\_cfg.h File Reference

```
#include "cfe_evs_interface_cfg.h"
```

### 12.92.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

#### Note

This file may be overridden/superceded by mission-provided defintions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.93 cfe/modules/evs/config/default\_cfe\_evs\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_evs_fcncodes.h"
#include "cfe_evs_msgstruct.h"
```

### 12.93.1 Detailed Description

Specification for the CFE Event Services (CFE\_EVS) command and telemetry message data types.

This is a compatibility header for the "cfe\_evs\_msg.h" file that has traditionally provided the message definitions for cFS apps.

#### Note

This file may be overridden/superceded by mission-provided defintions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.94 cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h File Reference

```
#include "cfe_evs_fcncodes.h"
```

### 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

### 12.94.1 Detailed Description

Specification for the CFE Event Services (CFE\_EVS) command and telemetry message constant definitions.  
For CFE\_EVS this is only the function/command code definitions

## 12.94.2 Macro Definition Documentation

**12.94.2.1 CFE\_EVS\_CRITICAL\_BIT** #define CFE\_EVS\_CRITICAL\_BIT 0x0008  
Definition at line 35 of file default\_cfe\_evs\_msgdefs.h.

**12.94.2.2 CFE\_EVS\_DEBUG\_BIT** #define CFE\_EVS\_DEBUG\_BIT 0x0001  
Definition at line 32 of file default\_cfe\_evs\_msgdefs.h.

**12.94.2.3 CFE\_EVS\_ERROR\_BIT** #define CFE\_EVS\_ERROR\_BIT 0x0004  
Definition at line 34 of file default\_cfe\_evs\_msgdefs.h.

**12.94.2.4 CFE\_EVS\_INFORMATION\_BIT** #define CFE\_EVS\_INFORMATION\_BIT 0x0002  
Definition at line 33 of file default\_cfe\_evs\_msgdefs.h.

**12.94.2.5 CFE\_EVS\_PORT1\_BIT** #define CFE\_EVS\_PORT1\_BIT 0x0001  
Definition at line 38 of file default\_cfe\_evs\_msgdefs.h.

**12.94.2.6 CFE\_EVS\_PORT2\_BIT** #define CFE\_EVS\_PORT2\_BIT 0x0002  
Definition at line 39 of file default\_cfe\_evs\_msgdefs.h.

**12.94.2.7 CFE\_EVS\_PORT3\_BIT** #define CFE\_EVS\_PORT3\_BIT 0x0004  
Definition at line 40 of file default\_cfe\_evs\_msgdefs.h.

**12.94.2.8 CFE\_EVS\_PORT4\_BIT** #define CFE\_EVS\_PORT4\_BIT 0x0008  
Definition at line 41 of file default\_cfe\_evs\_msgdefs.h.

## 12.95 cfe/modules/evs/config/default\_cfe\_evs\_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_evs_topicids.h"
```

### Macros

- #define CFE\_EVS\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_EVS\_CMD\_MSG /\* 0x1801 \*/
- #define CFE\_EVS\_SEND\_HK\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_EVS\_SEND\_HK\_MSG /\* 0x1809 \*/
- #define CFE\_EVS\_HK\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_EVS\_HK\_TLM\_MSG /\* 0x0801 \*/
- #define CFE\_EVS\_LONG\_EVENT\_MSG\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG /\* 0x0808 \*/
- #define CFE\_EVS\_SHORT\_EVENT\_MSG\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_EVS\_SHORT\_EVENT\_MS /\* 0x0809 \*/

### 12.95.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Message IDs

### 12.95.2 Macro Definition Documentation

**12.95.2.1 CFE\_EVS\_CMD\_MID** #define CFE\_EVS\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_EVS\_CMD\_MSG  
/\* 0x1801 \*/  
Definition at line 32 of file default\_cfe\_evs\_msgids.h.

**12.95.2.2 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 38 of file default\_cfe\_evs\_msgids.h.

**12.95.2.3 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 39 of file default\_cfe\_evs\_msgids.h.

**12.95.2.4 CFE\_EVS\_SEND\_HK\_MID** #define CFE\_EVS\_SEND\_HK\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_EVS\_SEND\_HK  
/\* 0x1809 \*/  
Definition at line 33 of file default\_cfe\_evs\_msgids.h.

**12.95.2.5 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 40 of file default\_cfe\_evs\_msgids.h.

## 12.96 cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h File Reference

```
#include "common_types.h"
#include "cfe_evs_msgdefs.h"
#include "cfe_evs_extern_typedefs.h"
#include "cfe_msg_hdr.h"
```

### Data Structures

- struct **CFE\_EVS\_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](#)

## 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 [CFE\\_EVS\\_NoArgsCmd\\_t](#) [CFE\\_EVS\\_SendHkCmd\\_t](#)
- typedef struct [CFE\\_EVS\\_LogFileCmd\\_Payload](#) [CFE\\_EVS\\_LogFileCmd\\_Payload\\_t](#)  
*Write Event Log to File Command Payload.*
- typedef struct [CFE\\_EVS\\_WriteLogFileCmd](#) [CFE\\_EVS\\_WriteLogFileCmd\\_t](#)

*Write Event Log to File Command.*

- **typedef struct CFE\_EVS\_AppDataCmd\_Payload CFE\_EVS\_AppDataCmd\_Payload\_t**  
*Write Event Services Application Information to File Command Payload.*
- **typedef struct CFE\_EVS\_WriteAppDataFileCmd CFE\_EVS\_WriteAppDataFileCmd\_t**  
*Write Event Services Application Information to File Command.*
- **typedef struct CFE\_EVS\_SetLogMode\_Payload CFE\_EVS\_SetLogMode\_Payload\_t**  
*Set Log Mode Command Payload.*
- **typedef struct CFE\_EVS\_SetLogModeCmd CFE\_EVS\_SetLogModeCmd\_t**  
*Set Log Mode Command.*
- **typedef struct CFE\_EVS\_SetEventFormatCode\_Payload CFE\_EVS\_SetEventFormatMode\_Payload\_t**  
*Set Event Format Mode Command Payload.*
- **typedef struct CFE\_EVS\_SetEventFormatModeCmd CFE\_EVS\_SetEventFormatModeCmd\_t**  
*Set Event Format Mode Command.*
- **typedef struct CFE\_EVS\_BitMaskCmd\_Payload CFE\_EVS\_BitMaskCmd\_Payload\_t**  
*Generic Bitmask Command Payload.*
- **typedef struct CFE\_EVS\_BitMaskCmd CFE\_EVS\_BitMaskCmd\_t**  
*Generic Bitmask Command.*
- **typedef CFE\_EVS\_BitMaskCmd\_t CFE\_EVS\_EnablePortsCmd\_t**
- **typedef CFE\_EVS\_BitMaskCmd\_t CFE\_EVS\_DisablePortsCmd\_t**
- **typedef CFE\_EVS\_BitMaskCmd\_t CFE\_EVS\_EnableEventTypeCmd\_t**
- **typedef CFE\_EVS\_BitMaskCmd\_t CFE\_EVS\_DisableEventTypeCmd\_t**
- **typedef struct CFE\_EVS\_AppNameCmd\_Payload CFE\_EVS\_AppNameCmd\_Payload\_t**  
*Generic App Name Command Payload.*
- **typedef struct CFE\_EVS\_AppNameCmd CFE\_EVS\_AppNameCmd\_t**  
*Generic App Name Command.*
- **typedef CFE\_EVS\_AppNameCmd\_t CFE\_EVS\_EnableAppEventsCmd\_t**
- **typedef CFE\_EVS\_AppNameCmd\_t CFE\_EVS\_DisableAppEventsCmd\_t**
- **typedef CFE\_EVS\_AppNameCmd\_t CFE\_EVS\_ResetAppCounterCmd\_t**
- **typedef CFE\_EVS\_AppNameCmd\_t CFE\_EVS\_ResetAllFiltersCmd\_t**
- **typedef struct CFE\_EVS\_AppNameEventIDCmd\_Payload CFE\_EVS\_AppNameEventIDCmd\_Payload\_t**  
*Generic App Name and Event ID Command Payload.*
- **typedef struct CFE\_EVS\_AppNameEventIDCmd CFE\_EVS\_AppNameEventIDCmd\_t**  
*Generic App Name and Event ID Command.*
- **typedef CFE\_EVS\_AppNameEventIDCmd\_t CFE\_EVS\_ResetFilterCmd\_t**
- **typedef CFE\_EVS\_AppNameEventIDCmd\_t CFE\_EVS\_DeleteEventFilterCmd\_t**
- **typedef struct CFE\_EVS\_AppNameBitMaskCmd\_Payload CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t**  
*Generic App Name and Bitmask Command Payload.*
- **typedef struct CFE\_EVS\_AppNameBitMaskCmd CFE\_EVS\_AppNameBitMaskCmd\_t**  
*Generic App Name and Bitmask Command.*
- **typedef CFE\_EVS\_AppNameBitMaskCmd\_t CFE\_EVS\_EnableAppEventTypeCmd\_t**
- **typedef CFE\_EVS\_AppNameBitMaskCmd\_t CFE\_EVS\_DisableAppEventTypeCmd\_t**
- **typedef struct CFE\_EVS\_AppNameEventIDMaskCmd\_Payload CFE\_EVS\_AppNameEventIDMaskCmd\_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`

### 12.96.1 Detailed Description

Purpose: cFE Executive Services (EVS) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide

Notes:

### 12.96.2 Typedef Documentation

#### 12.96.2.1 `CFE_EVS_AddEventFilterCmd_t` `typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_AddEventFilterCmd_t`

Definition at line 295 of file default\_cfe\_evs\_msgstruct.h.

#### 12.96.2.2 `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](#)

#### 12.96.2.3 `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](#)

#### 12.96.2.4 `CFE_EVS_AppNameBitMaskCmd_t` `typedef struct CFE_EVS_AppNameBitMaskCmd CFE_EVS_AppNameBitMaskCmd_t`

Generic App Name and Bitmask Command.

#### 12.96.2.5 `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](#)

#### 12.96.2.6 `CFE_EVS_AppNameCmd_t` `typedef struct CFE_EVS_AppNameCmd CFE_EVS_AppNameCmd_t`

Generic App Name Command.

#### 12.96.2.7 `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](#)

#### 12.96.2.8 `CFE_EVS_AppNameEventIDCmd_t` `typedef struct CFE_EVS_AppNameEventIDCmd CFE_EVS_AppNameEventIDCmd_t`

Generic App Name and Event ID Command.

**12.96.2.9 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](#)

**12.96.2.10 CFE\_EVS\_AppNameEventIDMaskCmd\_t** `typedef struct CFE_EVS_AppNameEventIDMaskCmd CFE_EVS_AppNameEventIDMaskCmd_t`

Generic App Name, Event ID, Mask Command.

**12.96.2.11 CFE\_EVS\_AppTlmData\_t** `typedef struct CFE_EVS_AppTlmData CFE_EVS_AppTlmData_t`

**12.96.2.12 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](#)

**12.96.2.13 CFE\_EVS\_BitMaskCmd\_t** `typedef struct CFE_EVS_BitMaskCmd CFE_EVS_BitMaskCmd_t`

Generic Bitmask Command.

**12.96.2.14 CFE\_EVS\_ClearLogCmd\_t** `typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ClearLogCmd_t`

Definition at line 60 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.15 CFE\_EVS\_DeleteEventFilterCmd\_t** `typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_DeleteEventFilterCmd_t`

Definition at line 235 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.16 CFE\_EVS\_DisableAppEventsCmd\_t** `typedef CFE_EVS_AppNameCmd_t CFE_EVS_DisableAppEventsCmd_t`

Definition at line 204 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.17 CFE\_EVS\_DisableAppEventTypeCmd\_t** `typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_DisableAppEventTypeCmd_t`

Definition at line 265 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.18 CFE\_EVS\_DisableEventTypeCmd\_t** `typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisableEventTypeCmd_t`

Definition at line 175 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.19 CFE\_EVS\_DisablePortsCmd\_t** `typedef CFE_EVS_BitMaskCmd_t CFE_EVS_DisablePortsCmd_t`

Definition at line 173 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.20 CFE\_EVS\_EnableAppEventsCmd\_t** `typedef CFE_EVS_AppNameCmd_t CFE_EVS_EnableAppEventsCmd_t`

Definition at line 203 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.21 CFE\_EVS\_EnableAppEventTypeCmd\_t** `typedef CFE_EVS_AppNameBitMaskCmd_t CFE_EVS_EnableAppEventTypeCmd_t`  
Definition at line 264 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.22 CFE\_EVS\_EnableEventTypeCmd\_t** `typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnableEventTypeCmd_t`  
Definition at line 174 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.23 CFE\_EVS\_EnablePortsCmd\_t** `typedef CFE_EVS_BitMaskCmd_t CFE_EVS_EnablePortsCmd_t`  
Definition at line 172 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.24 CFE\_EVS\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_EVS_HousekeepingTlm_Payload CFE_EVS_HousekeepingTlm_Payload_t`

**Name** Event Services Housekeeping Telemetry Packet

**12.96.2.25 CFE\_EVS\_HousekeepingTlm\_t** `typedef struct CFE_EVS_HousekeepingTlm CFE_EVS_HousekeepingTlm_t`

**12.96.2.26 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](#)

**12.96.2.27 CFE\_EVS\_LongEventTlm\_Payload\_t** `typedef struct CFE_EVS_LongEventTlm_Payload CFE_EVS_LongEventTlm_Payload_t`

**Name** Event Message Telemetry Packet (Long format)

**12.96.2.28 CFE\_EVS\_LongEventTlm\_t** `typedef struct CFE_EVS_LongEventTlm CFE_EVS_LongEventTlm_t`

**12.96.2.29 CFE\_EVS\_NoArgsCmd\_t** `typedef struct CFE_EVS_NoArgsCmd CFE_EVS_NoArgsCmd_t`  
Command with no additional arguments.

**12.96.2.30 CFE\_EVS\_NoopCmd\_t** `typedef CFE_EVS_NoArgsCmd_t CFE_EVS_NoopCmd_t`

Definition at line 58 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.31 CFE\_EVS\_PacketID\_t** `typedef struct CFE_EVS_PacketID CFE_EVS_PacketID_t`  
Telemetry packet structures

**12.96.2.32 CFE\_EVS\_ResetAllFiltersCmd\_t** `typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAllFiltersCmd_t`  
Definition at line 206 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.33 CFE\_EVS\_ResetAppCounterCmd\_t** `typedef CFE_EVS_AppNameCmd_t CFE_EVS_ResetAppCounterCmd_t`  
Definition at line 205 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.34 CFE\_EVS\_ResetCountersCmd\_t** `typedef CFE_EVS_NoArgsCmd_t CFE_EVS_ResetCountersCmd_t`  
Definition at line 59 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.35 CFE\_EVS\_ResetFilterCmd\_t** `typedef CFE_EVS_AppNameEventIDCmd_t CFE_EVS_ResetFilterCmd_t`  
Definition at line 234 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.36 CFE\_EVS\_SendHkCmd\_t** `typedef CFE_EVS_NoArgsCmd_t CFE_EVS_SendHkCmd_t`  
Definition at line 61 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.37 CFE\_EVS\_SetEventFormatMode\_Payload\_t** `typedef struct CFE_EVS_SetEventFormatCode_Payload CFE_EVS_SetEventFormatMode_Payload`  
Set Event Format Mode Command Payload.  
For command details, see [CFE\\_EVS\\_SET\\_EVENT\\_FORMAT\\_MODE\\_CC](#)

**12.96.2.38 CFE\_EVS\_SetEventFormatModeCmd\_t** `typedef struct CFE_EVS_SetEventFormatModeCmd CFE_EVS_SetEventFormatModeCmd_t`  
Set Event Format Mode Command.

**12.96.2.39 CFE\_EVS\_SetFilterCmd\_t** `typedef CFE_EVS_AppNameEventIDMaskCmd_t CFE_EVS_SetFilterCmd_t`  
Definition at line 296 of file default\_cfe\_evs\_msgstruct.h.

**12.96.2.40 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](#)

**12.96.2.41 CFE\_EVS\_SetLogModeCmd\_t** `typedef struct CFE_EVS_SetLogModeCmd CFE_EVS_SetLogModeCmd_t`  
Set Log Mode Command.

**12.96.2.42 CFE\_EVS\_ShortEventTlm\_Payload\_t** `typedef struct CFE_EVS_ShortEventTlm_Payload CFE_EVS_ShortEventTlm_Payload_t`

**Name** Event Message Telemetry Packet (Short format)

**12.96.2.43 CFE\_EVS\_ShortEventTlm\_t** `typedef struct CFE_EVS_ShortEventTlm CFE_EVS_ShortEventTlm_t`

**12.96.2.44 CFE\_EVS\_WriteAppDataFileCmd\_t** `typedef struct CFE_EVS_WriteAppDataFileCmd CFE_EVS_WriteAppDataFileCmd_t`  
Write Event Services Application Information to File Command.

**12.96.2.45 CFE\_EVS\_WriteLogFileCmd\_t** `typedef struct CFE_EVS_WriteLogFileCmd CFE_EVS_WriteLogFileCmd_t`  
Write Event Log to File Command.

## 12.97 cfe/modules/evs/config/default\_cfe\_evs\_platform\_cfg.h File Reference

```
#include "cfe_evs_mission_cfg.h"
#include "cfe_evs_internal_cfg.h"
```

### 12.97.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Platform Configuration Header File

This is a compatibility header for the "platform\_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.98 cfe/modules/evs/config/default\_cfe\_evs\_topicids.h File Reference

### Macros

- #define CFE\_MISSION\_EVS\_CMD\_MSG 1
- #define CFE\_MISSION\_EVS\_SEND\_HK\_MSG 9
- #define CFE\_MISSION\_EVS\_HK\_TLM\_MSG 1
- #define CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_MSG 8
- #define CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_MSG 9

### 12.98.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Topic IDs

### 12.98.2 Macro Definition Documentation

#### 12.98.2.1 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 EVS command messages

##### Limits

Not Applicable

Definition at line 35 of file default\_cfe\_evs\_topicids.h.

#### 12.98.2.2 CFE\_MISSION\_EVS\_HK\_TLM\_MSG #define CFE\_MISSION\_EVS\_HK\_TLM\_MSG 1

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE EVS telemetry messages

**Limits**

Not Applicable

Definition at line 47 of file default\_cfe\_evs\_topicids.h.

**12.98.2.3 CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_MSG** #define CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_MSG 8  
 Definition at line 48 of file default\_cfe\_evs\_topicids.h.

**12.98.2.4 CFE\_MISSION\_EVS\_SEND\_HK\_MSG** #define CFE\_MISSION\_EVS\_SEND\_HK\_MSG 9  
 Definition at line 36 of file default\_cfe\_evs\_topicids.h.

**12.98.2.5 CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_MSG** #define CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_MSG 9  
 SG 9  
 Definition at line 49 of file default\_cfe\_evs\_topicids.h.

## 12.99 cfe/modules/evs/fsw/inc/cfe\_evs\_eventids.h File Reference

**Macros****EVS event IDs**

- #define **CFE\_EVS\_NOOP\_EID** 0  
*EVS No-op Command Success Event ID.*
- #define **CFE\_EVS\_STARTUP\_EID** 1  
*EVS Initialization Event ID.*
- #define **CFE\_EVS\_ERR\_WRLOGFILE\_EID** 2  
*EVS Write Event Log Command File Write Entry Failed Event ID.*
- #define **CFE\_EVS\_ERR\_CRLOGFILE\_EID** 3  
*EVS Write Event Log Command Filename Parse or File Create Failed Event ID.*
- #define **CFE\_EVS\_ERR\_MSGID\_EID** 5  
*EVS Invalid Message ID Received Event ID.*
- #define **CFE\_EVS\_ERR\_EVTIDNOREGS\_EID** 6  
*EVS Command Event Not Registered For Filtering Event ID.*
- #define **CFE\_EVS\_ERR\_APPNOREGS\_EID** 7  
*EVS Command Application Not Registered With EVS Event ID.*
- #define **CFE\_EVS\_ERR\_ILLAPPIDRANGE\_EID** 8  
*EVS Command Get Application Data Failure Event ID.*
- #define **CFE\_EVS\_ERR\_NOAPPIDFOUND\_EID** 9  
*EVS Command Get Application ID Failure Event ID.*
- #define **CFE\_EVS\_ERR\_ILLEGALFMTMOD\_EID** 10  
*EVS Set Event Format Command Invalid Format Event ID.*
- #define **CFE\_EVS\_ERR\_MAXREGSFILTER\_EID** 11  
*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_WRITE_HEADER_ERR_EID` 14  
*EVS Write File Header to Log File Failure 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_ENAAPPEV_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.*
- #define `CFE_EVS_SQUELCHED_ERR_EID` 44  
*EVS Events Squelched Error Event ID.*

### 12.99.1 Detailed Description

cFE Event Services Event IDs

### 12.99.2 Macro Definition Documentation

**12.99.2.1 CFE\_EVS\_ADDFILTER\_EID** #define CFE\_EVS\_ADDFILTER\_EID 30  
EVS Add Event Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Add Event Filter Command](#) success.  
Definition at line 367 of file cfe\_evs\_eventids.h.

**12.99.2.2 CFE\_EVS\_DELFILTER\_EID** #define CFE\_EVS\_DELFILTER\_EID 31  
EVS Delete Event Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Delete Event Filter Command](#) success.  
Definition at line 378 of file cfe\_evs\_eventids.h.

**12.99.2.3 CFE\_EVS\_DISAPPENNTTYPE\_EID** #define CFE\_EVS\_DISAPPENNTTYPE\_EID 24  
EVS Disable App Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable App Event Type Command](#) success.  
Definition at line 301 of file cfe\_evs\_eventids.h.

**12.99.2.4 CFE\_EVS\_DISAPPEVT\_EID** #define CFE\_EVS\_DISAPPEVT\_EID 26  
EVS Disable App Events Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable App Events Command](#) success.  
Definition at line 323 of file cfe\_evs\_eventids.h.

**12.99.2.5 CFE\_EVS\_DISEVTTYPE\_EID** #define CFE\_EVS\_DISEVTTYPE\_EID 21  
EVS Disable Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable Event Type Command](#) success.  
Definition at line 268 of file cfe\_evs\_eventids.h.

**12.99.2.6 CFE\_EVS\_DISPORT\_EID** #define CFE\_EVS\_DISPORT\_EID 19  
EVS Disable Ports Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable Ports Command](#) success.  
Definition at line 246 of file cfe\_evs\_eventids.h.

**12.99.2.7 CFE\_EVS\_ENAAPPEVT\_EID** #define CFE\_EVS\_ENAAPPEVT\_EID 25  
EVS Enable App Events Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable App Events Command](#) success.  
Definition at line 312 of file cfe\_evs\_eventids.h.

**12.99.2.8 CFE\_EVS\_ENAAPPEVTTYPE\_EID** #define CFE\_EVS\_ENAAPPEVTTYPE\_EID 23  
EVS Enable App Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable App Event Type Command](#) success.  
Definition at line 290 of file cfe\_evs\_eventids.h.

**12.99.2.9 CFE\_EVS\_ENAEVTTYPE\_EID** #define CFE\_EVS\_ENAEVTTYPE\_EID 20  
EVS Enable Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable Event Type Command](#) success.  
Definition at line 257 of file cfe\_evs\_eventids.h.

**12.99.2.10 CFE\_EVS\_ENAPORT\_EID** #define CFE\_EVS\_ENAPORT\_EID 18  
EVS Enable Ports Command Success Event ID.

Type: DEBUG

Cause:

[EVS Enable Ports Command](#) success.  
Definition at line 235 of file cfe\_evs\_eventids.h.

**12.99.2.11 CFE\_EVS\_ERR\_APPNOREGS\_EID** #define CFE\_EVS\_ERR\_APPNOREGS\_EID 7  
EVS Command Application Not Registered With EVS Event ID.

Type: ERROR

Cause:

An EVS command handler failure due to the referenced application not being registered with EVS. OVERLOADED  
Definition at line 110 of file cfe\_evs\_eventids.h.

**12.99.2.12 CFE\_EVS\_ERR\_CC\_EID** #define CFE\_EVS\_ERR\_CC\_EID 15  
EVS Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE\\_EVS\\_CMD\\_MID](#) received on the EVS message pipe.  
Definition at line 202 of file cfe\_evs\_eventids.h.

**12.99.2.13 CFE\_EVS\_ERR\_CRDATFILE\_EID** #define CFE\_EVS\_ERR\_CRDATFILE\_EID 13  
EVS Write Application Data Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[Write Application Data Command](#) failed to parse the filename or open/create the file. OVERLOADED  
Definition at line 180 of file cfe\_evs\_eventids.h.

**12.99.2.14 CFE\_EVS\_ERR\_CRLOGFILE\_EID** #define CFE\_EVS\_ERR\_CRLOGFILE\_EID 3  
EVS Write Event Log Command Filename Parse or File Create Failed Event ID.

Type: ERROR

Cause:

[EVS Write Event Log Command](#) failure parsing the file name or during open/creation of the file. OVERLOADED  
Definition at line 77 of file cfe\_evs\_eventids.h.

**12.99.2.15 CFE\_EVS\_ERR\_EVTIDNOREGS\_EID** #define CFE\_EVS\_ERR\_EVTIDNOREGS\_EID 6  
EVS Command Event Not Registered For Filtering Event ID.

Type: ERROR

Cause:

An EVS command handler failure due to the event not being registered for filtering. OVERLOADED  
Definition at line 99 of file cfe\_evs\_eventids.h.

**12.99.2.16 CFE\_EVS\_ERR\_ILLAPPIDRANGE\_EID** #define CFE\_EVS\_ERR\_ILLAPPIDRANGE\_EID 8  
EVS Command Get Application Data Failure Event ID.

Type: ERROR

Cause:

An EVS command handler failure retrieving the application data. OVERLOADED  
Definition at line 121 of file cfe\_evs\_eventids.h.

**12.99.2.17 CFE\_EVS\_ERR\_ILLEGALFMTMOD\_EID** #define CFE\_EVS\_ERR\_ILLEGALFMTMOD\_EID 10  
EVS Set Event Format Command Invalid Format Event ID.

Type: ERROR

Cause:

[EVS Set Event Format Command](#) failure due to invalid format argument.  
Definition at line 144 of file cfe\_evs\_eventids.h.

**12.99.2.18 CFE\_EVS\_ERR\_INVALID\_BITMASK\_EID** #define CFE\_EVS\_ERR\_INVALID\_BITMASK\_EID 40  
EVS Port Or Event Type Bitmask Invalid Event ID.

Type: ERROR

Cause:

Invalid bitmask for EVS port or event type. OVERLOADED  
Definition at line 446 of file cfe\_evs\_eventids.h.

**12.99.2.19 CFE\_EVS\_ERR\_LOGMODE\_EID** #define CFE\_EVS\_ERR\_LOGMODE\_EID 39  
EVS Set Log Mode Command Invalid Mode Event ID.

Type: ERROR

Cause:

[EVS Set Log Mode Command](#) failure due to invalid log mode.  
Definition at line 435 of file cfe\_evs\_eventids.h.

**12.99.2.20 CFE\_EVS\_ERR\_MAXREGSFILTER\_EID** #define CFE\_EVS\_ERR\_MAXREGSFILTER\_EID 11  
EVS Add Filter Command Max Filters Exceeded Event ID.

Type: ERROR

Cause:

[EVS Add Filter Command](#) failure due to exceeding the maximum number of filters.  
Definition at line 156 of file cfe\_evs\_eventids.h.

**12.99.2.21 CFE\_EVS\_ERR\_MSGID\_EID** #define CFE\_EVS\_ERR\_MSGID\_EID 5  
EVS Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the EVS message pipe.  
Definition at line 88 of file cfe\_evs\_eventids.h.

**12.99.2.22 CFE\_EVS\_ERR\_NOAPPIDFOUND\_EID** #define CFE\_EVS\_ERR\_NOAPPIDFOUND\_EID 9  
EVS Command Get Application ID Failure Event ID.

Type: ERROR

Cause:

An EVS command handler failure retrieving the application ID. OVERLOADED  
Definition at line 132 of file cfe\_evs\_eventids.h.

**12.99.2.23 CFE\_EVS\_ERR\_UNREGISTERED\_EVS\_APP** #define CFE\_EVS\_ERR\_UNREGISTERED\_EVS\_APP 41  
EVS Send Event API App Not Registered With EVS Event ID.

Type: ERROR

Cause:

An EVS Send Event API called for application not registered with EVS.  
Definition at line 457 of file cfe\_evs\_eventids.h.

**12.99.2.24 CFE\_EVS\_ERR\_WRDATFILE\_EID** #define CFE\_EVS\_ERR\_WRDATFILE\_EID 12  
EVS Write Application Data Command Write Data Failure Event ID.

Type: ERROR

Cause:

[Write Application Data Command](#) failure to write application EVS data.  
Definition at line 168 of file cfe\_evs\_eventids.h.

**12.99.2.25 CFE\_EVS\_ERR\_WRLlogfile\_EID** #define CFE\_EVS\_ERR\_WRLlogfile\_EID 2  
EVS Write Event Log Command File Write Entry Failed Event ID.

Type: ERROR

Cause:

[EVS Write Event Log Command](#) failure writing data to the file.  
Definition at line 65 of file cfe\_evs\_eventids.h.

**12.99.2.26 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.  
Definition at line 412 of file cfe\_evs\_eventids.h.

**12.99.2.27 CFE\_EVS\_FILTER\_MAX\_EID** #define CFE\_EVS\_FILTER\_MAX\_EID 42  
EVS Filter Max Count Reached Event ID.

Type: INFORMATIONAL

Cause:

Filter count for the event reached CFE\_EVS\_MAX\_FILTER\_COUNT and is latched until filter is reset.  
Definition at line 468 of file cfe\_evs\_eventids.h.

**12.99.2.28 CFE\_EVS\_LEN\_ERR\_EID** #define CFE\_EVS\_LEN\_ERR\_EID 43  
EVS Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE\\_EVS\\_CMD\\_MID](#) received on the EVS message pipe.  
Definition at line 479 of file cfe\_evs\_eventids.h.

**12.99.2.29 CFE\_EVS\_LOGMODE\_EID** #define CFE\_EVS\_LOGMODE\_EID 38  
EVS Set Log Mode Command Success Event ID.

Type: DEBUG

Cause:

[EVS Set Log Mode Command](#) success.  
Definition at line 423 of file cfe\_evs\_eventids.h.

**12.99.2.30 CFE\_EVS\_NOOP\_EID** #define CFE\_EVS\_NOOP\_EID 0  
EVS No-op Command Success Event ID.

Type: INFORMATION

Cause:

[EVS NO-OP command](#) success.  
Definition at line 42 of file cfe\_evs\_eventids.h.

**12.99.2.31 CFE\_EVS\_RSTALLFILTER\_EID** #define CFE\_EVS\_RSTALLFILTER\_EID 29  
EVS Reset All Filters Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset All Filters Command](#) success.  
Definition at line 356 of file cfe\_evs\_eventids.h.

**12.99.2.32 CFE\_EVS\_RSTCNT\_EID** #define CFE\_EVS\_RSTCNT\_EID 16  
EVS Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset Counters Command](#) success.  
Definition at line 213 of file cfe\_evs\_eventids.h.

**12.99.2.33 CFE\_EVS\_RSTEVTCNT\_EID** #define CFE\_EVS\_RSTEVTCNT\_EID 27  
EVS Reset App Event Counter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset App Event Counter Command](#) success.  
Definition at line 334 of file cfe\_evs\_eventids.h.

**12.99.2.34 CFE\_EVS\_RSTFILTER\_EID** #define CFE\_EVS\_RSTFILTER\_EID 28  
EVS Reset App Event Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Reset App Event Filter Command](#) success.  
Definition at line 345 of file cfe\_evs\_eventids.h.

**12.99.2.35 CFE\_EVS\_SETEVTFMTMOD\_EID** #define CFE\_EVS\_SETEVTFMTMOD\_EID 22  
EVS Set Event Format Mode Command Success Event ID.

Type: DEBUG

Cause:

[EVS Set Event Format Mode Command](#) success.  
Definition at line 279 of file cfe\_evs\_eventids.h.

**12.99.2.36 CFE\_EVS\_SETFILTERMSK\_EID** #define CFE\_EVS\_SETFILTERMSK\_EID 17  
EVS Set Filter Command Success Event ID.

Type: DEBUG

Cause:

[EVS Set Filter Command](#) success.

Definition at line 224 of file cfe\_evs\_eventids.h.

**12.99.2.37 CFE\_EVS\_SQUELCHED\_ERR\_EID** #define CFE\_EVS\_SQUELCHED\_ERR\_EID 44  
EVS Events Squelched Error Event ID.

Type: ERROR

Cause:

Events generated in app at a rate in excess of [CFE\\_PLATFORM\\_EVS\\_MAX\\_APP\\_EVENT\\_BURST](#) in one moment or [CFE\\_PLATFORM\\_EVS\\_APP\\_EVENTS\\_PER\\_SEC](#) sustained

Definition at line 492 of file cfe\_evs\_eventids.h.

**12.99.2.38 CFE\_EVS\_STARTUP\_EID** #define CFE\_EVS\_STARTUP\_EID 1  
EVS Initialization Event ID.

Type: INFORMATION

Cause:

Event Services Task initialization complete.

Definition at line 53 of file cfe\_evs\_eventids.h.

**12.99.2.39 CFE\_EVS\_WRDAT\_EID** #define CFE\_EVS\_WRDAT\_EID 32  
EVS Write Application Data Command Success Event ID.

Type: DEBUG

Cause:

[EVS Write Application Data Command](#) success.

Definition at line 389 of file cfe\_evs\_eventids.h.

**12.99.2.40 CFE\_EVS\_WRITE\_HEADER\_ERR\_EID** #define CFE\_EVS\_WRITE\_HEADER\_ERR\_EID 14  
EVS Write File Header to Log File Failure Event ID.

Type: ERROR

Cause:

Bytes written during Write File Header to Log File was not equal to the expected header size.  
Definition at line 191 of file cfe\_evs\_eventids.h.

**12.99.2.41 CFE\_EVS\_WRLOG\_EID** #define CFE\_EVS\_WRLOG\_EID 33  
EVS Write Event Log Command Success Event ID.

Type: DEBUG

Cause:

EVS Write Event Log Command success.  
Definition at line 400 of file cfe\_evs\_eventids.h.

## 12.100 cfe/modules/fs/config/default\_cfe\_fs\_extern\_typedefs.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_fs_filedef.h"
```

### 12.100.1 Detailed Description

Declarations and prototypes for cfe\_fs\_extern\_typedefs module

## 12.101 cfe/modules/fs/config/default\_cfe\_fs\_filedef.h File Reference

```
#include "common_types.h"
#include "cfe_fs_interface_cfg.h"
```

### Data Structures

- struct [CFE\\_FS\\_Header](#)  
*Standard cFE File header structure definition.*

### Typedefs

- typedef uint32 [CFE\\_FS\\_SubType\\_Enum\\_t](#)  
*Content descriptor for File Headers.*
- typedef struct [CFE\\_FS\\_Header](#) [CFE\\_FS\\_Header\\_t](#)  
*Standard cFE File header structure definition.*

## Enumerations

- enum `CFE_FS_SubType` {  
    `CFE_FS_SubType_ES_ERLOG` = 1, `CFE_FS_SubType_ES_SYSLOG` = 2, `CFE_FS_SubType_ES_QUERYALL` = 3, `CFE_FS_SubType_ES_PERFDATA` = 4,  
    `CFE_FS_SubType_ES_CDS_REG` = 6, `CFE_FS_SubType_TBL_REG` = 9, `CFE_FS_SubType_TBL_IMG` = 8,  
    `CFE_FS_SubType_EVS_APPDATA` = 15,  
    `CFE_FS_SubType_EVS_EVENTLOG` = 16, `CFE_FS_SubType_SB_PIPE DATA` = 20, `CFE_FS_SubType_SB_ROUTEDATA` = 21, `CFE_FS_SubType_SB_MAPDATA` = 22,  
    `CFE_FS_SubType_ES_QUERYALLTASKS` = 23 }

*File subtypes used within cFE.*

### 12.101.1 Detailed Description

Declarations and prototypes for `cfe_fs_extern_typedefs` module

### 12.101.2 Typedef Documentation

#### 12.101.2.1 `CFE_FS_Header_t` `typedef struct CFE_FS_Header CFE_FS_Header_t`

Standard cFE File header structure definition.

#### 12.101.2.2 `CFE_FS_SubType_Enum_t` `typedef uint32 CFE_FS_SubType_Enum_t`

Content descriptor for File Headers.

See also

enum `CFE_FS_SubType`

Definition at line 176 of file `default_cfe_fs_filedef.h`.

### 12.101.3 Enumeration Type Documentation

#### 12.101.3.1 `CFE_FS_SubType` `enum CFE_FS_SubType`

File subtypes used within cFE.

This defines all the file subtypes used by cFE. Note apps can extend as needed but need to avoid conflicts (app context not currently included in the file header).

Enumerator

<code>CFE_FS_SubType_ES_ERLOG</code>	Executive Services Exception/Reset Log Type. Executive Services Exception/Reset Log File which is generated in response to a <code>\$sc_\$cpu_ES_WriteERLog2File</code> command.
<code>CFE_FS_SubType_ES_SYSLOG</code>	Executive Services System Log Type. Executive Services System Log File which is generated in response to a <code>\$sc_\$cpu_ES_WriteSysLog2File</code> command.
<code>CFE_FS_SubType_ES_QUERYALL</code>	Executive Services Information on All Applications File. Executive Services Information on All Applications File which is generated in response to a <code>\$sc_\$cpu_ES_WriteAppInfo2File</code> command.

## Enumerator

CFE_FS_SubType_ES_PERFDATA	Executive Services Performance Data File. Executive Services Performance Analyzer Data File which is generated in response to a <a href="#">\$sc_\$cpu_ES_StopLData</a> 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 <a href="#">\$sc_\$cpu_ES_WriteCDS2File</a> command.
CFE_FS_SubType_TBL_REG	Table Services Registry Dump File. Table Services Registry Dump File which is generated in response to a <a href="#">\$sc_\$cpu_TBL_WriteReg2File</a> 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 <a href="#">\$sc_\$cpu_TBL_DUMP</a> command.
CFE_FS_SubType_EVS_APPDATA	Event Services Application Data Dump File. Event Services Application Data Dump File which is generated in response to a <a href="#">\$sc_\$cpu_EVS_WriteAppData2File</a> 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 <a href="#">\$sc_\$cpu_EVS_WriteLog2File</a> command.
CFE_FS_SubType_SB_PIPE DATA	Software Bus Pipe Data Dump File. Software Bus Pipe Data Dump File which is generated in response to a <a href="#">\$sc_\$cpu_SB_WritePipe2File</a> 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 <a href="#">\$sc_\$cpu_SB_WriteRouting2File</a> 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 <a href="#">\$sc_\$cpu_SB_WriteMap2File</a> 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 <a href="#">\$sc_\$cpu_ES_WriteTaskInfo2File</a> command.

Definition at line 39 of file default\_cfe\_fs\_filedef.h.

## 12.102 cfe/modules/fs/config/default\_cfe\_fs\_interface\_cfg.h File Reference

### Macros

- #define CFE\_FS\_HDR\_DESC\_MAX\_LEN 32  
*Max length of description field in a standard cFE File Header.*
- #define CFE\_FS\_FILE\_CONTENT\_ID 0x63464531  
*Magic Number for cFE compliant files (= 'cFE1')*

### 12.102.1 Detailed Description

Declarations and prototypes for cfe\_fs\_extern\_typedefs module

### 12.102.2 Macro Definition Documentation

**12.102.2.1 CFE\_FS\_FILE\_CONTENT\_ID** #define CFE\_FS\_FILE\_CONTENT\_ID 0x63464531  
Magic Number for cFE compliant files (= 'cFE1')  
Definition at line 39 of file default\_cfe\_fs\_interface\_cfg.h.

**12.102.2.2 CFE\_FS\_HDR\_DESC\_MAX\_LEN** #define CFE\_FS\_HDR\_DESC\_MAX\_LEN 32  
Max length of description field in a standard cFE File Header.  
Definition at line 37 of file default\_cfe\_fs\_interface\_cfg.h.

## 12.103 cfe/modules/fs/config/default\_cfe\_fs\_mission\_cfg.h File Reference

```
#include "cfe_fs_interface_cfg.h"
```

### 12.103.1 Detailed Description

CFE File Services (CFE\_FS) Application Mission Configuration Header File  
This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.104 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.*

### 12.104.1 Detailed Description

Define CCSDS packet header types

- Avoid direct access for portability, use APIs
- Used to construct message structures

### 12.104.2 Typedef Documentation

**12.104.2.1 CCSDS\_ExtendedHeader\_t** `typedef struct CCSDS_ExtendedHeader CCSDS_ExtendedHeader_t`  
 CCSDS packet extended header.

**12.104.2.2 CCSDS\_PrimaryHeader\_t** `typedef struct CCSDS_PrimaryHeader CCSDS_PrimaryHeader_t`  
 CCSDS packet primary header.

## 12.105 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_RESOURCE_BASE_OFFSET` = CFE\_RESOURCEID\_MAKE\_BASE(CFE\_RESOURCEID\_SB\_PIPEID\_RESOURCE\_BASE\_OFFSET), `CFE_CONFIGID_BASE` = CFE\_RESOURCEID\_MAKE\_BASE(CFE\_RESOURCEID\_CONFIGID\_BASE\_OFFSET) }

### 12.105.1 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.

## 12.106 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.*

### 12.106.1 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.

### 12.106.2 Macro Definition Documentation

**12.106.2.1 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.

**12.106.2.2 CFE\_RESOURCEID\_MAX** `#define CFE_RESOURCEID_MAX OS_OBJECT_INDEX_MASK`

Definition at line 65 of file cfe\_resourceid\_basevalue.h.

**12.106.2.3 CFE\_RESOURCEID\_SHIFT** `#define CFE_RESOURCEID_SHIFT OS_OBJECT_TYPE_SHIFT`

Definition at line 64 of file cfe\_resourceid\_basevalue.h.

## 12.107 cfe/modules/sb/config/default\_cfe\_sb\_extern\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_resourceid_typedef.h"
```

### Data Structures

- struct [CFE\\_SB\\_MsgId\\_t](#)  
*CFE\_SB\_MsgId\_t type definition.*
- struct [CFE\\_SB\\_Qos\\_t](#)  
*Quality Of Service Type Definition.*

### Macros

- `#define CFE_SB_SUB_ENTRIES_PER_PKT 20`  
*Configuration parameter used by SBN App.*

### Typedefs

- `typedef uint8 CFE_SB_QosPriority_Enum_t`  
*Selects the priority level for message routing.*
- `typedef uint8 CFE_SB_QosReliability_Enum_t`

Selects the reliability level for message routing.

- `typedef uint16 CFE_SB_Routeld_Atom_t`  
*An integer type that should be used for indexing into the Routing Table.*
- `typedef uint32 CFE_SB_MsgId_Atom_t`  
*CFE\_SB\_MsgId\_Atom\_t primitive type definition.*
- `typedef CFE_RESOURCEID_BASE_TYPE CFE_SB_Pipeld_t`  
*CFE\_SB\_Pipeld\_t to primitive type definition.*

## Enumerations

- `enum CFE_SB_QosPriority { CFE_SB_QosPriority_LOW = 0, CFE_SB_QosPriority_HIGH = 1 }`  
*Label definitions associated with CFE\_SB\_QosPriority\_Enum\_t.*
- `enum CFE_SB_QosReliability { CFE_SB_QosReliability_LOW = 0, CFE_SB_QosReliability_HIGH = 1 }`  
*Label definitions associated with CFE\_SB\_QosReliability\_Enum\_t.*

### 12.107.1 Detailed Description

Declarations and prototypes for cfe\_sb\_extern\_typedefs module

### 12.107.2 Macro Definition Documentation

#### 12.107.2.1 CFE\_SB\_SUB\_ENTRIES\_PER\_PKT #define CFE\_SB\_SUB\_ENTRIES\_PER\_PKT 20

Configuration parameter used by SBN App.

Definition at line 32 of file default\_cfe\_sb\_extern\_typedefs.h.

### 12.107.3 Typedef Documentation

#### 12.107.3.1 CFE\_SB\_MsgId\_Atom\_t `typedef uint32 CFE_SB_MsgId_Atom_t`

CFE\_SB\_MsgId\_Atom\_t primitive type definition.

This is an integer type capable of holding any Message ID value Note: This value is limited via `CFE_PLATFORM_SB_HIGHEST_VALID_MSG_ID`.  
Definition at line 91 of file default\_cfe\_sb\_extern\_typedefs.h.

#### 12.107.3.2 CFE\_SB\_Pipeld\_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_SB_PipeId_t`

CFE\_SB\_Pipeld\_t to primitive type definition.

Software Bus pipe identifier used in many SB APIs, as well as SB Telemetry messages and data files.

Definition at line 114 of file default\_cfe\_sb\_extern\_typedefs.h.

#### 12.107.3.3 CFE\_SB\_QosPriority\_Enum\_t `typedef uint8 CFE_SB_QosPriority_Enum_t`

Selects the priority level for message routing.

See also

`enum CFE_SB_QosPriority`

Definition at line 55 of file default\_cfe\_sb\_extern\_typedefs.h.

**12.107.3.4 CFE\_SB\_QosReliability\_Enum\_t** `typedef uint8 CFE_SB_QosReliability_Enum_t`  
Selects the reliability level for message routing.

See also

enum [CFE\\_SB\\_QosReliability](#)

Definition at line 78 of file default\_cfe\_sb\_extern\_typedefs.h.

**12.107.3.5 CFE\_SB\_RoutId\_Atom\_t** `typedef uint16 CFE_SB_RouteId_Atom_t`

An integer type that should be used for indexing into the Routing Table.

Definition at line 83 of file default\_cfe\_sb\_extern\_typedefs.h.

## 12.107.4 Enumeration Type Documentation

**12.107.4.1 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 37 of file default\_cfe\_sb\_extern\_typedefs.h.

**12.107.4.2 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 60 of file default\_cfe\_sb\_extern\_typedefs.h.

## 12.108 cfe/modules/sb/config/default\_cfe\_sb\_fcncodes.h File Reference

### Macros

- `#define CFE_SB_NOOP_CC 0`
- `#define CFE_SB_RESET_COUNTERS_CC 1`
- `#define CFE_SB_SEND_SB_STATS_CC 2`
- `#define CFE_SB_WRITE_ROUTING_INFO_CC 3`
- `#define CFE_SB_ENABLE_ROUTE_CC 4`
- `#define CFE_SB_DISABLE_ROUTE_CC 5`
- `#define CFE_SB_WRITE_PIPE_INFO_CC 7`
- `#define CFE_SB_WRITE_MAP_INFO_CC 8`
- `#define CFE_SB_ENABLE_SUB_REPORTING_CC 9`
- `#define CFE_SB_DISABLE_SUB_REPORTING_CC 10`
- `#define CFE_SB_SEND_PREV_SUBS_CC 11`

### 12.108.1 Detailed Description

Specification for the CFE Event Services (CFE\_SB) command function codes

#### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

### 12.108.2 Macro Definition Documentation

#### 12.108.2.1 CFE\_SB\_DISABLE\_ROUTE\_CC #define CFE\_SB\_DISABLE\_ROUTE\_CC 5

**Name** Disable Software Bus Route

#### Description

This command will disable a particular destination. The destination is specified in terms of MsgID and PipeID. The MsgId and PipeID are parameters in the command. All destinations are enabled by default.

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_DisRoute

#### Command Structure

[CFE\\_SB\\_DisableRouteCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_SB\\_CMDPC](#) - command execution counter will increment
- View routing information [CFE\\_SB\\_WRITE\\_ROUTING\\_INFO\\_CC](#) to verify enable/disable state change
- The [CFE\\_SB\\_DSBL RTE2 EID](#) debug event message will be generated
- Destination will stop receiving messages

#### Error Conditions

This command may fail for the following reason(s):

- the MsgId or PipeId parameters do not pass validation
- the destination does not exist.

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_SB\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases. See [CFE\\_SB\\_DSBL RTE1 EID](#) or [CFE\\_SB\\_DSBL RTE3 EID](#)

#### Criticality

This command is not intended to be used in nominal conditions. It is possible to get into a state where a destination cannot be re-enabled without resetting the processor. For instance, sending this command with [CFE\\_SB\\_CMD\\_MID](#) and the SB\_Cmd\_Pipe would inhibit any ground commanding to the software bus until the processor was reset. There are similar problems that may occur when using this command.

Definition at line 271 of file default\_cfe\_sb\_fcncodes.h.

**12.108.2.2 CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC** #define CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC 10**Name** Disable Subscription Reporting Command**Description**

This command will disable subscription reporting and is intended to be used only by the CFS SBN (Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When subscription reporting is enabled, SB will generate and send a software bus packet for each subscription received. The software bus packet that is sent contains the information received in the subscription API. This subscription report is needed by SBN if offboard routing is required.

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_DisSubRptg**Command Structure**[CFE\\_SB\\_DisableSubReportingCmd\\_t](#)**Command Verification**

Successful execution of this command will result in the suppression of packets (with the [CFE\\_SB\\_ONESUB\\_TLM\\_MID](#) MsgId) for each subscription received by SB through the subscription APIs.

**Error Conditions**

None

**Criticality**

None

**See also**[CFE\\_SB\\_SingleSubscriptionTlm\\_t](#), [CFE\\_SB\\_ENABLE\\_SUB\\_REPORTING\\_CC](#), [CFE\\_SB\\_SEND\\_PREV\\_SUBS\\_CC](#)

Definition at line 428 of file default\_cfe\_sb\_fcncodes.h.

**12.108.2.3 CFE\_SB\_ENABLE\_ROUTE\_CC** #define CFE\_SB\_ENABLE\_ROUTE\_CC 4**Name** Enable Software Bus Route**Description**

This command will enable a particular destination. The destination is specified in terms of MsgID and PipeID. The MsgId and PipeID are parameters in the command. All destinations are enabled by default. This command is needed only after a [CFE\\_SB\\_DISABLE\\_ROUTE\\_CC](#) command is used.

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_EnaRoute**Command Structure**[CFE\\_SB\\_EnableRouteCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_SB_CMDPC` - command execution counter will increment
- View routing information `CFE_SB_WRITE_ROUTING_INFO_CC` to verify enable/disable state change
- The `CFE_SB_ENBL RTE2 EID` debug event message will be generated
- Destination will begin receiving messages

### Error Conditions

This command may fail for the following reason(s):

- the MsgId or PipeId parameters do not pass validation
- the destination does not exist.

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_SB_CMDEC` - command error counter will increment
- A command specific error event message is issued for all error cases. See `CFE_SB_ENBL RTE1 EID` or `CFE_SB_ENBL RTE3 EID`

### Criticality

This command is not inherently dangerous.

Definition at line 230 of file default\_cfe\_sb\_fcncodes.h.

#### 12.108.2.4 CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC #define CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC 9

##### Name Enable Subscription Reporting Command

##### Description

This command will enable subscription reporting and is intended to be used only by the CFS SBN (Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When subscription reporting is enabled, SB will generate and send a software bus packet for each subscription received. The software bus packet that is sent contains the information received in the subscription API. This subscription report is needed by SBN if offboard routing is required.

##### Command Mnemonic(s) \$sc\_\$cpu\_SB\_EnaSubRptg

##### Command Structure

`CFE_SB_EnableSubReportingCmd_t`

##### Command Verification

Successful execution of this command will result in the sending of a packet (with the `CFE_SB_ONESUB_TLM_MID` MsgId) for each subscription received by SB through the subscription APIs.

##### Error Conditions

None

**Criticality**

None

**See also**

[CFE\\_SB\\_SingleSubscriptionTlm\\_t](#), [CFE\\_SB\\_DISABLE\\_SUB\\_REPORTING\\_CC](#), [CFE\\_SB\\_SEND\\_PREV\\_SUBS\\_CC](#)

Definition at line 395 of file default\_cfe\_sb\_fcncodes.h.

**12.108.2.5 CFE\_SB\_NOOP\_CC #define CFE\_SB\_NOOP\_CC 0**

**Name** Software Bus No-Op

**Description**

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Software Bus task.

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_NOOP

**Command Structure**

[CFE\\_SB\\_NoopCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_SB\_CMDPC** - command execution counter will increment
- The [CFE\\_SB\\_CMD0\\_RCVD\\_EID](#) informational event message will be generated

**Error Conditions**

There are no error conditions for this command. If the Software Bus receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

**Criticality**

None

**See also**

Definition at line 66 of file default\_cfe\_sb\_fcncodes.h.

**12.108.2.6 CFE\_SB\_RESET\_COUNTERS\_CC #define CFE\_SB\_RESET\_COUNTERS\_CC 1**

**Name** Software Bus Reset Counters

## Description

This command resets the following counters within the Software Bus housekeeping telemetry:

- Command Execution Counter (\$sc\_\$cpu\_SB\_CMDPC)
- Command Error Counter (\$sc\_\$cpu\_SB\_CMDEC)
- No Subscribers Counter (\$sc\_\$cpu\_SB\_NoSubEC)
- Duplicate Subscriptions Counter (\$sc\_\$cpu\_SB\_DupSubCnt)
- Msg Send Error Counter (\$sc\_\$cpu\_SB\_MsgSndEC)
- Msg Receive Error Counter (\$sc\_\$cpu\_SB\_MsgRecEC)
- Internal Error Counter (\$sc\_\$cpu\_SB\_InternalEC)
- Create Pipe Error Counter (\$sc\_\$cpu\_SB\_NewPipeEC)
- Subscribe Error Counter (\$sc\_\$cpu\_SB\_SubscrEC)
- Pipe Overflow Error Counter (\$sc\_\$cpu\_SB\_PipeOvrEC)
- Msg Limit Error Counter (\$sc\_\$cpu\_SB\_MsgLimEC)

## Command Mnemonic(s) \$sc\_\$cpu\_SB\_ResetCtrs

### Command Structure

[CFE\\_SB\\_ResetCountersCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_SB\_CMDPC** - command execution counter will be reset to 0
- All other counters listed in description will be reset to 0
- The [CFE\\_SB\\_CMD1\\_RCVD\\_EID](#) informational event message will be generated

### Error Conditions

There are no error conditions for this command. If the Software Bus receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

### Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

### See also

Definition at line 113 of file default\_cfe\_sb\_fcncodes.h.

**12.108.2.7 CFE\_SB\_SEND\_PREV\_SUBS\_CC** #define CFE\_SB\_SEND\_PREV\_SUBS\_CC 11

**Name** Send Previous Subscriptions Command

This command generates a series of packets that contain information

regarding all subscriptions previously received by SB. This command is intended to be used only by the CFS SBN(Software Bus Networking) Application. It is not intended to be sent from the ground or used by operations. When this command is received the software bus will generate and send a series of packets containing information about all subscription previously received.

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_SendPrevSubs

Command Structure

[CFE\\_SB\\_SendPrevSubsCmd\\_t](#)

Command Verification

Successful execution of this command will result in a series of packets (with the [CFE\\_SB\\_ALLSUBS\\_TLM\\_MID](#) MsgId) being sent on the software bus.

Error Conditions

None

Criticality

None

See also

[CFE\\_SB\\_AllSubscriptionsTlm\\_t](#), [CFE\\_SB\\_ENABLE\\_SUB\\_REPORTING\\_CC](#), [CFE\\_SB\\_DISABLE\\_SUB\\_REPORTING\\_CC](#)

Definition at line 460 of file default\_cfe\_sb\_fcncodes.h.

**12.108.2.8 CFE\_SB\_SEND\_SB\_STATS\_CC** #define CFE\_SB\_SEND\_SB\_STATS\_CC 2

**Name** Send Software Bus Statistics

Description

This command will cause the SB task to send a statistics packet containing current utilization figures and high water marks which may be useful for checking the margin of the SB platform configuration settings.

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_DumpStats

Command Structure

[CFE\\_SB\\_SendSbStatsCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_SB\_CMDPC** - command execution counter will increment
- Receipt of statistics packet with MsgId [CFE\\_SB\\_STATS\\_TLM\\_MID](#)
- The [CFE\\_SB SND STATS EID](#) debug event message will be generated

### Error Conditions

There are no error conditions for this command. If the Software Bus receives the command, the debug event is sent and the counter is incremented unconditionally.

### Criticality

This command is not inherently dangerous. It will create and send a message on the software bus. If performed repeatedly, it is possible that receiver pipes may overflow.

### See also

Definition at line 147 of file default\_cfe\_sb\_fcncodes.h.

## 12.108.2.9 CFE\_SB\_WRITE\_MAP\_INFO\_CC #define CFE\_SB\_WRITE\_MAP\_INFO\_CC 8

**Name** Write Map Info to a File

This command will create a file containing the software bus message

map information. The message map is a lookup table (an array of uint16s) that allows fast access to the correct routing table element during a software bus send operation. This is diagnostic information that may be needed due to the dynamic nature of the cFE software bus. An absolute path and filename may be specified in the command. If this command field contains an empty string (NULL terminator as the first character) the default file path and name is used. The default file path and name is defined in the platform configuration file as [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_MAP\\_FILENAME](#).

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_WriteMap2File

### Command Structure

[CFE\\_SB\\_WriteMapInfoCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_SB\_CMDPC** - command execution counter will increment. NOTE: the command counter is incremented when the request is accepted, before writing the file, which is performed as a background task.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_MAP\\_FILENAME](#) configuration parameter) will be updated with the latest information.
- The [CFE\\_SB SND RTG EID](#) debug event message will be generated

### Error Conditions

This command may fail for the following reason(s):

- A previous request to write a software bus information file has not yet completed
- The specified FileName cannot be parsed

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_SB_CMDEC` - command error counter will increment
- A command specific error event message is issued for all error cases. See [CFE\\_SB SND RTG\\_ERR1\\_EID](#) and [CFE\\_SB\\_FILEWRITE\\_ERR\\_EID](#)

### Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

Definition at line 362 of file default\_cfe\_sb\_fcncodes.h.

## 12.108.2.10 CFE\_SB\_WRITE\_PIPE\_INFO\_CC #define CFE\_SB\_WRITE\_PIPE\_INFO\_CC 7

**Name** Write Pipe Info to a File

### Description

This command will create a file containing the software bus pipe information. The pipe information contains information about every pipe that has been created through the [CFE\\_SB\\_CreatePipe](#) API. An absolute path and filename may be specified in the command. If this command field contains an empty string (NULL terminator as the first character) the default file path and name is used. The default file path and name is defined in the platform configuration file as [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_PIPE\\_FILENAME](#).

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_WritePipe2File

### Command Structure

[CFE\\_SB\\_WritePipeInfoCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_SB_CMDPC` - command execution counter will increment. NOTE: the command counter is incremented when the request is accepted, before writing the file, which is performed as a background task.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_PIPE\\_FILENAME](#) configuration parameter) will be updated with the latest information.
- The [CFE\\_SB SND RTG\\_EID](#) debug event message will be generated

### Error Conditions

This command may fail for the following reason(s):

- A previous request to write a software bus information file has not yet completed
- The specified FileName cannot be parsed

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_SB_CMDEC` - command error counter will increment
- A command specific error event message is issued for all error cases. See [CFE\\_SB SND RTG\\_ERR1\\_EID](#) and [CFE\\_SB\\_FILEWRITE\\_ERR\\_EID](#)

### Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

Definition at line 316 of file default\_cfe\_sb\_fcncodes.h.

#### 12.108.2.11 CFE\_SB\_WRITE\_ROUTING\_INFO\_CC #define CFE\_SB\_WRITE\_ROUTING\_INFO\_CC 3

**Name** Write Software Bus Routing Info to a File

### Description

This command will create a file containing the software bus routing information. The routing information contains information about every subscription that has been received through the SB subscription APIs. An absolute path and filename may be specified in the command. If this command field contains an empty string (NULL terminator as the first character) the default file path and name is used. The default file path and name is defined in the platform configuration file as [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_ROUTING\\_FILENAME](#).

**Command Mnemonic(s)** \$sc\_\$cpu\_SB\_WriteRouting2File

### Command Structure

[CFE\\_SB\\_WriteRoutingInfoCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_SB_CMDPC` - command execution counter will increment. NOTE: the command counter is incremented when the request is accepted, before writing the file, which is performed as a background task.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_ROUTING\\_FILENAME](#) configuration parameter) will be updated with the latest information.
- The [CFE\\_SB SND RTG EID](#) debug event message will be generated

### Error Conditions

This command may fail for the following reason(s):

- A previous request to write a software bus information file has not yet completed
- The specified FileName cannot be parsed

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_SB_CMDEC` - command error counter will increment
- A command specific error event message is issued for all error cases. See [CFE\\_SB SND RTG\\_ERR1\\_EID](#) and [CFE\\_SB\\_FILEWRITE\\_ERR\\_EID](#)

### Criticality

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

Definition at line 192 of file default\_cfe\_sb\_fcncodes.h.

## 12.109 cfe/modules/sb/config/default\_cfe\_sb\_interface\_cfg.h File Reference

### Macros

- `#define CFE_MISSION_SB_MAX_SB_MSG_SIZE 32768`
- `#define CFE_MISSION_SB_MAX_PIPES 64`

### 12.109.1 Detailed Description

CFE Software Bus (CFE\_SB) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.109.2 Macro Definition Documentation

#### 12.109.2.1 CFE\_MISSION\_SB\_MAX\_PIPES `#define CFE_MISSION_SB_MAX_PIPES 64`

**Purpose** Maximum Number of pipes that SB command/telemetry messages may hold

##### Description:

Dictates the maximum number of unique Pipes the SB message definitions will hold.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

##### Limits

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 67 of file default\_cfe\_sb\_interface\_cfg.h.

**12.109.2.2 CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE #define CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE 32768**

**Purpose** Maximum SB Message Size

**Description:**

The following definition dictates the maximum message size allowed on the software bus. SB checks the pkt length field in the header of all messages sent. If the pkt length field indicates the message is larger than this define, SB sends an event and rejects the send.

**Limits**

This parameter has a lower limit of 6 (CCSDS primary header size). There are no restrictions on the upper limit however, the maximum message size is system dependent and should be verified. Total message size values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 50 of file default\_cfe\_sb\_interface\_cfg.h.

**12.110 cfe/modules/sb/config/default\_cfe\_sb\_internal\_cfg.h File Reference**

**Macros**

- #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256
- #define CFE\_PLATFORM\_SB\_MAX\_PIPES 64
- #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4
- #define CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES 524288
- #define CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID 0x1FFF
- #define CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME "/ram/cfe\_sb\_route.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME "/ram/cfe\_sb\_pipe.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME "/ram/cfe\_sb\_msgmap.dat"
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_SB\_SEND\_NO\_SUBS\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_EVS\_FIRST\_4\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_SB\_DUP\_SUBSCRIPTION\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_EVS\_FIRST\_4\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT3 CFE\_SB\_MSGID\_LIM\_ERR\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_EVS\_FIRST\_16\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT4 CFE\_SB\_Q\_FULL\_ERR\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_EVS\_FIRST\_16\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT5 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT6 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT7 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT8 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK8 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 20
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 36
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 96

- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE (CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE + 128)
- #define CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY 64
- #define CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

### 12.110.1 Detailed Description

CFE Software Bus (CFE\_SB) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.110.2 Macro Definition Documentation

#### 12.110.2.1 CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES #define CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYT← ES 524288

**Purpose** Size of the SB buffer memory pool

**Description:**

Dictates the size of the SB memory pool. For each message the SB sends, the SB dynamically allocates from this memory pool, the memory needed to process the message. The memory needed to process each message is msg size + msg descriptor(CFE\_SB\_BufferD\_t). This memory pool is also used to allocate destination descriptors (CFE\_SB\_DestinationD\_t) during the subscription process. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'. Some memory statistics have been added to the SB housekeeping packet. NOTE: It is important to monitor these statistics to ensure the desired memory margin is met.

#### Limits

This parameter has a lower limit of 512 and an upper limit of UINT\_MAX (4 Gigabytes).

Definition at line 123 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.2 CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILE←  
NAME "/ram/cfe\_sb\_msgmap.dat"

**Purpose** Default Message Map Filename

**Description:**

The value of this constant defines the filename used to store the software bus message map information. This filename is used only when no filename is specified in the command. The message map is a lookup table (array of 16bit words) that has an element for each possible MsgId value and holds the routing table index for that MsgId. The Msg Map provides fast access to the destinations of a message.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 194 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.3 CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT** #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4

**Purpose** Default Subscription Message Limit

**Description:**

Dictates the default Message Limit when using the [CFE\\_SB\\_Subscribe](#) API. This will limit the number of messages with a specific message ID that can be received through a subscription. This only changes the default; other message limits can be set on a per subscription basis using [CFE\\_SB\\_SubscribeEx](#).

**Limits**

This parameter has a lower limit of 4 and an upper limit of 65535.

Definition at line 101 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.4 CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FIL←  
ENAME "/ram/cfe\_sb\_pipe.dat"

**Purpose** Default Pipe Information Filename

**Description:**

The value of this constant defines the filename used to store the software bus pipe information. This filename is used only when no filename is specified in the command.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 177 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.5 CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME "/ram/cfe\_sb\_route.dat"

**Purpose** Default Routing Information Filename

**Description:**

The value of this constant defines the filename used to store the software bus routing information. This filename is used only when no filename is specified in the command.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 163 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.6 CFE\_PLATFORM\_SB\_FILTER\_MASK1** #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_EVS\_FIRST\_4\_STOP  
Definition at line 212 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.7 CFE\_PLATFORM\_SB\_FILTER\_MASK2** #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_EVS\_FIRST\_4\_STOP  
Definition at line 215 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.8 CFE\_PLATFORM\_SB\_FILTER\_MASK3** #define CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_EVS\_FIRST\_16\_STOP  
Definition at line 218 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.9 CFE\_PLATFORM\_SB\_FILTER\_MASK4** #define CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_EVS\_FIRST\_16\_STOP  
Definition at line 221 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.10 CFE\_PLATFORM\_SB\_FILTER\_MASK5** #define CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_EVS\_NO\_FILTER  
Definition at line 224 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.11 CFE\_PLATFORM\_SB\_FILTER\_MASK6** #define CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_EVS\_NO\_FILTER  
Definition at line 227 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.12 CFE\_PLATFORM\_SB\_FILTER\_MASK7** #define CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_EVS\_NO\_FILTER  
Definition at line 230 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.13 CFE\_PLATFORM\_SB\_FILTER\_MASK8** #define CFE\_PLATFORM\_SB\_FILTER\_MASK8 CFE\_EVS\_NO\_FILTER  
Definition at line 233 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.14 CFE\_PLATFORM\_SB\_FILTERED\_EVENT1**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT1 CFE_SB_SEND_NO_SUBS_EID
```

**Purpose** SB Event Filtering

**Description:**

This group of configuration parameters dictates what SB events will be filtered through SB. The filtering will begin after the SB task initializes and stay in effect until a cmd to SB changes it. This allows the operator to set limits on the number of event messages that are sent during system initialization. NOTE: Set all unused event values and mask values to zero

**Limits**

This filtering applies only to SB events. These parameters have a lower limit of 0 and an upper limit of 65535.

Definition at line 211 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.15 CFE\_PLATFORM\_SB\_FILTERED\_EVENT2**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT2 CFE_SB_DUP_SUBSCRIP_EID
```

Definition at line 214 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.16 CFE\_PLATFORM\_SB\_FILTERED\_EVENT3**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT3 CFE_SB_MSGID_LIM_ERR_EID
```

Definition at line 217 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.17 CFE\_PLATFORM\_SB\_FILTERED\_EVENT4**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT4 CFE_SB_Q_FULL_ERR_EID
```

Definition at line 220 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.18 CFE\_PLATFORM\_SB\_FILTERED\_EVENT5**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT5 0
```

Definition at line 223 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.19 CFE\_PLATFORM\_SB\_FILTERED\_EVENT6**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT6 0
```

Definition at line 226 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.20 CFE\_PLATFORM\_SB\_FILTERED\_EVENT7**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT7 0
```

Definition at line 229 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.21 CFE\_PLATFORM\_SB\_FILTERED\_EVENT8**

```
#define CFE_PLATFORM_SB_FILTERED_EVENT8 0
```

Definition at line 232 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.22 CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID**

```
#define CFE_PLATFORM_SB_HIGHEST_VALID_MSGID 0x1FFF
```

**Purpose** Highest Valid Message Id

**Description:**

The value of this constant dictates the range of valid message ID's, from 0 to CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID (inclusive).

Although this can be defined differently across platforms, each platform can only publish/subscribe to message ids within their allowable range. Typically this value is set the same across all mission platforms to avoid this complexity.

**Limits**

CFE\_SB\_INVALID\_MSG is set to the maximum representable number of type [CFE\\_SB\\_MsgId\\_t](#). CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID lower limit is 1, up to CFE\_SB\_INVALID\_MSG\_ID - 1.

When using the direct message map implementation for software bus routing, this value is used to size the map where a value of 0xFFFF results in a 16 KBytes map and 0xFFFF is 128 KBytes.

When using the hash implementation for software bus routing, a multiple of the CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS is used to size the message map. In that case the range selected here does not impact message map memory use, so it's reasonable to use up to the full range supported by the message ID implementation.

Definition at line 149 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.23 CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE ([CFE\\_MISSION\\_SB\\_MAX\\_SB\\_MESSAGE\\_BLOCK\\_SIZE](#)  
+ 128)

Definition at line 262 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.24 CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT** #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16

**Purpose** Maximum Number of unique local destinations a single MsgId can have

**Description:**

Dictates the maximum number of unique local destinations a single MsgId can have.

**Limits**

This parameter has a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of destinations per packet is system dependent and should be verified. Destination number values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 86 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.25 CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS** #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256

**Purpose** Maximum Number of Unique Message IDs SB Routing Table can hold

**Description:**

Dictates the maximum number of unique MsgIds the SB routing table will hold. This constant has a direct effect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

### Limits

This must be a power of two if software bus message routing hash implementation is being used. Lower than 64 will cause unit test failures, and telemetry reporting is impacted below 32. There is no hard upper limit, but impacts memory footprint. For software bus message routing search implementation the number of msg ids subscribed to impacts performance.

Definition at line 53 of file default\_cfe\_sb\_internal\_cfg.h.

#### 12.110.2.26 CFE\_PLATFORM\_SB\_MAX\_PIPES `#define CFE_PLATFORM_SB_MAX_PIPES 64`

**Purpose** Maximum Number of Unique Pipes SB Routing Table can hold

##### Description:

Dictates the maximum number of unique Pipes the SB routing table will hold. This constant has a direct effect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

### Limits

This parameter has a lower limit of 1. This parameter must also be less than or equal to OS\_MAX\_QUEUES.

Definition at line 70 of file default\_cfe\_sb\_internal\_cfg.h.

#### 12.110.2.27 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 `#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_01 8`

**Purpose** Define SB Memory Pool Block Sizes

##### Description:

Software Bus Memory Pool Block Sizes

### Limits

These sizes MUST be increasing and MUST be an integral multiple of 4. The number of block sizes defined cannot exceed [CFE\\_PLATFORM\\_ES\\_POOL\\_MAX\\_BUCKETS](#)

Definition at line 246 of file default\_cfe\_sb\_internal\_cfg.h.

#### 12.110.2.28 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 `#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_02 16`

Definition at line 247 of file default\_cfe\_sb\_internal\_cfg.h.

#### 12.110.2.29 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 `#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_03 20`

Definition at line 248 of file default\_cfe\_sb\_internal\_cfg.h.

#### 12.110.2.30 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 `#define CFE_PLATFORM_SB_MEM_BLOCK_SIZE_04 36`

Definition at line 249 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.31 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 64  
Definition at line 250 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.32 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 96  
Definition at line 251 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.33 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 128  
Definition at line 252 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.34 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 160  
Definition at line 253 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.35 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 256  
Definition at line 254 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.36 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 512  
Definition at line 255 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.37 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 1024  
Definition at line 256 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.38 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 2048  
Definition at line 257 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.39 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 4096  
Definition at line 258 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.40 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 8192  
Definition at line 259 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.41 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 16384  
Definition at line 260 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.42 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 16 32768

Definition at line 261 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.43 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 273 of file default\_cfe\_sb\_internal\_cfg.h.

**12.110.2.44 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 288 of file default\_cfe\_sb\_internal\_cfg.h.

## 12.111 cfe/modules/sb/config/default\_cfe\_sb\_mission\_cfg.h File Reference

```
#include "cfe_sb_interface_cfg.h"
```

### 12.111.1 Detailed Description

CFE Event Services (CFE\_SB) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.112 cfe/modules/sb/config/default\_cfe\_sb\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_sb_msgdefs.h"
#include "cfe_sb_msgstruct.h"
```

### 12.112.1 Detailed Description

Specification for the CFE Event Services (CFE\_SB) command and telemetry message data types.  
This is a compatibility header for the "cfe\_sb\_msg.h" file that has traditionally provided the message definitions for cFS apps.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.113 cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h File Reference

```
#include "cfe_sb_fcncodes.h"
```

### 12.113.1 Detailed Description

Specification for the CFE Event Services (CFE\_SB) command and telemetry message constant definitions.  
For CFE\_SB this is only the function/command code definitions

## 12.114 cfe/modules/sb/config/default\_cfe\_sb\_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_sb_topicids.h"
```

### Macros

- #define CFE\_SB\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_SB\_CMD\_MSG /\* 0x1803 \*/
- #define CFE\_SB\_SEND\_HK\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_SB\_SEND\_HK\_MSG /\* 0x180B \*/
- #define CFE\_SB\_SUB\_RPT\_CTRL\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_MSG /\* 0x180E \*/
- #define CFE\_SB\_HK\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_SB\_HK\_TLM\_MSG /\* 0x0803 \*/
- #define CFE\_SB\_STATS\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_SB\_STATS\_TLM\_MSG /\* 0x080A \*/
- #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 \*/

### 12.114.1 Detailed Description

CFE Event Services (CFE\_SB) Application Message IDs

## 12.114.2 Macro Definition Documentation

**12.114.2.1 CFE\_SB\_ALLSUBS\_TLM\_MID** #define CFE\_SB\_ALLSUBS\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_SB\_ALLSUBS\_TLM\_MSG /\* 0x080D \*/  
Definition at line 41 of file default\_cfe\_sb\_msgids.h.

**12.114.2.2 CFE\_SB\_CMD\_MID** #define CFE\_SB\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_SB\_CMD\_MSG /\* 0x1803 \*/  
Definition at line 32 of file default\_cfe\_sb\_msgids.h.

**12.114.2.3 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 39 of file default\_cfe\_sb\_msgids.h.

**12.114.2.4 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 42 of file default\_cfe\_sb\_msgids.h.

**12.114.2.5 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 33 of file default\_cfe\_sb\_msgids.h.

**12.114.2.6 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 40 of file default\_cfe\_sb\_msgids.h.

**12.114.2.7 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 34 of file default\_cfe\_sb\_msgids.h.

## 12.115 cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h File Reference

```
#include "cfe_sb_interface_cfg.h"
#include "cfe_msg_hdr.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](#)

## 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 CFE\_MSG\_CommandHeader\_t [CFE\\_SB\\_SendHkCmd\\_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`

### 12.115.1 Detailed Description

Purpose: cFE Executive Services (SB) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide

Notes:

### 12.115.2 Typedef Documentation

#### 12.115.2.1 `CFE_SB_AllSubscriptionsTlm_Payload_t` `typedef struct CFE_SB_AllSubscriptionsTlm_Payload CFE_SB_AllSubscriptionsTlm_Payload_t`

**Name** SB Previous Subscriptions Packet

This structure defines the pkt(s) sent by SB that contains a list of all current subscriptions. This pkt is generated on cmd and intended to be used primarily by the Software Bus Networking Application (SBN). Typically, when the cmd is received there are more subscriptions than can fit in one pkt. The complete list of subscriptions is sent via a series of segmented pkts.

#### 12.115.2.2 `CFE_SB_AllSubscriptionsTlm_t` `typedef struct CFE_SB_AllSubscriptionsTlm CFE_SB_AllSubscriptionsTlm_t`

#### 12.115.2.3 `CFE_SB_DisableRouteCmd_t` `typedef CFE_SB_RouteCmd_t CFE_SB_DisableRouteCmd_t`

Definition at line 114 of file default\_cfe\_sb\_msgstruct.h.

#### 12.115.2.4 `CFE_SB_DisableSubReportingCmd_t` `typedef CFE_MSG_CommandHeader_t CFE_SB_DisableSubReportingCmd_t`

Definition at line 55 of file default\_cfe\_sb\_msgstruct.h.

#### 12.115.2.5 `CFE_SB_EnableRouteCmd_t` `typedef CFE_SB_RouteCmd_t CFE_SB_EnableRouteCmd_t`

Definition at line 113 of file default\_cfe\_sb\_msgstruct.h.

#### 12.115.2.6 `CFE_SB_EnableSubReportingCmd_t` `typedef CFE_MSG_CommandHeader_t CFE_SB_EnableSubReportingCmd_t`

Definition at line 54 of file default\_cfe\_sb\_msgstruct.h.

**12.115.2.7 CFE\_SB\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_SB_HousekeepingTlm_Payload CFE_SB_HousekeepingTlm_Payload_t`

**Name** Software Bus task housekeeping Packet

**12.115.2.8 CFE\_SB\_HousekeepingTlm\_t** `typedef struct CFE_SB_HousekeepingTlm CFE_SB_HousekeepingTlm_t`

**12.115.2.9 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](#)

**12.115.2.10 CFE\_SB\_NoopCmd\_t** `typedef CFE_MSG_CommandHeader_t CFE_SB_NoopCmd_t`  
Definition at line 52 of file default\_cfe\_sb\_msgstruct.h.

**12.115.2.11 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](#)

**12.115.2.12 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.

**12.115.2.13 CFE\_SB\_ResetCountersCmd\_t** `typedef CFE_MSG_CommandHeader_t CFE_SB_ResetCountersCmd_t`  
Definition at line 53 of file default\_cfe\_sb\_msgstruct.h.**12.115.2.14 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 PipId combination.

**12.115.2.15 CFE\_SB\_RouteCmd\_t** `typedef struct CFE_SB_RouteCmd CFE_SB_RouteCmd_t`  
Enable/Disable Route Command.**12.115.2.16 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](#)

**12.115.2.17 CFE\_SB\_SendHkCmd\_t** `typedef CFE_MSG_CommandHeader_t CFE_SB_SendHkCmd_t`  
Definition at line 58 of file default\_cfe\_sb\_msgstruct.h.

**12.115.2.18 CFE\_SB\_SendPrevSubsCmd\_t** `typedef CFE_MSG_CommandHeader_t CFE_SB_SendPrevSubsCmd_t`  
Definition at line 57 of file default\_cfe\_sb\_msgstruct.h.

**12.115.2.19 CFE\_SB\_SendSbStatsCmd\_t** `typedef CFE_MSG_CommandHeader_t CFE_SB_SendSbStatsCmd_t`  
Definition at line 56 of file default\_cfe\_sb\_msgstruct.h.

**12.115.2.20 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](#)

**12.115.2.21 CFE\_SB\_SingleSubscriptionTlm\_t** `typedef struct CFE_SB_SingleSubscriptionTlm CFE_SB_SingleSubscriptionTlm_t`

**12.115.2.22 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](#)

**12.115.2.23 CFE\_SB\_StatsTlm\_t** `typedef struct CFE_SB_StatsTlm CFE_SB_StatsTlm_t`

**12.115.2.24 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](#)

**12.115.2.25 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

**12.115.2.26 CFE\_SB\_WriteFileInfoCmd\_t** `typedef struct CFE_SB_WriteFileInfoCmd CFE_SB_WriteFileInfoCmd_t`  
Write File Info Command.

**12.115.2.27 CFE\_SB\_WriteMapInfoCmd\_t** `typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteMapInfoCmd_t`  
Definition at line 84 of file default\_cfe\_sb\_msgstruct.h.

**12.115.2.28 CFE\_SB\_WritePipeInfoCmd\_t** `typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WritePipeInfoCmd_t`  
Definition at line 83 of file default\_cfe\_sb\_msgstruct.h.

**12.115.2.29 CFE\_SB\_WriteRoutingInfoCmd\_t** `typedef CFE_SB_WriteFileInfoCmd_t CFE_SB_WriteRoutingInfoCmd_t`  
Definition at line 82 of file default\_cfe\_sb\_msgstruct.h.

## 12.116 cfe/modules/sb/config/default\_cfe\_sb\_platform\_cfg.h File Reference

```
#include "cfe_sb_mission_cfg.h"
#include "cfe_sb_internal_cfg.h"
```

### 12.116.1 Detailed Description

CFE Software Bus (CFE\_SB) Application Platform Configuration Header File

This is a compatibility header for the "platform\_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.117 cfe/modules/sb/config/default\_cfe\_sb\_topicids.h File Reference

### Macros

- `#define CFE_MISSION_SB_CMD_MSG 3`
- `#define CFE_MISSION_SB_SEND_HK_MSG 11`
- `#define CFE_MISSION_SB_SUB_RPT_CTRL_MSG 14`
- `#define CFE_MISSION_SB_HK_TLM_MSG 3`
- `#define CFE_MISSION_SB_STATS_TLM_MSG 10`
- `#define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13`
- `#define CFE_MISSION_SB_ONESUB_TLM_MSG 14`

### 12.117.1 Detailed Description

CFE Software Bus (CFE\_SB) Application Topic IDs

### 12.117.2 Macro Definition Documentation

**12.117.2.1 CFE\_MISSION\_SB\_ALLSUBS\_TLM\_MSG** `#define CFE_MISSION_SB_ALLSUBS_TLM_MSG 13`  
Definition at line 50 of file default\_cfe\_sb\_topicids.h.

**12.117.2.2 CFE\_MISSION\_SB\_CMD\_MSG** `#define CFE_MISSION_SB_CMD_MSG 3`

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE command messages

**Limits**

Not Applicable

Definition at line 35 of file default\_cfe\_sb\_topicids.h.

**12.117.2.3 CFE\_MISSION\_SB\_HK\_TLM\_MSG #define CFE\_MISSION\_SB\_HK\_TLM\_MSG 3****Purpose** cFE Portable Message Numbers for Telemetry**Description:**

Portable message numbers for the cFE telemetry messages

**Limits**

Not Applicable

Definition at line 48 of file default\_cfe\_sb\_topicids.h.

**12.117.2.4 CFE\_MISSION\_SB\_ONESUB\_TLM\_MSG #define CFE\_MISSION\_SB\_ONESUB\_TLM\_MSG 14**

Definition at line 51 of file default\_cfe\_sb\_topicids.h.

**12.117.2.5 CFE\_MISSION\_SB\_SEND\_HK\_MSG #define CFE\_MISSION\_SB\_SEND\_HK\_MSG 11**

Definition at line 36 of file default\_cfe\_sb\_topicids.h.

**12.117.2.6 CFE\_MISSION\_SB\_STATS\_TLM\_MSG #define CFE\_MISSION\_SB\_STATS\_TLM\_MSG 10**

Definition at line 49 of file default\_cfe\_sb\_topicids.h.

**12.117.2.7 CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_MSG #define CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_MSG 14**

Definition at line 37 of file default\_cfe\_sb\_topicids.h.

**12.118 cfe/modules/sb/fsw/inc/cfe\_sb\_eventids.h File Reference****Macros****SB event IDs**

- #define **CFE\_SB\_INIT\_EID** 1  
*SB Initialization Event ID.*
- #define **CFE\_SB\_CR\_PIPE\_BAD\_ARG\_EID** 2  
*SB Create Pipe API Bad Argument Event ID.*
- #define **CFE\_SB\_MAX\_PIPES\_MET\_EID** 3  
*SB Create Pipe API Max Pipes Exceeded Event ID.*
- #define **CFE\_SB\_CR\_PIPE\_ERR\_EID** 4  
*SB Create Pipe API Queue Create Failure Event ID.*

- #define `CFE_SB_PIPE_ADDED_EID` 5  
    *SB Create Pipe API Success Event ID.*
- #define `CFE_SB_SUB_ARG_ERR_EID` 6  
    *SB Subscribe API Bad Argument Event ID.*
- #define `CFE_SB_DUP_SUBSCRIP_EID` 7  
    *SB Subscribe API Duplicate MsgId Subscription Event ID.*
- #define `CFE_SB_MAX_MSGS_MET_EID` 8  
    *SB Subscribe API Max Subscriptions Exceeded Event ID.*
- #define `CFE_SB_MAX_DESTS_MET_EID` 9  
    *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 Success Event ID.*

- #define `CFE_SB_DSBL RTE1_EID` 36  
*SB Enable Route Command Invalid MsgId or Pipe Event ID.*
- #define `CFE_SB_DSBL RTE2_EID` 37  
*SB Disable Route Command Invalid MsgId/PipeId Pair Event ID.*
- #define `CFE_SB_DSBL RTE3_EID` 38  
*SB Disable Route Command Success 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_SETPPIPEOPTS_ID_ERR_EID` 55  
*SB Set Pipe Opts API Invalid Pipe Event ID.*
- #define `CFE_SB_SETPPIPEOPTS_OWNER_ERR_EID` 56  
*SB Set Pipe Opts API Not Owner Event ID.*
- #define `CFE_SB_SETPPIPEOPTS_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.*

### 12.118.1 Detailed Description

cFE Software Bus Services Event IDs

### 12.118.2 Macro Definition Documentation

**12.118.2.1 CFE\_SB\_BAD\_CMD\_CODE\_EID** #define CFE\_SB\_BAD\_CMD\_CODE\_EID 42  
SB Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE\\_SB\\_CMD\\_MID](#) or [CFE\\_SB\\_SUB\\_RPT\\_CTRL\\_MID](#) received on the SB message pipe. OVERLOADED

Definition at line 461 of file cfe\_sb\_eventids.h.

**12.118.2.2 CFE\_SB\_BAD\_MSGID\_EID** #define CFE\_SB\_BAD\_MSGID\_EID 43  
SB Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the SB message pipe.

Definition at line 472 of file cfe\_sb\_eventids.h.

**12.118.2.3 CFE\_SB\_BAD\_PIPEID\_EID** #define CFE\_SB\_BAD\_PIPEID\_EID 19  
SB Receive Buffer API Invalid Pipe Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_ReceiveBuffer](#) API failure due to an invalid Pipe ID.  
Definition at line 244 of file cfe\_sb\_eventids.h.

**12.118.2.4 CFE\_SB\_CMD0\_RCVD\_EID** #define CFE\_SB\_CMD0\_RCVD\_EID 28  
SB No-op Command Success Event ID.

Type: INFORMATION

Cause:

[SB NO-OP Command](#) success.  
Definition at line 335 of file cfe\_sb\_eventids.h.

**12.118.2.5 CFE\_SB\_CMD1\_RCVD\_EID** #define CFE\_SB\_CMD1\_RCVD\_EID 29  
SB Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[SB Reset Counters Command](#) success.  
Definition at line 346 of file cfe\_sb\_eventids.h.

**12.118.2.6 CFE\_SB\_CR\_PIPE\_BAD\_ARG\_EID** #define CFE\_SB\_CR\_PIPE\_BAD\_ARG\_EID 2  
SB Create Pipe API Bad Argument Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_CreatePipe](#) API failure due to a bad input argument.  
Definition at line 53 of file cfe\_sb\_eventids.h.

**12.118.2.7 CFE\_SB\_CR\_PIPE\_ERR\_EID** #define CFE\_SB\_CR\_PIPE\_ERR\_EID 4  
SB Create Pipe API Queue Create Failure Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_CreatePipe](#) API failure creating the queue.  
Definition at line 75 of file cfe\_sb\_eventids.h.

**12.118.2.8 CFE\_SB\_CR\_PIPE\_NAME\_TAKEN\_EID** #define CFE\_SB\_CR\_PIPE\_NAME\_TAKEN\_EID 69  
SB Create Pipe API Name Taken Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_CreatePipe](#) API failure due to pipe name taken.  
Definition at line 750 of file cfe\_sb\_eventids.h.

**12.118.2.9 CFE\_SB\_CR\_PIPE\_NO\_FREE\_EID** #define CFE\_SB\_CR\_PIPE\_NO\_FREE\_EID 70  
SB Create Pipe API Queues Exhausted Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_CreatePipe](#) API failure due to no free queues.  
Definition at line 761 of file cfe\_sb\_eventids.h.

**12.118.2.10 CFE\_SB\_DEL\_PIPE\_ERR1\_EID** #define CFE\_SB\_DEL\_PIPE\_ERR1\_EID 46  
SB Pipe Delete API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Delete Pipe API failed due to an invalid input argument.  
Definition at line 507 of file cfe\_sb\_eventids.h.

**12.118.2.11 CFE\_SB\_DEL\_PIPE\_ERR2\_EID** #define CFE\_SB\_DEL\_PIPE\_ERR2\_EID 54  
SB Delete Pipe API Not Owner Event ID.

Type: ERROR

Cause:

An SB Delete Pipe API failed due to not being the pipe owner.  
Definition at line 595 of file cfe\_sb\_eventids.h.

**12.118.2.12 CFE\_SB\_DEST\_BLK\_ERR\_EID** #define CFE\_SB\_DEST\_BLK\_ERR\_EID 20  
SB Subscribe API Get Destination Block Failure Event ID.

Type: ERROR

Cause:

An SB Subscribe API call failed to get a destination block.  
Definition at line 255 of file cfe\_sb\_eventids.h.

**12.118.2.13 CFE\_SB\_DSBL RTE1\_EID** #define CFE\_SB\_DSBL RTE1\_EID 36  
SB Disable Route Command Invalid MsgId/PipeId Pair Event ID.

Type: ERROR

Cause:

[SB Disable Route Command](#) failure due to the Message ID not being subscribed to the pipe.  
Definition at line 404 of file cfe\_sb\_eventids.h.

**12.118.2.14 CFE\_SB\_DSBL RTE2\_EID** #define CFE\_SB\_DSBL RTE2\_EID 37  
SB Disable Route Command Success Event ID.

Type: DEBUG

Cause:

[SB Disable Route Command](#) success.  
Definition at line 415 of file cfe\_sb\_eventids.h.

**12.118.2.15 CFE\_SB\_DSBL RTE3\_EID** #define CFE\_SB\_DSBL RTE3\_EID 38  
SB Disable Route Command Invalid MsgId or Pipe Event ID.

Type: ERROR

Cause:

[SB Disable Route Command](#) failure due to an invalid MsgId or Pipe.  
Definition at line 427 of file cfe\_sb\_eventids.h.

**12.118.2.16 CFE\_SB\_DUP\_SUBSCRIP\_EID** #define CFE\_SB\_DUP\_SUBSCRIP\_EID 7  
SB Subscribe API Duplicate MsgId Subscription Event ID.

Type: INFORMATION

Cause:

An SB Subscribe API was called with a Message ID that was already subscribed on the pipe on the pipe.  
Definition at line 109 of file cfe\_sb\_eventids.h.

**12.118.2.17 CFE\_SB\_ENBL RTE1\_EID** #define CFE\_SB\_ENBL RTE1\_EID 33  
SB Enable Route Command Invalid MsgId/PipeID Pair Event ID.

Type: ERROR

Cause:

[SB Enable Route Command](#) failure due to the Message ID not being subscribed to the pipe.  
Definition at line 369 of file cfe\_sb\_eventids.h.

**12.118.2.18 CFE\_SB\_ENBL RTE2\_EID** #define CFE\_SB\_ENBL RTE2\_EID 34  
SB Enable Route Command Success Event ID.

Type: DEBUG

Cause:

[SB Enable Route Command](#) success.  
Definition at line 380 of file cfe\_sb\_eventids.h.

**12.118.2.19 CFE\_SB\_ENBL RTE3\_EID** #define CFE\_SB\_ENBL RTE3\_EID 35  
SB Enable Route Command Invalid MsgId or Pipe Event ID.

Type: ERROR

Cause:

[SB Enable Route Command](#) failure due to an invalid MsgId or Pipe.  
Definition at line 392 of file cfe\_sb\_eventids.h.

**12.118.2.20 CFE\_SB\_FILEWRITE\_ERR\_EID** #define CFE\_SB\_FILEWRITE\_ERR\_EID 49  
SB File Write Failed Event ID.

Type: ERROR

Cause:

An SB file write failure encountered when writing to the file.  
Definition at line 540 of file cfe\_sb\_eventids.h.

**12.118.2.21 CFE\_SB\_FULL\_SUB\_PKT\_EID** #define CFE\_SB\_FULL\_SUB\_PKT\_EID 44  
SB Send Previous Subscriptions Command Full Packet Sent Event ID.

Type: DEBUG

Cause:

[SB Send Previous Subscriptions Command](#) processing sent a full subscription packet.  
Definition at line 484 of file cfe\_sb\_eventids.h.

**12.118.2.22 CFE\_SB\_GET\_BUF\_ERR\_EID** #define CFE\_SB\_GET\_BUF\_ERR\_EID 16  
SB Transmit API Buffer Request Failure Event ID.

Type: ERROR

Cause:

An SB Transmit API call buffer request failed.  
Definition at line 210 of file cfe\_sb\_eventids.h.

**12.118.2.23 CFE\_SB\_GETPIPEIDBYNAME\_EID** #define CFE\_SB\_GETPIPEIDBYNAME\_EID 65  
SB Get Pipe ID By Name API Success Event ID.

Type: DEBUG

Cause:

[CFE\\_SB\\_GetPipeldByName](#) success.  
Definition at line 705 of file cfe\_sb\_eventids.h.

**12.118.2.24 CFE\_SB\_GETPIPEIDBYNAME\_NAME\_ERR\_EID** #define CFE\_SB\_GETPIPEIDBYNAME\_NAME\_ERR\_EID 67  
SB Get Pipe ID By Name API Name Not Found Or ID Not Matched Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_GetPipeldByName](#) failure due to name not found or ID mismatch. OVERLOADED  
Definition at line 727 of file cfe\_sb\_eventids.h.

**12.118.2.25 CFE\_SB\_GETPIPEIDBYNAME\_NULL\_ERR\_EID** #define CFE\_SB\_GETPIPEIDBYNAME\_NULL\_ERR\_EID 66  
SB Get Pipe ID By Name API Invalid Pointer Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_GetPipeldByName](#) failure due to invalid pointer.  
Definition at line 716 of file cfe\_sb\_eventids.h.

**12.118.2.26 CFE\_SB\_GETPIPENAME\_EID** #define CFE\_SB\_GETPIPENAME\_EID 62  
SB Get Pipe Name API Success Event ID.

Type: DEBUG

Cause:

[CFE\\_SB\\_GetPipeName](#) success.  
Definition at line 672 of file cfe\_sb\_eventids.h.

**12.118.2.27 CFE\_SB\_GETPIPENAME\_ID\_ERR\_EID** #define CFE\_SB\_GETPIPENAME\_ID\_ERR\_EID 64  
SB Get Pipe Name API Invalid Pipe or Resource Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_GetPipeName](#) failure due to invalid pipe ID or failure in retrieving resource name. OVERLOADED  
Definition at line 694 of file cfe\_sb\_eventids.h.

**12.118.2.28 CFE\_SB\_GETPIPENAME\_NULL\_PTR\_EID** #define CFE\_SB\_GETPIPENAME\_NULL\_PTR\_EID 63  
SB Get Pipe Name API Invalid Pointer Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_GetPipeName](#) failure due to invalid pointer.  
Definition at line 683 of file cfe\_sb\_eventids.h.

**12.118.2.29 CFE\_SB\_GETPIPEOPTS\_EID** #define CFE\_SB\_GETPIPEOPTS\_EID 60  
SB Get Pipe Opt Success Event ID.

Type: DEBUG

Cause:

[CFE\\_SB\\_GetPipeOpt](#) success.  
Definition at line 661 of file cfe\_sb\_eventids.h.

**12.118.2.30 CFE\_SB\_GETPIPEOPTS\_ID\_ERR\_EID** #define CFE\_SB\_GETPIPEOPTS\_ID\_ERR\_EID 58  
SB Get Pipe Opt API Invalid Pipe Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_GetPipeOpt](#) failure due to invalid pipe ID.  
Definition at line 639 of file cfe\_sb\_eventids.h.

**12.118.2.31 CFE\_SB\_GETPIPEOPTS\_PTR\_ERR\_EID** #define CFE\_SB\_GETPIPEOPTS\_PTR\_ERR\_EID 59  
SB Get Pipe Opt API Invalid Pointer Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_GetPipeOpts](#) failure due to invalid pointer.  
Definition at line 650 of file cfe\_sb\_eventids.h.

**12.118.2.32 CFE\_SB\_HASHCOLLISION\_EID** #define CFE\_SB\_HASHCOLLISION\_EID 23  
SB Subscribe API Message Table Hash Collision Event ID.

Type: DEBUG

Cause:

An SB Subscribe API call caused a message table hash collision, which will impact message transmission performance.  
This can be resolved by deconflicting MsgId values or increasing [CFE\\_PLATFORM\\_SB\\_MAX\\_MSG\\_IDS](#).  
Definition at line 290 of file cfe\_sb\_eventids.h.

**12.118.2.33 CFE\_SB\_INIT\_EID** #define CFE\_SB\_INIT\_EID 1  
SB Initialization Event ID.

Type: INFORMATION

Cause:

Software Bus Services Task initialization complete.  
Definition at line 42 of file cfe\_sb\_eventids.h.

**12.118.2.34 CFE\_SB\_LEN\_ERR\_EID** #define CFE\_SB\_LEN\_ERR\_EID 68  
SB Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE\\_SB\\_CMD\\_MID](#) or [CFE\\_SB\\_SUB\\_RPT\\_CTRL\\_MID](#) received  
on the SB message pipe.  
Definition at line 739 of file cfe\_sb\_eventids.h.

**12.118.2.35 CFE\_SB\_MAX\_DESTS\_MET\_EID** #define CFE\_SB\_MAX\_DESTS\_MET\_EID 9  
SB Subscribe API Max Destinations Exceeded Event ID.

Type: ERROR

Cause:

An SB Subscribe API was called with a message id that already has the maximum allowed number of destinations.  
Definition at line 133 of file cfe\_sb\_eventids.h.

**12.118.2.36 CFE\_SB\_MAX\_MSGS\_MET\_EID** #define CFE\_SB\_MAX\_MSGS\_MET\_EID 8  
SB Subscribe API Max Subscriptions Exceeded Event ID.

Type: ERROR

Cause:

An SB Subscribe API was called on a pipe that already has the maximum allowed number of subscriptions.  
Definition at line 121 of file cfe\_sb\_eventids.h.

**12.118.2.37 CFE\_SB\_MAX\_PIPES\_MET\_EID** #define CFE\_SB\_MAX\_PIPES\_MET\_EID 3  
SB Create Pipe API Max Pipes Exceeded Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_CreatePipe](#) API failure to do maximum number of pipes being exceeded.  
Definition at line 64 of file cfe\_sb\_eventids.h.

**12.118.2.38 CFE\_SB\_MSG\_TOO\_BIG\_EID** #define CFE\_SB\_MSG\_TOO\_BIG\_EID 15  
SB Transmit API Message Size Limit Exceeded Event ID.

Type: ERROR

Cause:

An SB Transmit API was called with a message that is too big.  
Definition at line 199 of file cfe\_sb\_eventids.h.

**12.118.2.39 CFE\_SB\_MSGID\_LIM\_ERR\_EID** #define CFE\_SB\_MSGID\_LIM\_ERR\_EID 17  
SB Transmit API MsgId Pipe Limit Exceeded Event ID.

Type: ERROR

Cause:

An SB Transmit API call failed to deliver the MsgId to a pipe due to the limit for the number of messages with that MsgId for that pipe being exceeded.

Definition at line 222 of file cfe\_sb\_eventids.h.

**12.118.2.40 CFE\_SB\_PART\_SUB\_PKT\_EID** #define CFE\_SB\_PART\_SUB\_PKT\_EID 45  
SB Send Previous Subscriptions Command Partial Packet Sent Event ID.

Type: DEBUG

Cause:

[SB Send Previous Subscriptions Command](#) processing sent a partial subscription packet.

Definition at line 496 of file cfe\_sb\_eventids.h.

**12.118.2.41 CFE\_SB\_PIPE\_ADDED\_EID** #define CFE\_SB\_PIPE\_ADDED\_EID 5  
SB Create Pipe API Success Event ID.

Type: DEBUG

Cause:

[CFE\\_SB\\_CreatePipe](#) API successfully completed.

Definition at line 86 of file cfe\_sb\_eventids.h.

**12.118.2.42 CFE\_SB\_PIPE\_DELETED\_EID** #define CFE\_SB\_PIPE\_DELETED\_EID 47  
SB Pipe Delete API Success Event ID.

Type: DEBUG

Cause:

An SB Delete Pipe API successfully completed.

Definition at line 518 of file cfe\_sb\_eventids.h.

**12.118.2.43 CFE\_SB\_Q\_FULL\_ERR\_EID** #define CFE\_SB\_Q\_FULL\_ERR\_EID 25  
SB Transmit API Pipe Overflow Event ID.

Type: ERROR

Cause:

An SB Transmit API call failed to deliver the Message ID to a pipe due to the pipe queue being full.  
Definition at line 302 of file cfe\_sb\_eventids.h.

**12.118.2.44 CFE\_SB\_Q\_RD\_ERR\_EID** #define CFE\_SB\_Q\_RD\_ERR\_EID 27  
SB Transmit API Queue Read Failure Event ID.

Type: ERROR

Cause:

An SB Transmit API called failed due to a pipe queue read failure.  
Definition at line 324 of file cfe\_sb\_eventids.h.

**12.118.2.45 CFE\_SB\_Q\_WR\_ERR\_EID** #define CFE\_SB\_Q\_WR\_ERR\_EID 26  
SB Transmit API Queue Write Failure Event ID.

Type: ERROR

Cause:

An SB Transmit API call failed due to a pipe queue write failure.  
Definition at line 313 of file cfe\_sb\_eventids.h.

**12.118.2.46 CFE\_SB\_RCV\_BAD\_ARG\_EID** #define CFE\_SB\_RCV\_BAD\_ARG\_EID 18  
SB Receive Buffer API Bad Argument Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_ReceiveBuffer](#) API failure due to a bad input argument.  
Definition at line 233 of file cfe\_sb\_eventids.h.

**12.118.2.47 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\_eventids.h.

**12.118.2.48 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\_eventids.h.

**12.118.2.49 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\_eventids.h.

**12.118.2.50 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\_eventids.h.

**12.118.2.51 CFE\_SB\_SETPipeopts\_ID\_ERR\_EID** #define CFE\_SB\_SETPipeopts\_ID\_ERR\_EID 55  
SB Set Pipe Opt API Invalid Pipe Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_SetPipeOpts](#) API failure due to an invalid pipe ID  
Definition at line 606 of file cfe\_sb\_eventids.h.

**12.118.2.52 CFE\_SB\_SETPipeopts\_Owner\_ERR\_EID** #define CFE\_SB\_SETPipeopts\_Owner\_ERR\_EID 56  
SB Set Pipe Opt API Not Owner Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_SetPipeOpts](#) API failure due to not being the pipe owner.  
Definition at line 617 of file cfe\_sb\_eventids.h.

**12.118.2.53 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\_eventids.h.

**12.118.2.54 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\_eventids.h.

**12.118.2.55 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\_eventids.h.

**12.118.2.56 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\_eventids.h.

**12.118.2.57 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\_eventids.h.

**12.118.2.58 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\_eventids.h.

**12.118.2.59 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\_eventids.h.

**12.118.2.60 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\_eventids.h.

**12.118.2.61 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\_eventids.h.

**12.118.2.62 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\_eventids.h.

**12.118.2.63 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\_eventids.h.

**12.118.2.64 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\_eventids.h.

**12.118.2.65 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\_eventids.h.

## 12.119 cfe/modules/tbl/config/default\_cfe\_tbl\_extern\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_mission_cfg.h"
```

### Data Structures

- struct [CFE\\_TBL\\_File\\_Hdr](#)

*The definition of the header fields that are included in CFE Table Data files.*

## Typedefs

- `typedef uint16 CFE_TBL_BufferSelect_Enum_t`  
*Selects the buffer to operate on for validate or dump commands.*
- `typedef struct CFE_TBL_File_Hdr CFE_TBL_File_Hdr_t`  
*The definition of the header fields that are included in CFE Table Data files.*

## Enumerations

- `enum CFE_TBL_BufferSelect { CFE_TBL_BufferSelect_INACTIVE = 0, CFE_TBL_BufferSelect_ACTIVE = 1 }`  
*Label definitions associated with CFE\_TBL\_BufferSelect\_Enum\_t.*

### 12.119.1 Detailed Description

Declarations and prototypes for cfe\_tbl\_extern\_typedefs module

### 12.119.2 Typedef Documentation

#### 12.119.2.1 CFE\_TBL\_BufferSelect\_Enum\_t `typedef uint16 CFE_TBL_BufferSelect_Enum_t`

Selects the buffer to operate on for validate or dump commands.

See also

`enum CFE_TBL_BufferSelect`

Definition at line 53 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 12.119.2.2 CFE\_TBL\_File\_Hdr\_t `typedef struct CFE_TBL_File_Hdr CFE_TBL_File_Hdr_t`

The definition of the header fields that are included in CFE Table Data files.

This header follows the CFE\_FS header and precedes the actual table data.

Note

The Offset and NumBytes fields in the table header are to 32 bits for backward compatibility with existing CFE versions. This means that even on 64-bit CPUs, individual table files will be limited to 4GiB in size.

### 12.119.3 Enumeration Type Documentation

#### 12.119.3.1 CFE\_TBL\_BufferSelect `enum CFE_TBL_BufferSelect`

Label definitions associated with CFE\_TBL\_BufferSelect\_Enum\_t.

Enumerator

<code>CFE_TBL_BufferSelect_INACTIVE</code>	Select the Inactive buffer for validate or dump.
<code>CFE_TBL_BufferSelect_ACTIVE</code>	Select the Active buffer for validate or dump.

Definition at line 35 of file default\_cfe\_tbl\_extern\_typedefs.h.

## 12.120 cfe/modules/tbl/config/default\_cfe\_tbl\_fcncodes.h File Reference

### Macros

#### Table Services Command Codes

- #define CFE\_TBL\_NOOP\_CC 0
- #define CFE\_TBL\_RESET\_COUNTERS\_CC 1
- #define CFE\_TBL\_LOAD\_CC 2
- #define CFE\_TBL\_DUMP\_CC 3
- #define CFE\_TBL\_VALIDATE\_CC 4
- #define CFE\_TBL\_ACTIVATE\_CC 5
- #define CFE\_TBL\_DUMP\_REGISTRY\_CC 6
- #define CFE\_TBL\_SEND\_REGISTRY\_CC 7
- #define CFE\_TBL\_DELETE\_CDS\_CC 8
- #define CFE\_TBL\_ABORT\_LOAD\_CC 9

### 12.120.1 Detailed Description

Specification for the CFE Event Services (CFE\_TBL) command function codes

#### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

### 12.120.2 Macro Definition Documentation

#### 12.120.2.1 CFE\_TBL\_ABORT\_LOAD\_CC #define CFE\_TBL\_ABORT\_LOAD\_CC 9

**Name** Abort Table Load

#### Description

This command will cause Table Services to discard the contents of a table buffer that was previously loaded with the data in a file as specified by a Table Load command. For single buffered tables, the allocated shared working buffer is freed and becomes available for other Table Load commands.

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_LOADABORT

#### Command Structure

CFE\_TBL\_AbortLoadCmd\_t

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- \$sc\_\$cpu\_TBL\_CMDPC - command execution counter will increment
- The CFE\_TBL\_LOAD\_ABORT\_INF\_EID informational event message is generated
- If the load was aborted for a single buffered table, the \$sc\_\$cpu\_TBL\_NumFreeShrBuf telemetry point should increment

### Error Conditions

This command may fail for the following reason(s):

- The specified table name was not found in the table registry.
- The specified table did not have a load in progress to be aborted.

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_TBL_CMDEC` - command error counter will increment
- Error specific event message

### Criticality

This command will cause the loss of data put into an inactive table buffer.

### See also

[CFE\\_TBL\\_LOAD\\_CC](#), [CFE\\_TBL\\_DUMP\\_CC](#), [CFE\\_TBL\\_VALIDATE\\_CC](#), [CFE\\_TBL\\_ACTIVATE\\_CC](#)

Definition at line 461 of file default\_cfe\_tbl\_fcncodes.h.

## 12.120.2.2 CFE\_TBL\_ACTIVATE\_CC #define CFE\_TBL\_ACTIVATE\_CC 5

### Name

 Activate Table

### Description

This command will cause Table Services to notify a table's owner that an update is pending. The owning application will then update the contents of the active table buffer with the contents of the associated inactive table buffer at a time of their convenience.

### Command Mnemonic(s)

 \$sc\_\$cpu\_TBL\_ACTIVATE

### Command Structure

[CFE\\_TBL\\_ActivateCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_TBL_CMDPC` - command execution counter will increment
- The [CFE\\_TBL\\_UPDATE\\_SUCCESS\\_INF\\_EID](#) informational event message will be generated

### Error Conditions

This command may fail for the following reason(s):

- The specified table name was not found in the table registry.
- The table was registered as a "dump only" type and thus cannot be activated
- The table buffer has not been validated.

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_TBL_CMDEC` - command error counter will increment
- Command specific error event message are issued for all error cases

### Criticality

This command will cause the contents of the specified table to be updated with the contents in the inactive table buffer.

### See also

[CFE\\_TBL\\_LOAD\\_CC](#), [CFE\\_TBL\\_DUMP\\_CC](#), [CFE\\_TBL\\_VALIDATE\\_CC](#), [CFE\\_TBL\\_ABORT\\_LOAD\\_CC](#)

Definition at line 299 of file default\_cfe\_tbl\_fcncodes.h.

### **12.120.2.3 CFE\_TBL\_DELETE\_CDS\_CC #define CFE\_TBL\_DELETE\_CDS\_CC 8**

**Name** Delete Critical Table from Critical Data Store

#### Description

This command will delete the Critical Data Store (CDS) associated with the specified Critical Table. Note that any table still present in the Table Registry is unable to be deleted from the Critical Data Store. All Applications that are accessing the critical table must release and unregister their access before the CDS can be deleted.

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_DeleteCDS

#### Command Structure

[CFE\\_TBL\\_DeleteCDSCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_TBL\\_CDS\\_DELETED\\_INFO\\_EID](#) informational event message will be generated

#### Error Conditions

This command may fail for the following reason(s):

- The specified table name was not found in the critical data store registry
- The specified table name WAS found in the table registry (all registrations/sharing of the table must be unregistered before the table's CDS can be deleted)
- The table's owning application is still active

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDEC](#) - command error counter will increment
- Error specific event message

#### Criticality

This command will cause the loss of the specified table's contents before the owning Application was terminated.

### See also

[CFE\\_ES\\_DUMP\\_CDS\\_REGISTRY\\_CC](#), [CFE\\_ES\\_DELETE\\_CDS\\_CC](#)

Definition at line 422 of file default\_cfe\_tbl\_fcncodes.h.

**12.120.2.4 CFE\_TBL\_DUMP\_CC** #define CFE\_TBL\_DUMP\_CC 3**Name** Dump Table**Description**

This command will cause the Table Services to put the contents of the specified table buffer into the command specified file.

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_DUMP**Command Structure**[CFE\\_TBL\\_DumpCmd\\_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDPC](#) - command execution counter will increment
- Either the [CFE\\_TBL\\_OVERWRITE\\_DUMP\\_INF\\_EID](#) OR the [CFE\\_TBL\\_WRITE\\_DUMP\\_INF\\_EID](#) informational event message will be generated

**Error Conditions**

This command may fail for the following reason(s):

- A single buffered table's inactive buffer was requested to be dumped and no such buffer is currently allocated.
- Error occurred during write operation to file. Possible causes might be insufficient space in the file system or the filename or file path is improperly specified.
- The specified table name was not found in the table registry.

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDEC](#) - command error counter will increment
- A command specific error event message is issued for all error cases

**Criticality**

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

**See also**[CFE\\_TBL\\_LOAD\\_CC](#), [CFE\\_TBL\\_VALIDATE\\_CC](#), [CFE\\_TBL\\_ACTIVATE\\_CC](#), [CFE\\_TBL\\_ABORT\\_LOAD\\_CC](#)

Definition at line 202 of file default\_cfe\_tbl\_fcncodes.h.

**12.120.2.5 CFE\_TBL\_DUMP\_REGISTRY\_CC** #define CFE\_TBL\_DUMP\_REGISTRY\_CC 6**Name** Dump Table Registry**Description**

This command will cause Table Services to write some of the contents of the Table Registry to the command specified file. This allows the operator to see the current state and configuration of all tables that have been registered with the cFE.

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_WriteReg2File**Command Structure**[CFE\\_TBL\\_DumpRegistryCmd\\_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDPC](#) - command execution counter will increment
- The generation of either [CFE\\_TBL\\_OVERWRITE\\_REG\\_DUMP\\_INF\\_EID](#) or [CFE\\_TBL\\_WRITE\\_REG\\_DUMP\\_INF\\_EID](#) debug event messages
- The specified file should appear (or be updated) at the specified location in the file system

**Error Conditions**

This command may fail for the following reason(s):

- A table registry dump is already in progress, not yet completed
- The specified DumpFilename could not be parsed
- Error occurred during write operation to file. Possible causes might be insufficient space in the file system or the filename or file path is improperly specified.

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDEC](#) - command error counter will increment
- An Error specific event message

**Criticality**

This command is not inherently dangerous. It will create a new file in the file system and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

**See also**[CFE\\_TBL\\_SEND\\_REGISTRY\\_CC](#)

Definition at line 343 of file default\_cfe\_tbl\_fcncodes.h.

**12.120.2.6 CFE\_TBL\_LOAD\_CC #define CFE\_TBL\_LOAD\_CC 2****Name** Load Table**Description**

This command loads the contents of the specified file into an inactive buffer for the table specified within the file.

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_Load**Command Structure**[CFE\\_TBL\\_LoadCmd\\_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_TBL\\_FILE\\_LOADED\\_INF\\_EID](#) informational event message will be generated

**Error Conditions**

This command can fail for the following reasons:

- Table name found in table image file's table header is not found in table registry (ie - The table associated with the table image in the file has not been registered by an application).
- The table image file has an invalid or incorrect size. The size of the image file must match the size field within in the header, and must also match the expected size of the table indicated in the registry.
- No working buffers are available for the load. This would indicate that too many single-buffered table loads are in progress at the same time.
- An attempt is being made to load an uninitialized table with a file containing only a partial table image.
- The table image file was unable to be opened. Either the file does not exist at the specified location, the filename is in error, or the file system has been corrupted.

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDEC](#) - command error counter will increment
- Command specific error event messages are issued for all error cases

**Criticality**

This command is not inherently dangerous. It is performing the first step of loading a table and can be aborted (using the Abort Table Load command described below) without affecting the contents of the active table image.

**See also**[CFE\\_TBL\\_DUMP\\_CC](#), [CFE\\_TBL\\_VALIDATE\\_CC](#), [CFE\\_TBL\\_ACTIVATE\\_CC](#), [CFE\\_TBL\\_ABORT\\_LOAD\\_CC](#)

Definition at line 159 of file default\_cfe\_tbl\_fcncodes.h.

**12.120.2.7 CFE\_TBL\_NOOP\_CC** #define CFE\_TBL\_NOOP\_CC 0**Name** Table No-Op**Description**

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Table Services task.

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_NOOP**Command Structure**

[CFE\\_TBL\\_NoopCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TBL\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_TBL\\_NOOP\\_INF\\_EID](#) informational event message will be generated

**Error Conditions**

There are no error conditions for this command. If the Table Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

**Criticality**

None

**See also**

Definition at line 68 of file default\_cfe\_tbl\_fcncodes.h.

**12.120.2.8 CFE\_TBL\_RESET\_COUNTERS\_CC** #define CFE\_TBL\_RESET\_COUNTERS\_CC 1**Name** Table Reset Counters**Description**

This command resets the following counters within the Table Services housekeeping telemetry:

- Command Execution Counter (\$sc\_\$cpu\_TBL\_CMDPC)
- Command Error Counter (\$sc\_\$cpu\_TBL\_CMDEC)
- Successful Table Validations Counter (\$sc\_\$cpu\_TBL\_ValSuccessCtr)
- Failed Table Validations Counter (\$sc\_\$cpu\_TBL\_ValFailedCtr)
- Number of Table Validations Requested (\$sc\_\$cpu\_TBL\_ValReqCtr)
- Number of completed table validations (\$sc\_\$cpu\_TBL\_ValCompldCtr)

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_ResetCtrs

#### Command Structure

[CFE\\_TBL\\_ResetCountersCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_TBL_CMDPC` - command execution counter will be reset to 0
- The [CFE\\_TBL\\_RESET\\_INF\\_EID](#) debug event message will be generated

#### Error Conditions

There are no error conditions for this command. If the Table Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

#### Criticality

This command is not inherently dangerous. However, it is possible for ground systems and on-board safing procedures to be designed such that they react to changes in the counter values that are reset by this command.

#### See also

Definition at line 109 of file default\_cfe\_tbl\_fcncodes.h.

### 12.120.2.9 CFE\_TBL\_SEND\_REGISTRY\_CC #define CFE\_TBL\_SEND\_REGISTRY\_CC 7

**Name** Telemeter One Table Registry Entry

#### Description

This command will cause Table Services to telemeter the contents of the Table Registry for the command specified table.

**Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_TLMReg

#### Command Structure

[CFE\\_TBL\\_SendRegistryCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- `$sc_$cpu_TBL_CMDPC` - command execution counter will increment
- Receipt of a Table Registry Info Packet (see [CFE\\_TBL\\_TableRegistryTlm\\_t](#))
- The [CFE\\_TBL\\_TLM\\_REG\\_CMD\\_INF\\_EID](#) debug event message will be generated

## Error Conditions

This command may fail for the following reason(s):

- The specified table name was not found in the table registry.

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_TBL\_CMDEC** - command error counter will increment
- Error specific event message

## Criticality

This command is not inherently dangerous. It will generate additional telemetry.

## See also

[CFE\\_TBL\\_DUMP\\_REGISTRY\\_CC](#)

Definition at line 378 of file default\_cfe\_tbl\_fcncodes.h.

### **12.120.2.10 CFE\_TBL\_VALIDATE\_CC** #define CFE\_TBL\_VALIDATE\_CC 4

#### **Name** Validate Table

#### Description

This command will cause Table Services to calculate the Data Integrity Value for the specified table and to notify the owning application that the table's validation function should be executed. The results of both the Data Integrity Value computation and the validation function are reported in Table Services Housekeeping Telemetry.

#### **Command Mnemonic(s)** \$sc\_\$cpu\_TBL\_VALIDATE

#### Command Structure

[CFE\\_TBL\\_ValidateCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_TBL\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_TBL\_ValReqCtr** - table validation request counter will increment
- **\$sc\_\$cpu\_TBL\_LastValCRC** - calculated data integrity value will be updated
- The [CFE\\_TBL\\_VAL\\_REQ\\_MADE\\_INF\\_EID](#) debug event message (indicating the application is being notified of a validation request)

If the specified table has an associated validation function, then the following telemetry will also change:

- Either **\$sc\_\$cpu\_TBL\_ValSuccessCtr** OR **\$sc\_\$cpu\_TBL\_ValFailedCtr** will increment
- **\$sc\_\$cpu\_TBL\_ValCompltdCtr** - table validations performed counter will increment
- **\$sc\_\$cpu\_TBL\_LastVals** - table validation function return status will update
- The [CFE\\_TBL\\_VALIDATION\\_INF\\_EID](#) informational event message (indicating the validation function return status) will be generated

### Error Conditions

This command may fail for the following reason(s):

- A single buffered table's inactive buffer was requested to be validated and no such buffer is currently allocated.
- Too many validations have been requested simultaneously. The operator must wait for one or more applications to perform their table validation functions before trying again.
- The specified table name was not found in the table registry.

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_TBL_CMDEC` - command error counter will increment
- Command specific error event message are issued for all error cases

### Criticality

The success or failure of a table validation does not have any immediate impact on table contents. The results are sent to the operator in telemetry and the operator must determine whether the results are acceptable and send a command to activate the validated table image.

### See also

[CFE\\_TBL\\_LOAD\\_CC](#), [CFE\\_TBL\\_DUMP\\_CC](#), [CFE\\_TBL\\_ACTIVATE\\_CC](#), [CFE\\_TBL\\_ABORT\\_LOAD\\_CC](#)

Definition at line 259 of file default\_cfe\_tbl\_fcncodes.h.

## 12.121 cfe/modules/tbl/config/default\_cfe\_tbl\_interface\_cfg.h File Reference

### Macros

- `#define CFE_MISSION_TBL_MAX_NAME_LENGTH 16`
- `#define CFE_MISSION_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_LEN + 4)`

### 12.121.1 Detailed Description

CFE Table Services (CFE\_TBL) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.121.2 Macro Definition Documentation

**12.121.2.1 CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN** `#define CFE_MISSION_TBL_MAX_FULL_NAME_L←  
EN (CFE_MISSION_TBL_MAX_NAME_LENGTH + CFE_MISSION_MAX_API_LEN + 4)`

**Purpose** Maximum Length of Full Table Name in messages

**Description:**

Indicates the maximum length (in characters) of the entire table name within software bus messages, in "App←Name.TableName" notation.

This affects the layout of command/telemetry messages but does not affect run time behavior or internal allocation.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 69 of file default\_cfe\_tbl\_interface\_cfg.h.

**12.121.2.2 CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH #define CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH 16****Purpose** Maximum Table Name Length**Description:**

Indicates the maximum length (in characters) of the table name ('TblName') portion of a Full Table Name of the following form: "ApplicationName.TblName"

This length does not need to include an extra character for NULL termination.

**Limits**

This value should be kept as a multiple of 4, to maintain alignment of any possible neighboring fields without implicit padding.

Definition at line 49 of file default\_cfe\_tbl\_interface\_cfg.h.

**12.122 cfe/modules/tbl/config/default\_cfe\_tbl\_internal\_cfg.h File Reference****Macros**

- #define CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY 70
- #define CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES 524288
- #define CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE 16384
- #define CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE 16384
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128
- #define CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES 32
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256
- #define CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS 4
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS 10
- #define CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE "/ram/cfe\_tbl\_reg.log"
- #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT 0
- #define CFE\_PLATFORM\_TBL\_U32FROM4CHARS(\_C1, \_C2, \_C3, \_C4) ((uint32)(\_C1) << 24 | (uint32)(\_C2) << 16 | (uint32)(\_C3) << 8 | (uint32)(\_C4))
- #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 (0x42)
- #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT 0
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 (1)
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 0
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 0

### 12.122.1 Detailed Description

CFE Table Services (CFE\_TBL) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.122.2 Macro Definition Documentation

**12.122.2.1 CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES** #define CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYT←  
ES 524288

**Purpose** Size of Table Services Table Memory Pool

#### Description:

Defines the TOTAL size of the memory pool that cFE Table Services allocates from the system. The size must be large enough to provide memory for each registered table, the inactive buffers for double buffered tables and for the shared inactive buffers for single buffered tables.

#### Limits

The cFE does not place a limit on the size of this parameter.

Definition at line 75 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.2 CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE** #define CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_D←  
UMP\_FILE "/ram/cfe\_tbl\_reg.log"

**Purpose** Default Filename for a Table Registry Dump

#### Description:

Defines the file name used to store the table registry when no filename is specified in the dump registry command.

#### Limits

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 189 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.3 CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TA←  
BLES 32

**Purpose** Maximum Number of Critical Tables that can be Registered

**Description:**

Defines the maximum number of critical tables supported by this processor's Table Services.

**Limits**

This number must be less than 32767. It should be recognized that this parameter determines the size of the Critical Table Registry which is maintained in the Critical Data Store. An excessively high number will waste Critical Data Store memory. Therefore, this number must not exceed the value defined in CFE\_ES\_CDS\_MAX\_CRITICAL\_TABLES.

Definition at line 130 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.4 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\_B

Definition at line 87 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.5 CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256

**Purpose** Maximum Number of Table Handles**Description:**

Defines the maximum number of Table Handles.

**Limits**

This number must be less than 32767. This number must be at least as big as the number of tables (CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES) and should be set higher if tables are shared between applications.

Definition at line 143 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.6 CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128

**Purpose** Maximum Number of Tables Allowed to be Registered**Description:**

Defines the maximum number of tables supported by this processor's Table Services.

**Limits**

This number must be less than 32767. It should be recognized that this parameter determines the size of the Table Registry. An excessively high number will waste memory.

Definition at line 116 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.7 CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS 10

**Purpose** Maximum Number of Simultaneous Table Validations

**Description:**

Defines the maximum number of pending validations that the Table Services can handle at any one time. When a table has a validation function, a validation request is made of the application to perform that validation. This number determines how many of those requests can be outstanding at any one time.

**Limits**

This number must be less than 32767. An excessively high number will degrade system performance and waste memory. A number less than 20 is suggested but not required.

Definition at line 176 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.8 CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS** #define CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS 4

**Purpose** Maximum Number of Simultaneous Loads to Support

**Description:**

Defines the maximum number of single buffered tables that can be loaded simultaneously. This number is used to determine the number of shared buffers to allocate.

**Limits**

This number must be less than 32767. An excessively high number will degrade system performance and waste memory. A number less than 5 is suggested but not required.

Definition at line 158 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.9 CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE** #define CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE 16384

**Purpose** Maximum Size Allowed for a Single Buffered Table

**Description:**

Defines the maximum allowed size (in bytes) of a single buffered table. **NOTE:** This size determines the size of all shared table buffers. Therefore, this size will be multiplied by [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) below when allocating memory for shared tables.

**Limits**

The cFE does not place a limit on the size of this parameter but it must be small enough to allow for [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) number of tables to fit into [CFE\\_PLATFORM\\_TBL\\_BUF\\_MEMORY\\_BYT](#)

Definition at line 103 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.10 CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TBL\_START\_TASK\_PRIO←  
RITY 70

**Purpose** Define TBL Task Priority

**Description:**

Defines the cFE\_TBL Task priority.

**Limits**

Not Applicable

Definition at line 44 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.11 CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TBL\_START\_TASK\_S←  
TACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define TBL Task Stack Size

**Description:**

Defines the cFE\_TBL Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 59 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.12 CFE\_PLATFORM\_TBL\_U32FROM4CHARS** #define CFE\_PLATFORM\_TBL\_U32FROM4CHARS (←  
\_C1,  
\_C2,  
\_C3,  
\_C4 ) ((uint32) (\_C1) << 24 | (uint32) (\_C2) << 16 | (uint32) (\_C3) << 8 | (uint32) (←  
\_C4))

Definition at line 211 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.13 CFE\_PLATFORM\_TBL\_VALID\_PRID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 (1)

**Purpose** Processor ID values used for table load validation

**Description:**

Defines the processor ID values used for validating the processor ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

**Limits**

This value can be any 32 bit unsigned integer.

Definition at line 260 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.14 CFE\_PLATFORM\_TBL\_VALID\_PRID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CH  
'b', 'c', 'd'))

Definition at line 261 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.15 CFE\_PLATFORM\_TBL\_VALID\_PRID\_3** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 0

Definition at line 262 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.16 CFE\_PLATFORM\_TBL\_VALID\_PRID\_4** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 0

Definition at line 263 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.17 CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT 0

**Purpose** Number of Processor ID's specified for validation

**Description:**

Defines the number of specified processor ID values that are verified during table loads. If the number is zero then no validation of the processor ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of processor ID's defined below are compared to the processor ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified processor ID values.

**Limits**

This number must be greater than or equal to zero and less than or equal to 4.

Definition at line 246 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.18 CFE\_PLATFORM\_TBL\_VALID\_SCID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 (0x42)

**Purpose** Spacecraft ID values used for table load validation

**Description:**

Defines the spacecraft ID values used for validating the spacecraft ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

**Limits**

This value can be any 32 bit unsigned integer.

Definition at line 226 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.19 CFE\_PLATFORM\_TBL\_VALID\_SCID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CH  
'b', 'c', 'd'))

Definition at line 227 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.122.2.20 CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT 0

**Purpose** Number of Spacecraft ID's specified for validation

**Description:**

Defines the number of specified spacecraft ID values that are verified during table loads. If the number is zero then no validation of the spacecraft ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of spacecraft ID's defined below are compared to the spacecraft ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified spacecraft ID values.

**Limits**

This number must be greater than or equal to zero and less than or equal to 2.

Definition at line 208 of file default\_cfe\_tbl\_internal\_cfg.h.

**12.123 cfe/modules/tbl/config/default\_cfe\_tbl\_mission\_cfg.h File Reference**

```
#include "cfe_tbl_interface_cfg.h"
```

**12.123.1 Detailed Description**

CFE Event Services (CFE\_TBL) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

**Note**

This file may be overridden/superceded by mission-provided defintions either by overriding this header or by generating definitions from a command/data dictionary tool.

**12.124 cfe/modules/tbl/config/default\_cfe\_tbl\_msg.h File Reference**

```
#include "cfe_mission_cfg.h"
#include "cfe_tbl_msgdefs.h"
#include "cfe_tbl_msgstruct.h"
```

**12.124.1 Detailed Description**

Specification for the CFE Event Services (CFE\_TBL) command and telemetry message data types.

This is a compatibility header for the "cfe\_tbl\_msg.h" file that has traditionally provided the message definitions for cFS apps.

**Note**

This file may be overridden/superceded by mission-provided defintions either by overriding this header or by generating definitions from a command/data dictionary tool.

**12.125 cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h File Reference**

```
#include "cfe_tbl_fcncodes.h"
```

### 12.125.1 Detailed Description

Specification for the CFE Event Services (CFE\_TBL) command and telemetry message constant definitions.  
For CFE\_TBL this is only the function/command code definitions

## 12.126 cfe/modules/tbl/config/default\_cfe\_tbl\_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_tbl_topicids.h"
```

### Macros

- #define CFE\_TBL\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TBL\_CMD\_MSG /\* 0x1804 \*/
- #define CFE\_TBL\_SEND\_HK\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TBL\_SEND\_HK\_MSG /\* 0x180C \*/
- #define CFE\_TBL\_HK\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_TBL\_HK\_TLM\_MSG /\* 0x0804 \*/
- #define CFE\_TBL\_REG\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_TBL\_REG\_TLM\_MSG /\* 0x080C \*/

### 12.126.1 Detailed Description

CFE Event Services (CFE\_TBL) Application Message IDs

### 12.126.2 Macro Definition Documentation

**12.126.2.1 CFE\_TBL\_CMD\_MID** #define CFE\_TBL\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TBL\_CMD\_MSG  
/\* 0x1804 \*/

Definition at line 32 of file default\_cfe\_tbl\_msgids.h.

**12.126.2.2 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 38 of file default\_cfe\_tbl\_msgids.h.

**12.126.2.3 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 39 of file default\_cfe\_tbl\_msgids.h.

**12.126.2.4 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 33 of file default\_cfe\_tbl\_msgids.h.

## 12.127 cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_tbl_msgdefs.h"
```

```
#include "cfe_msg_hdr.h"
```

## Data Structures

- struct [CFE\\_TBL\\_NoArgsCmd](#)  
*Generic "no arguments" command.*
- struct [CFE\\_TBL\\_LoadCmd\\_Payload](#)  
*Load Table Command Payload.*
- struct [CFE\\_TBL\\_LoadCmd](#)  
*Load Table Command.*
- struct [CFE\\_TBL\\_DumpCmd\\_Payload](#)  
*Dump Table Command Payload.*
- struct [CFE\\_TBL\\_DumpCmd](#)
- struct [CFE\\_TBL\\_ValidateCmd\\_Payload](#)  
*Validate Table Command Payload.*
- struct [CFE\\_TBL\\_ValidateCmd](#)  
*Validate Table Command.*
- struct [CFE\\_TBL\\_ActivateCmd\\_Payload](#)  
*Activate Table Command Payload.*
- struct [CFE\\_TBL\\_ActivateCmd](#)  
*Activate Table Command.*
- struct [CFE\\_TBL\\_DumpRegistryCmd\\_Payload](#)  
*Dump Registry Command Payload.*
- struct [CFE\\_TBL\\_DumpRegistryCmd](#)  
*Dump Registry Command.*
- struct [CFE\\_TBL\\_SendRegistryCmd\\_Payload](#)  
*Send Table Registry Command Payload.*
- struct [CFE\\_TBL\\_SendRegistryCmd](#)  
*Send Table Registry Command.*
- struct [CFE\\_TBL\\_DeleteCDSCmd\\_Payload](#)  
*Delete Critical Table CDS Command Payload.*
- struct [CFE\\_TBL\\_DeleteCDSCmd](#)  
*Delete Critical Table CDS Command.*
- struct [CFE\\_TBL\\_AbortLoadCmd\\_Payload](#)  
*Abort Load Command Payload.*
- struct [CFE\\_TBL\\_AbortLoadCmd](#)  
*Abort Load Command.*
- struct [CFE\\_TBL\\_NotifyCmd\\_Payload](#)  
*Table Management Notification Command Payload.*
- struct [CFE\\_TBL\\_NotifyCmd](#)
- struct [CFE\\_TBL\\_HousekeepingTlm\\_Payload](#)
- struct [CFE\\_TBL\\_HousekeepingTlm](#)
- struct [CFE\\_TBL\\_TblRegPacket\\_Payload](#)
- struct [CFE\\_TBL\\_TableRegistryTlm](#)

## 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 CFE_TBL_NoArgsCmd_t CFE_TBL_SendHkCmd_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`

### 12.127.1 Detailed Description

Purpose: cFE Executive Services (TBL) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide

Notes:

### 12.127.2 Typedef Documentation

**12.127.2.1 CFE\_TBL\_AbortLoadCmd\_Payload\_t** `typedef struct CFE_TBL_AbortLoadCmd_Payload CFE_TBL_AbortLoadCmd_Pay`  
Abort Load Command Payload.

For command details, see [CFE\\_TBL\\_ABORT\\_LOAD\\_CC](#)

**12.127.2.2 CFE\_TBL\_AbortLoadCmd\_t** `typedef struct CFE_TBL_AbortLoadCmd CFE_TBL_AbortLoadCmd_t`  
Abort Load Command.

**12.127.2.3 CFE\_TBL\_ActivateCmd\_Payload\_t** `typedef struct CFE_TBL_ActivateCmd_Payload CFE_TBL_ActivateCmd_Pay`  
Activate Table Command Payload.

For command details, see [CFE\\_TBL\\_ACTIVATE\\_CC](#)

**12.127.2.4 CFE\_TBL\_ActivateCmd\_t** `typedef struct CFE_TBL_ActivateCmd CFE_TBL_ActivateCmd_t`  
Activate Table Command.

**12.127.2.5 CFE\_TBL\_DelCDSCmd\_Payload\_t** `typedef struct CFE_TBL_DelCDSCmd_Payload CFE_TBL_DelCDSCmd_Pay`  
Delete Critical Table CDS Command Payload.

For command details, see [CFE\\_TBL\\_DELETE\\_CDS\\_CC](#)

**12.127.2.6 CFE\_TBL\_DeleteCDSCmd\_t** `typedef struct CFE_TBL_DeleteCDSCmd CFE_TBL_DeleteCDSCmd_t`  
Delete Critical Table CDS Command.

**12.127.2.7 CFE\_TBL\_DumpCmd\_Payload\_t** `typedef struct CFE_TBL_DumpCmd_Payload CFE_TBL_DumpCmd_Pay`  
Dump Table Command Payload.

For command details, see [CFE\\_TBL\\_DUMP\\_CC](#)

**12.127.2.8 CFE\_TBL\_DumpCmd\_t** `typedef struct CFE_TBL_DumpCmd CFE_TBL_DumpCmd_t`  
/brief Dump Table Command

**12.127.2.9 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](#)

**12.127.2.10 CFE\_TBL\_DumpRegistryCmd\_t** `typedef struct CFE_TBL_DumpRegistryCmd CFE_TBL_DumpRegistryCmd_t`  
Dump Registry Command.

**12.127.2.11 CFE\_TBL\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_TBL_HousekeepingTlm_Payload CFE_TBL_HousekeepingTlm_Payload_t`

**Name** Table Services Housekeeping Packet

**12.127.2.12 CFE\_TBL\_HousekeepingTlm\_t** `typedef struct CFE_TBL_HousekeepingTlm CFE_TBL_HousekeepingTlm_t`

**12.127.2.13 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](#)

**12.127.2.14 CFE\_TBL\_LoadCmd\_t** `typedef struct CFE_TBL_LoadCmd CFE_TBL_LoadCmd_t`  
Load Table Command.

**12.127.2.15 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](#))

**12.127.2.16 CFE\_TBL\_NoopCmd\_t** `typedef CFE_TBL_NoArgsCmd_t CFE_TBL_NoopCmd_t`  
Definition at line 64 of file default\_cfe\_tbl\_msgstruct.h.

**12.127.2.17 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).

**12.127.2.18 CFE\_TBL\_NotifyCmd\_t** `typedef struct CFE_TBL_NotifyCmd CFE_TBL_NotifyCmd_t`  
/brief Table Management Notification Command

**12.127.2.19 CFE\_TBL\_ResetCountersCmd\_t** `typedef CFE_TBL_NoArgsCmd_t CFE_TBL_ResetCountersCmd_t`  
Definition at line 65 of file default\_cfe\_tbl\_msgstruct.h.

**12.127.2.20 CFE\_TBL\_SendHkCmd\_t** `typedef CFE_TBL_NoArgsCmd_t CFE_TBL_SendHkCmd_t`  
Definition at line 66 of file default\_cfe\_tbl\_msgstruct.h.

**12.127.2.21 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](#)

**12.127.2.22 CFE\_TBL\_SendRegistryCmd\_t** `typedef struct CFE_TBL_SendRegistryCmd CFE_TBL_SendRegistryCmd_t`  
Send Table Registry Command.

**12.127.2.23 CFE\_TBL\_TableRegistryTlm\_t** `typedef struct CFE_TBL_TableRegistryTlm CFE_TBL_TableRegistryTlm_t`

**12.127.2.24 CFE\_TBL\_TblRegPacket\_Payload\_t** `typedef struct CFE_TBL_TblRegPacket_Payload CFE_TBL_TblRegPacket_Pay`  
**Name** Table Registry Info Packet

**12.127.2.25 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](#)

**12.127.2.26 CFE\_TBL\_ValidateCmd\_t** `typedef struct CFE_TBL_ValidateCmd CFE_TBL_ValidateCmd_t`  
Validate Table Command.

## 12.128 cfe/modules/tbl/config/default\_cfe\_tbl\_platform\_cfg.h File Reference

```
#include "cfe_tbl_mission_cfg.h"
#include "cfe_tbl_internal_cfg.h"
```

### 12.128.1 Detailed Description

CFE Table Services (CFE\_TBL) Application Platform Configuration Header File  
This is a compatibility header for the "platform\_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.  
These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.129 cfe/modules/tbl/config/default\_cfe\_tbl\_topicids.h File Reference

### Macros

- `#define CFE_MISSION_TBL_CMD_MSG 4`
- `#define CFE_MISSION_TBL_SEND_HK_MSG 12`
- `#define CFE_MISSION_TBL_HK_TLM_MSG 4`
- `#define CFE_MISSION_TBL_REG_TLM_MSG 12`

### 12.129.1 Detailed Description

CFE Table Services (CFE\_TBL) Application Topic IDs

## 12.129.2 Macro Definition Documentation

### 12.129.2.1 CFE\_MISSION\_TBL\_CMD\_MSG #define CFE\_MISSION\_TBL\_CMD\_MSG 4

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE command messages

**Limits**

Not Applicable

Definition at line 35 of file default\_cfe\_tbl\_topicids.h.

### 12.129.2.2 CFE\_MISSION\_TBL\_HK\_TLM\_MSG #define CFE\_MISSION\_TBL\_HK\_TLM\_MSG 4

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE telemetry messages

**Limits**

Not Applicable

Definition at line 47 of file default\_cfe\_tbl\_topicids.h.

### 12.129.2.3 CFE\_MISSION\_TBL\_REG\_TLM\_MSG #define CFE\_MISSION\_TBL\_REG\_TLM\_MSG 12

Definition at line 48 of file default\_cfe\_tbl\_topicids.h.

### 12.129.2.4 CFE\_MISSION\_TBL\_SEND\_HK\_MSG #define CFE\_MISSION\_TBL\_SEND\_HK\_MSG 12

Definition at line 36 of file default\_cfe\_tbl\_topicids.h.

## 12.130 cfe/modules/tbl/fsw/inc/cfe\_tbl\_eventids.h File Reference

### Macros

#### TBL event IDs

- #define CFE\_TBL\_INIT\_INF\_EID 1  
*TB Initialization Event ID.*
- #define CFE\_TBL\_NOOP\_INF\_EID 10  
*TBL No-op Command Success Event ID.*
- #define CFE\_TBL\_RESET\_INF\_EID 11  
*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_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

- #define `CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID` 66  
*TBL Write Table File Data Failure Event ID.*
- #define `CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID` 67  
*TBL Validate Or Write Table Command No Inactive Buffer Event ID.*
- #define `CFE_TBL_WRITE_TBL_REG_ERR_EID` 68  
*TBL Validate Table Command Result Storage Exceeded Event ID.*
- #define `CFE_TBL_LOAD_ABORT_ERR_EID` 69  
*TBL Write Table Registry File Data Failure Event ID.*
- #define `CFE_TBL_ACTIVATE_ERR_EID` 70  
*TBL Abort Table Load No Load Started Event ID.*
- #define `CFE_TBL_FILE_INCOMPLETE_ERR_EID` 71  
*TBL Activate Table Command No Inactive Buffer 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.*

### 12.130.1 Detailed Description

cFE Table Services Event IDs

### 12.130.2 Macro Definition Documentation

**12.130.2.1 CFE\_TBL\_ACTIVATE\_DUMP\_ONLY\_ERR\_EID** #define [CFE\\_TBL\\_ACTIVATE\\_DUMP\\_ONLY\\_ERR\\_EID](#) 78  
TBL Activate Table Command For Dump Only Table Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to table being dump only.

Definition at line 544 of file cfe\_tbl\_eventids.h.

**12.130.2.2 CFE\_TBL\_ACTIVATE\_ERR\_EID** #define [CFE\\_TBL\\_ACTIVATE\\_ERR\\_EID](#) 70  
TBL Activate Table Command No Inactive Buffer Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to no associated inactive buffer.

Definition at line 450 of file cfe\_tbl\_eventids.h.

**12.130.2.3 CFE\_TBL\_ASSUMED\_VALID\_INF\_EID** #define CFE\_TBL\_ASSUMED\_VALID\_INF\_EID 23  
TBL Validate Table Valid Due To No Validation Function Event ID.

Type: INFORMATION

Cause:

[TBL Validate Table Command](#) marking table as valid due to no validation function being registered.  
Definition at line 180 of file cfe\_tbl\_eventids.h.

**12.130.2.4 CFE\_TBL\_CC1\_ERR\_EID** #define CFE\_TBL\_CC1\_ERR\_EID 51  
TBL Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID [CFE\\_TBL\\_CMD\\_MID](#) received on the TBL message pipe.  
Definition at line 246 of file cfe\_tbl\_eventids.h.

**12.130.2.5 CFE\_TBL\_CDS\_DELETE\_ERR\_EID** #define CFE\_TBL\_CDS\_DELETE\_ERR\_EID 86  
TBL Delete Table CDS Command Internal Error Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to an internal error. See the system log for more information.  
Definition at line 640 of file cfe\_tbl\_eventids.h.

**12.130.2.6 CFE\_TBL\_CDS\_DELETED\_INFO\_EID** #define CFE\_TBL\_CDS\_DELETED\_INFO\_EID 38  
TBL Delete Table CDS Command Success Event ID.

Type: INFORMATION

Cause:

[TBL Delete Table CDS Command](#) success.  
Definition at line 224 of file cfe\_tbl\_eventids.h.

**12.130.2.7 CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID** #define CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID 85  
TBL Delete Table CDS Command Not In CDS Registry Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table name not found in the CDS registry.  
Definition at line 628 of file cfe\_tbl\_eventids.h.

**12.130.2.8 CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID** #define CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID 87  
TBL Delete Table CDS Command App Active Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the owning application being active.  
Definition at line 652 of file cfe\_tbl\_eventids.h.

**12.130.2.9 CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID** #define CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID 62  
TBL Write File Creation Failure Event ID.

Type: ERROR

Cause:

TBL Write Table or Table Registry File failed to create file. OVERLOADED  
Definition at line 357 of file cfe\_tbl\_eventids.h.

**12.130.2.10 CFE\_TBL\_DUMP\_PENDING\_ERR\_EID** #define CFE\_TBL\_DUMP\_PENDING\_ERR\_EID 77  
TBL Write Table Command Already In Progress Event ID.

Type: ERROR

Cause:

[TBL Write Table Command](#) failure due to a dump already in progress for the same table.  
Definition at line 532 of file cfe\_tbl\_eventids.h.

**12.130.2.11 CFE\_TBL\_FAIL\_HK\_SEND\_ERR\_EID** #define CFE\_TBL\_FAIL\_HK\_SEND\_ERR\_EID 56  
TBL Send Housekeeping Command Transmit Failure Event ID.

Type: ERROR

Cause:

[TBL Send Housekeeping Command](#) failure transmitting the housekeeping message.  
Definition at line 302 of file cfe\_tbl\_eventids.h.

**12.130.2.12 CFE\_TBL\_FAIL\_NOTIFY\_SEND\_ERR\_EID** #define CFE\_TBL\_FAIL\_NOTIFY\_SEND\_ERR\_EID 89  
TBL Send Notification Transmit Failed Event ID.

Type: ERROR

Cause:

TBL send notification transmit message failure.  
Definition at line 674 of file cfe\_tbl\_eventids.h.

**12.130.2.13 CFE\_TBL\_FILE\_ACCESS\_ERR\_EID** #define CFE\_TBL\_FILE\_ACCESS\_ERR\_EID 53  
TBL Load Table File Open Failure Event ID.

Type: ERROR

Cause:

Load Table failure opening the file. OVERLOADED  
Definition at line 268 of file cfe\_tbl\_eventids.h.

**12.130.2.14 CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID** #define CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID 71  
TBL Load Table Incomplete Load Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to inability to read the size of data specified in the table header from file. OVERLOADED  
Definition at line 462 of file cfe\_tbl\_eventids.h.

**12.130.2.15 CFE\_TBL\_FILE\_LOADED\_INF\_EID** #define CFE\_TBL\_FILE\_LOADED\_INF\_EID 12  
TBL Load Table Command Success Event ID.

Type: INFORMATION

Cause:

[TBL Load Table Command](#) successfully loaded the new table data to the working buffer.  
Definition at line 76 of file cfe\_tbl\_eventids.h.

**12.130.2.16 CFE\_TBL\_FILE\_STD\_HDR\_ERR\_EID** #define CFE\_TBL\_FILE\_STD\_HDR\_ERR\_EID 54  
TBL Load Table File Read Standard Header Failure Event ID.

Type: ERROR

Cause:

Load Table failure reading the file standard header.  
Definition at line 279 of file cfe\_tbl\_eventids.h.

**12.130.2.17 CFE\_TBL\_FILE\_SUBTYPE\_ERR\_EID** #define CFE\_TBL\_FILE\_SUBTYPE\_ERR\_EID 59  
TBL Load Table Invalid File Subtype Event ID.

Type: ERROR

Cause:

TBL Load Table Failure due to invalid file subtype.  
Definition at line 335 of file cfe\_tbl\_eventids.h.

**12.130.2.18 CFE\_TBL\_FILE\_TBL\_HDR\_ERR\_EID** #define CFE\_TBL\_FILE\_TBL\_HDR\_ERR\_EID 55  
TBL Load Table File Read Table Header Failure Event ID.

Type: ERROR

Cause:

Load Table failure reading the file table header.  
Definition at line 290 of file cfe\_tbl\_eventids.h.

**12.130.2.19 CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID** #define CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID 75  
TBL Load Table File Excess Data Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified size of data being smaller than the actual data contained in the file. OVERLOADED

Definition at line 508 of file cfe\_tbl\_eventids.h.

**12.130.2.20 CFE\_TBL\_FILE\_TYPE\_ERR\_EID** #define CFE\_TBL\_FILE\_TYPE\_ERR\_EID 58  
TBL Load Table Invalid File Content ID Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to invalid file content ID.

Definition at line 324 of file cfe\_tbl\_eventids.h.

**12.130.2.21 CFE\_TBL\_HANDLE\_ACCESS\_ERR\_EID** #define CFE\_TBL\_HANDLE\_ACCESS\_ERR\_EID 103  
TBL Load Table API Access Violation Event ID.

Type: ERROR

Cause:

[CFE\\_TBL\\_Load](#) API failure due to the application not owning the table.

Definition at line 817 of file cfe\_tbl\_eventids.h.

**12.130.2.22 CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID** #define CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID 80  
TBL Validate Or Write Table Command Invalid Buffer Event ID.

Type: ERROR

Cause:

[TBL Validate Table Command](#) or [TBL Write Table Command](#) failure due to an invalid buffer selection. OVERLOADED  
Definition at line 568 of file cfe\_tbl\_eventids.h.

**12.130.2.23 CFE\_TBL\_IN\_REGISTRY\_ERR\_EID** #define CFE\_TBL\_IN\_REGISTRY\_ERR\_EID 82  
TBL Delete Table CDS Command For Registered Table Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table being currently registered.  
Definition at line 592 of file cfe\_tbl\_eventids.h.

**12.130.2.24 CFE\_TBL\_INIT\_INF\_EID** #define CFE\_TBL\_INIT\_INF\_EID 1  
TB Initialization Event ID.

Type: INFORMATION

Cause:

Table Services Task initialization complete.  
Definition at line 42 of file cfe\_tbl\_eventids.h.

**12.130.2.25 CFE\_TBL\_LEN\_ERR\_EID** #define CFE\_TBL\_LEN\_ERR\_EID 52  
TBL Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the message ID and command code received on the TBL message pipe.  
Definition at line 257 of file cfe\_tbl\_eventids.h.

**12.130.2.26 CFE\_TBL\_LOAD\_ABORT\_ERR\_EID** #define CFE\_TBL\_LOAD\_ABORT\_ERR\_EID 69  
TBL Abort Table Load No Load Started Event ID.

Type: ERROR

Cause:

[TBL Abort Table Load Command](#) failure due to no load in progress.  
Definition at line 438 of file cfe\_tbl\_eventids.h.

**12.130.2.27 CFE\_TBL\_LOAD\_ABORT\_INF\_EID** #define CFE\_TBL\_LOAD\_ABORT\_INF\_EID 21  
TBL Abort Table Load Success Event ID.

Type: INFORMATION

Cause:

[TBL Abort Table Load Command](#) success.

Definition at line 157 of file cfe\_tbl\_eventids.h.

**12.130.2.28 CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID** #define CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID 72  
TBL Load Table File Exceeds Table Size Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified offset and/or size of data exceeding the table size. OVERLOADED  
Definition at line 474 of file cfe\_tbl\_eventids.h.

**12.130.2.29 CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID** #define CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID 101  
TBL Load Table Filename Too Long Event ID.

Type: ERROR

Cause:

Load table filename too long.

Definition at line 795 of file cfe\_tbl\_eventids.h.

**12.130.2.30 CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID** #define CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID 100  
TBL Load Table API Load Already In Progress Event ID.

Type: ERROR

Cause:

[CFE\\_TBL\\_Load](#) API failure due to load already in progress.

Definition at line 784 of file cfe\_tbl\_eventids.h.

**12.130.2.31 CFE\_TBL\_LOAD\_PEND\_REQ\_INF\_EID** #define CFE\_TBL\_LOAD\_PEND\_REQ\_INF\_EID 17  
TBL Load Table Pending Notification Success Event ID.

Type: DEBUG

Cause:

TBL load table pending notification successfully sent.  
Definition at line 134 of file cfe\_tbl\_eventids.h.

**12.130.2.32 CFE\_TBL\_LOAD\_SUCCESS\_INF\_EID** #define CFE\_TBL\_LOAD\_SUCCESS\_INF\_EID 35  
TBL Load Table API Success Event ID.

Type: DEBUG (the first time) and INFORMATION (normally)

Cause:

[CFE\\_TBL\\_Load](#) API success for dump only or normal table. OVERLOADED  
Definition at line 191 of file cfe\_tbl\_eventids.h.

**12.130.2.33 CFE\_TBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID** #define CFE\_TBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID 102  
TBL Load Table Name Mismatch Event ID.

Type: ERROR

Cause:

Load table name in the table file header does not match the specified table name.  
Definition at line 806 of file cfe\_tbl\_eventids.h.

**12.130.2.34 CFE\_TBL\_LOAD\_TYPE\_ERR\_EID** #define CFE\_TBL\_LOAD\_TYPE\_ERR\_EID 94  
TBL Load Table API Invalid Source Type Event ID.

Type: ERROR

Cause:

[CFE\\_TBL\\_Load](#) API valid due to invalid source type.  
Definition at line 729 of file cfe\_tbl\_eventids.h.

**12.130.2.35 CFE\_TBL\_LOAD\_VAL\_ERR\_EID** #define CFE\_TBL\_LOAD\_VAL\_ERR\_EID 93  
TBL Validation Function Invalid Return Code Event ID.

Type: ERROR

Cause:

Invalid table validation function return code.  
Definition at line 718 of file cfe\_tbl\_eventids.h.

**12.130.2.36 CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID** #define CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID 79  
TBL Load Table For Dump Only Table Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to table being dump only. OVERLOADED  
Definition at line 555 of file cfe\_tbl\_eventids.h.

**12.130.2.37 CFE\_TBL\_LOADING\_PENDING\_ERR\_EID** #define CFE\_TBL\_LOADING\_PENDING\_ERR\_EID 88  
TBL Load Table Command Load Pending Event ID.

Type: ERROR

Cause:

[TBL Load Table Command](#) failed due to a load already pending.  
Definition at line 663 of file cfe\_tbl\_eventids.h.

**12.130.2.38 CFE\_TBL\_MID\_ERR\_EID** #define CFE\_TBL\_MID\_ERR\_EID 50  
TBL Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the TBL message pipe.  
Definition at line 235 of file cfe\_tbl\_eventids.h.

**12.130.2.39 CFE\_TBL\_NO\_INACTIVE\_BUFFER\_ERR\_EID** #define CFE\_TBL\_NO\_INACTIVE\_BUFFER\_ERR\_EID 66  
TBL Validate Or Write Table Command No Inactive Buffer Event ID.

Type: ERROR

Cause:

[TBL Validate Table Command](#) or [TBL Write Table Command](#) failure due to requesting non-existent inactive buffer. OVERLOADED  
Definition at line 403 of file cfe\_tbl\_eventids.h.

**12.130.2.40 CFE\_TBL\_NO SUCH\_TABLE\_ERR\_EID** #define CFE\_TBL\_NO SUCH\_TABLE\_ERR\_EID 57  
TBL Table Name Not Found Event ID.

Type: ERROR

Cause:

TBL command handler unable to find table name. OVERLOADED  
Definition at line 313 of file cfe\_tbl\_eventids.h.

**12.130.2.41 CFE\_TBL\_NO\_WORK\_BUFFERS\_ERR\_EID** #define CFE\_TBL\_NO\_WORK\_BUFFERS\_ERR\_EID 60  
TBL Load Or Dump Table No Working Buffers Available Event ID.

Type: ERROR

Cause:

TBL Load or Dump failure due to no working buffers available or internal error. OVERLOADED  
Definition at line 346 of file cfe\_tbl\_eventids.h.

**12.130.2.42 CFE\_TBL\_NOOP\_INF\_EID** #define CFE\_TBL\_NOOP\_INF\_EID 10  
TBL No-op Command Success Event ID.

Type: INFORMATION

Cause:

[NO-OP TBL No-op Command](#) success.  
Definition at line 53 of file cfe\_tbl\_eventids.h.

**12.130.2.43 CFE\_TBL\_NOT\_CRITICAL\_TBL\_ERR\_EID** #define CFE\_TBL\_NOT\_CRITICAL\_TBL\_ERR\_EID 83  
TBL Delete Table CDS Command Invalid CDS Type Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to CDS being in the table registry but not registered as a table within ES.  
Definition at line 604 of file cfe\_tbl\_eventids.h.

**12.130.2.44 CFE\_TBL\_NOT\_IN\_CRIT\_REG\_ERR\_EID** #define CFE\_TBL\_NOT\_IN\_CRIT\_REG\_ERR\_EID 84  
TBL Delete Table CDS Command Not In Critical Table Registry Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table not being in the critical table registry.  
Definition at line 616 of file cfe\_tbl\_eventids.h.

**12.130.2.45 CFE\_TBL\_OVERWRITE\_DUMP\_INF\_EID** #define CFE\_TBL\_OVERWRITE\_DUMP\_INF\_EID 13  
TBL Write Table To Existing File Success Event ID.

Type: INFORMATION

Cause:

TBL write table to an existing file success.  
Definition at line 87 of file cfe\_tbl\_eventids.h.

**12.130.2.46 CFE\_TBL\_OVERWRITE\_REG\_DUMP\_INF\_EID** #define CFE\_TBL\_OVERWRITE\_REG\_DUMP\_INF\_EID 15  
TBL Write Table Registry To Existing File Success Event ID.

Type: DEBUG

Cause:

TBL Write Table Registry to an existing file completed successfully.  
Definition at line 109 of file cfe\_tbl\_eventids.h.

**12.130.2.47 CFE\_TBL\_PARTIAL\_LOAD\_ERR\_EID** #define CFE\_TBL\_PARTIAL\_LOAD\_ERR\_EID 74  
TBL Load Table Uninitialized Partial Load Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to attempting a partial load to an uninitialized table. OVERLOADED  
Definition at line 496 of file cfe\_tbl\_eventids.h.

**12.130.2.48 CFE\_TBL\_PROCESSOR\_ID\_ERR\_EID** #define CFE\_TBL\_PROCESSOR\_ID\_ERR\_EID 98  
TBL Read Header Invalid Processor ID Event ID.

Type: ERROR

Cause:

Invalid processor ID in table file header.  
Definition at line 773 of file cfe\_tbl\_eventids.h.

**12.130.2.49 CFE\_TBL\_REGISTER\_ERR\_EID** #define CFE\_TBL\_REGISTER\_ERR\_EID 90  
TBL Register Table Failed Event ID.

Type: ERROR

Cause:

TBL table registration failure. See system log for more information.  
Definition at line 685 of file cfe\_tbl\_eventids.h.

**12.130.2.50 CFE\_TBL\_RESET\_INF\_EID** #define CFE\_TBL\_RESET\_INF\_EID 11  
TBL Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[TBL Reset Counters Command](#) success.  
Definition at line 64 of file cfe\_tbl\_eventids.h.

**12.130.2.51 CFE\_TBL\_SHARE\_ERR\_EID** #define CFE\_TBL\_SHARE\_ERR\_EID 91  
TBL Share Table Failed Event ID.

Type: ERROR

Cause:

TBL share table failure. See system log for more information.  
Definition at line 696 of file cfe\_tbl\_eventids.h.

**12.130.2.52 CFE\_TBL\_SPACECRAFT\_ID\_ERR\_EID** #define CFE\_TBL\_SPACECRAFT\_ID\_ERR\_EID 97  
TBL Read Header Invalid Spacecraft ID Event ID.

Type: ERROR

Cause:

Invalid spacecraft ID in table file header.  
Definition at line 762 of file cfe\_tbl\_eventids.h.

**12.130.2.53 CFE\_TBL\_TLM\_REG\_CMD\_INF\_EID** #define CFE\_TBL\_TLM\_REG\_CMD\_INF\_EID 18  
TBL Telemeter Table Registry Entry Command Success Event ID.

Type: DEBUG

Cause:

[TBL Telemeter Table Registry Entry command](#) successfully set the table registry index to telemeter in the next house-keeping packet.  
Definition at line 146 of file cfe\_tbl\_eventids.h.

**12.130.2.54 CFE\_TBL\_TOO\_MANY\_DUMPS\_ERR\_EID** #define CFE\_TBL\_TOO\_MANY\_DUMPS\_ERR\_EID 76  
TBL Write Table Command Dump Only Control Blocks Exceeded Event ID.

Type: ERROR

Cause:

[TBL Write Table Command](#) failure due to exceeding the allocated number of control blocks available to write a dump only table.  
Definition at line 520 of file cfe\_tbl\_eventids.h.

**12.130.2.55 CFE\_TBL\_TOO\_MANY\_VALIDATIONS\_ERR\_EID** #define CFE\_TBL\_TOO\_MANY\_VALIDATIONS\_ERR\_EID 67

TBL Validate Table Command Result Storage Exceeded Event ID.

Type: ERROR

Cause:

[TBL Validate Table Command](#) failure due to exceeding result storage.

Definition at line 415 of file cfe\_tbl\_eventids.h.

**12.130.2.56 CFE\_TBL\_UNREGISTER\_ERR\_EID** #define CFE\_TBL\_UNREGISTER\_ERR\_EID 92

TBL Unregister Table Failed Event ID.

Type: ERROR

Cause:

TBL unregister table failure. See system log for more information.

Definition at line 707 of file cfe\_tbl\_eventids.h.

**12.130.2.57 CFE\_TBL\_UNVALIDATED\_ERR\_EID** #define CFE\_TBL\_UNVALIDATED\_ERR\_EID 81

TBL Activate Table Command Inactive Image Not Validated Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to the inactive image not being validated.

Definition at line 580 of file cfe\_tbl\_eventids.h.

**12.130.2.58 CFE\_TBL\_UPDATE\_ERR\_EID** #define CFE\_TBL\_UPDATE\_ERR\_EID 95

TBL Update Table Failed Event ID.

Type: ERROR

Cause:

TBL update table failure due to an internal error. OVERLOADED

Definition at line 740 of file cfe\_tbl\_eventids.h.

**12.130.2.59 CFE\_TBL\_UPDATE\_SUCCESS\_INF\_EID** #define CFE\_TBL\_UPDATE\_SUCCESS\_INF\_EID 37  
TBL Update Table Success Event ID.

Type: INFORMATION

Cause:

Table update successfully completed.

Definition at line 213 of file cfe\_tbl\_eventids.h.

**12.130.2.60 CFE\_TBL\_VAL\_REQ\_MADE\_INF\_EID** #define CFE\_TBL\_VAL\_REQ\_MADE\_INF\_EID 16  
TBL Validate Table Request Success Event ID.

Type: DEBUG

Cause:

[TBL Validate Table Command](#) success. Note this event signifies the request to validate the table has been successfully submitted. Completion will generate a [CFE\\_TBL\\_VALIDATION\\_INF\\_EID](#) or [CFE\\_TBL\\_VALIDATION\\_ERR\\_EID](#) event messages.

Definition at line 123 of file cfe\_tbl\_eventids.h.

**12.130.2.61 CFE\_TBL\_VALIDATION\_ERR\_EID** #define CFE\_TBL\_VALIDATION\_ERR\_EID 96  
TBL Validate Table Validation Failed Event ID.

Type: ERROR

Cause:

TBL validate table function indicates validation failed. OVERLOADED

Definition at line 751 of file cfe\_tbl\_eventids.h.

**12.130.2.62 CFE\_TBL\_VALIDATION\_INF\_EID** #define CFE\_TBL\_VALIDATION\_INF\_EID 36  
TBL Validate Table Success Event ID.

Type: INFORMATION

Cause:

Table active or inactive image successfully validated by the registered validation function. OVERLOADED

Definition at line 202 of file cfe\_tbl\_eventids.h.

**12.130.2.63 CFE\_TBL\_WRITE\_CFE\_HDR\_ERR\_EID** #define CFE\_TBL\_WRITE\_CFE\_HDR\_ERR\_EID 63  
TBL Write Standard File Header Failure Event ID.

Type: ERROR

Cause:

TBL Write Table or Table Registry File failure writing the standard file header. OVERLOADED  
Definition at line 368 of file cfe\_tbl\_eventids.h.

**12.130.2.64 CFE\_TBL\_WRITE\_DUMP\_INF\_EID** #define CFE\_TBL\_WRITE\_DUMP\_INF\_EID 14  
TBL Write Table To New File Success Event ID.

Type: INFORMATION

Cause:

TBL write table to a new file success.  
Definition at line 98 of file cfe\_tbl\_eventids.h.

**12.130.2.65 CFE\_TBL\_WRITE\_REG\_DUMP\_INF\_EID** #define CFE\_TBL\_WRITE\_REG\_DUMP\_INF\_EID 22  
TBL Write Table Registry To New File Success Event ID.

Type: DEBUG

Cause:

TBL Write Table Registry to a new file completed successfully.  
Definition at line 168 of file cfe\_tbl\_eventids.h.

**12.130.2.66 CFE\_TBL\_WRITE\_TBL\_HDR\_ERR\_EID** #define CFE\_TBL\_WRITE\_TBL\_HDR\_ERR\_EID 64  
TBL Write Table File Header Failure Event ID.

Type: ERROR

Cause:

TBL Write Table failure writing the table image file header.  
Definition at line 379 of file cfe\_tbl\_eventids.h.

**12.130.2.67 CFE\_TBL\_WRITE\_TBL\_IMG\_ERR\_EID** #define CFE\_TBL\_WRITE\_TBL\_IMG\_ERR\_EID 65  
TBL Write Table File Data Failure Event ID.

Type: ERROR

Cause:

TBL Write Table failure writing the table data.  
Definition at line 390 of file cfe\_tbl\_eventids.h.

**12.130.2.68 CFE\_TBL\_WRITE\_TBL\_REG\_ERR\_EID** #define CFE\_TBL\_WRITE\_TBL\_REG\_ERR\_EID 68  
TBL Write Table Registry File Data Failure Event ID.

Type: ERROR

Cause:

TB Write Table Registry failure writing file data.  
Definition at line 426 of file cfe\_tbl\_eventids.h.

**12.130.2.69 CFE\_TBL\_ZERO\_LENGTH\_LOAD\_ERR\_EID** #define CFE\_TBL\_ZERO\_LENGTH\_LOAD\_ERR\_EID 73  
TBL Load Table File Zero Length Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified size of data being zero.  
Definition at line 485 of file cfe\_tbl\_eventids.h.

## 12.131 cfe/modules/time/config/default\_cfe\_time\_extern\_typedefs.h File Reference

```
#include "common_types.h"
```

### Data Structures

- struct [CFE\\_TIME\\_SysTime](#)

*Data structure used to hold system time values.*

## Typedefs

- `typedef struct CFE_TIME_SysTime CFE_TIME_SysTime_t`  
*Data structure used to hold system time values.*
- `typedef uint8 CFE_TIME_FlagBit_Enum_t`  
*Bit positions of the various clock state flags.*
- `typedef int16 CFE_TIME_ClockState_Enum_t`  
*Enumerated types identifying the quality of the current time.*
- `typedef uint8 CFE_TIME_SourceSelect_Enum_t`  
*Clock Source Selection Parameters.*
- `typedef uint8 CFE_TIME_ToneSignalSelect_Enum_t`  
*Tone Signal Selection Parameters.*
- `typedef uint8 CFE_TIME_AdjustDirection_Enum_t`  
*STCF adjustment direction (for both one-time and 1Hz adjustments)*
- `typedef uint8 CFE_TIME_FlywheelState_Enum_t`  
*Fly-wheel status values.*
- `typedef uint8 CFE_TIME_SetState_Enum_t`  
*Clock status values (has the clock been set to correct time)*

## Enumerations

- `enum CFE_TIME_FlagBit {  
 CFE_TIME_FlagBit_CLKSET = 0, CFE_TIME_FlagBit_FLYING = 1, CFE_TIME_FlagBit_SRCINT = 2,  
 CFE_TIME_FlagBit_SIGPRI = 3,  
 CFE_TIME_FlagBit_SRVFLY = 4, CFE_TIME_FlagBit_CMDFLY = 5, CFE_TIME_FlagBit_ADDADJ = 6,  
 CFE_TIME_FlagBit_ADD1HZ = 7,  
 CFE_TIME_FlagBit_ADDTCL = 8, CFE_TIME_FlagBit_SERVER = 9, CFE_TIME_FlagBit_GDTONE = 10 }`  
*Label definitions associated with CFE\_TIME\_FlagBit\_Enum\_t.*
- `enum CFE_TIME_ClockState { CFE_TIME_ClockState_INVALID = -1, CFE_TIME_ClockState_VALID = 0,  
 CFE_TIME_ClockState_FLYWHEEL = 1 }`  
*Label definitions associated with CFE\_TIME\_ClockState\_Enum\_t.*
- `enum CFE_TIME_SourceSelect { CFE_TIME_SourceSelect_INTERNAL = 1, CFE_TIME_SourceSelect_EXTERNAL  
 = 2 }`  
*Label definitions associated with CFE\_TIME\_SourceSelect\_Enum\_t.*
- `enum CFE_TIME_ToneSignalSelect { CFE_TIME_ToneSignalSelect_PRIMARY = 1, CFE_TIME_ToneSignalSelect_REDUNDANT  
 = 2 }`  
*Label definitions associated with CFE\_TIME\_ToneSignalSelect\_Enum\_t.*
- `enum CFE_TIME_AdjustDirection { CFE_TIME_AdjustDirection_ADD = 1, CFE_TIME_AdjustDirection_SUBTRACT  
 = 2 }`  
*Label definitions associated with CFE\_TIME\_AdjustDirection\_Enum\_t.*
- `enum CFE_TIME_FlywheelState { CFE_TIME_FlywheelState_NO_FLY = 0, CFE_TIME_FlywheelState_IS_FLY  
 = 1 }`  
*Label definitions associated with CFE\_TIME\_FlywheelState\_Enum\_t.*
- `enum CFE_TIME_SetState { CFE_TIME_SetState_NOT_SET = 0, CFE_TIME_SetState_WAS_SET = 1 }`  
*Label definitions associated with CFE\_TIME\_SetState\_Enum\_t.*

### 12.131.1 Detailed Description

Declarations and prototypes for cfe\_time\_extern\_typedefs module

## 12.131.2 Typedef Documentation

**12.131.2.1 CFE\_TIME\_AdjustDirection\_Enum\_t** `typedef uint8 CFE_TIME_AdjustDirection_Enum_t`  
STCF adjustment direction (for both one-time and 1Hz adjustments)

See also

enum [CFE\\_TIME\\_AdjustDirection](#)

Definition at line 234 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.2.2 CFE\_TIME\_ClockState\_Enum\_t** `typedef int16 CFE_TIME_ClockState_Enum_t`  
Enumerated types identifying the quality of the current time.

Description

The [CFE\\_TIME\\_ClockState\\_Enum\\_t](#) enumerations identify the three recognized states of the current time. If the clock has never been successfully synchronized with the primary onboard clock source, the time is considered to be [CFE\\_TIME\\_ClockState\\_INVALID](#). If the time is currently synchronized (i.e. - the primary synchronization mechanism has not been dropped for any significant amount of time), then the current time is considered to be [CFE\\_TIME\\_ClockState\\_VALID](#). If the time had, at some point in the past, been synchronized, but the synchronization with the primary onboard clock has since been lost, then the time is considered to be [CFE\\_TIME\\_ClockState\\_FLYWHEEL](#). Since different clocks drift at different rates from one another, the accuracy of the time while in [CFE\\_TIME\\_ClockState\\_FLYWHEEL](#) is dependent upon the time spent in that state.

See also

enum [CFE\\_TIME\\_ClockState](#)

Definition at line 165 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.2.3 CFE\_TIME\_FlagBit\_Enum\_t** `typedef uint8 CFE_TIME_FlagBit_Enum_t`  
Bit positions of the various clock state flags.

See also

enum [CFE\\_TIME\\_FlagBit](#)

Definition at line 113 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.2.4 CFE\_TIME\_FlywheelState\_Enum\_t** `typedef uint8 CFE_TIME_FlywheelState_Enum_t`  
Fly-wheel status values.

See also

enum [CFE\\_TIME\\_FlywheelState](#)

Definition at line 257 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.2.5 CFE\_TIME\_SetState\_Enum\_t** `typedef uint8 CFE_TIME_SetState_Enum_t`  
Clock status values (has the clock been set to correct time)

See also

enum [CFE\\_TIME\\_SetState](#)

Definition at line 280 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.2.6 CFE\_TIME\_SourceSelect\_Enum\_t** `typedef uint8 CFE_TIME_SourceSelect_Enum_t`  
Clock Source Selection Parameters.

See also

enum [CFE\\_TIME\\_SourceSelect](#)

Definition at line 188 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.2.7 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.

**12.131.2.8 CFE\_TIME\_ToneSignalSelect\_Enum\_t** `typedef uint8 CFE_TIME_ToneSignalSelect_Enum_t`  
Tone Signal Selection Parameters.

See also

enum [CFE\\_TIME\\_ToneSignalSelect](#)

Definition at line 211 of file default\_cfe\_time\_extern\_typedefs.h.

### 12.131.3 Enumeration Type Documentation

**12.131.3.1 CFE\_TIME\_AdjustDirection** `enum CFE_TIME_AdjustDirection`  
Label definitions associated with `CFE_TIME_AdjustDirection_Enum_t`.

Enumerator

<code>CFE_TIME_AdjustDirection_ADD</code>	Add time adjustment.
<code>CFE_TIME_AdjustDirection_SUBTRACT</code>	Subtract time adjustment.

Definition at line 216 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.3.2 CFE\_TIME\_ClockState** `enum CFE_TIME_ClockState`  
Label definitions associated with `CFE_TIME_ClockState_Enum_t`.

## Enumerator

CFE_TIME_ClockState_INVALID	The spacecraft time has not been set since the last clock reset. Times returned by clock routines have no relationship to any ground-based time reference.
CFE_TIME_ClockState_VALID	The spacecraft time has been set at least once since the last clock reset, and it is synchronized with the primary on-board time base. Times returned by clock routines can be trusted.
CFE_TIME_ClockState_FLYWHEEL	The spacecraft time has been set at least once since the last clock reset, but it is not currently synchronized with the primary on-board time base. Times returned by clock routines are a "best guess" based on a non-optimal oscillator.

Definition at line 118 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.3.3 CFE\_TIME\_FlagBit enum CFE\_TIME\_FlagBit**

Label definitions associated with CFE\_TIME\_FlagBit\_Enum\_t.

## Enumerator

CFE_TIME_FlagBit_CLKSET	The spacecraft time has been set.
CFE_TIME_FlagBit_FLYING	This instance of Time Services is flywheeling.
CFE_TIME_FlagBit_SRCINT	The clock source is set to internal.
CFE_TIME_FlagBit_SIGPRI	The clock signal is set to primary.
CFE_TIME_FlagBit_SRVFLY	The Time Server is in flywheel mode.
CFE_TIME_FlagBit_CMDFLY	This instance of Time Services was commanded into flywheel mode.
CFE_TIME_FlagBit_ADDADJ	One time STCF Adjustment is to be done in positive direction.
CFE_TIME_FlagBit_ADD1HZ	1 Hz STCF Adjustment is to be done in a positive direction
CFE_TIME_FlagBit_ADDTCL	Time Client Latency is applied in a positive direction.
CFE_TIME_FlagBit_SERVER	This instance of Time Services is a Time Server.
CFE_TIME_FlagBit_GDTONE	The tone received is good compared to the last tone received.

Definition at line 50 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.3.4 CFE\_TIME\_FlywheelState enum CFE\_TIME\_FlywheelState**

Label definitions associated with CFE\_TIME\_FlywheelState\_Enum\_t.

## Enumerator

CFE_TIME_FlywheelState_NO_FLY	Not in flywheel state.
CFE_TIME_FlywheelState_IS_FLY	In flywheel state.

Definition at line 239 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.3.5 CFE\_TIME\_SetState enum CFE\_TIME\_SetState**

Label definitions associated with CFE\_TIME\_SetState\_Enum\_t.

**Enumerator**

CFE_TIME_SetState_NOT_SET	Spacecraft time has not been set.
CFE_TIME_SetState_WAS_SET	Spacecraft time has been set.

Definition at line 262 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.3.6 CFE\_TIME\_SourceSelect enum CFE\_TIME\_SourceSelect**

Label definitions associated with CFE\_TIME\_SourceSelect\_Enum\_t.

**Enumerator**

CFE_TIME_SourceSelect_INTERNAL	Use Internal Source.
CFE_TIME_SourceSelect_EXTERNAL	Use External Source.

Definition at line 170 of file default\_cfe\_time\_extern\_typedefs.h.

**12.131.3.7 CFE\_TIME\_ToneSignalSelect enum CFE\_TIME\_ToneSignalSelect**

Label definitions associated with CFE\_TIME\_ToneSignalSelect\_Enum\_t.

**Enumerator**

CFE_TIME_ToneSignalSelect_PRIMARY	Primary Source.
CFE_TIME_ToneSignalSelect_REDUNDANT	Redundant Source.

Definition at line 193 of file default\_cfe\_time\_extern\_typedefs.h.

**12.132 cfe/modules/time/config/default\_cfe\_time\_fcncodes.h File Reference****Macros****Time Services Command Codes**

- #define CFE\_TIME\_NOOP\_CC 0 /\* no-op command \*/
- #define CFE\_TIME\_RESET\_COUNTERS\_CC 1 /\* reset counters \*/
- #define CFE\_TIME\_SEND\_DIAGNOSTIC\_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) \*/

### 12.132.1 Detailed Description

Specification for the CFE Time Services (CFE\_TIME) command function codes

#### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

### 12.132.2 Macro Definition Documentation

**12.132.2.1 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 601 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.2 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\_AddSTCFAadj

### Command Structure

[CFE\\_TIME\\_AddAdjustCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_TIME\_STCFSecs** - Housekeeping Telemetry point indicating new STCF seconds value
- **\$sc\_\$cpu\_TIME\_STCFSubsecs** - Housekeeping Telemetry point indicating new STCF subseconds value
- The [CFE\\_TIME\\_DELTA\\_EID](#) informational event message will be generated

### Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDEC** - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_DELTA\\_ERR\\_EID](#) or [CFE\\_TIME\\_DELTA\\_CFG\\_EID](#))

### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

### See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#), [CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#),  
[CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#)

Definition at line 521 of file default\_cfe\_time\_fcncodes.h.

### 12.132.2.3 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\\_DLlatentS](#), [\\$sc\\_\\$cpu\\_TIME\\_DLlatentSs](#) - Housekeeping Telemetry point indicating command specified values
- [\\$sc\\_\\$cpu\\_TIME\\_DLlatentDir](#) - Diagnostic Telemetry point indicating commanded latency direction
- The [CFE\\_TIME\\_DELAY\\_EID](#) informational event message will be generated

#### Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Client

Evidence of Failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_DELAY\\_CFG\\_EID](#) or [CFE\\_TIME\\_DELAY\\_ERR\\_EID](#))

#### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

#### See also

[CFE\\_TIME\\_SUB\\_DELAY\\_CC](#)

Definition at line 290 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.4 CFE\_TIME\_NOOP\_CC** #define CFE\_TIME\_NOOP\_CC 0 /\* no-op command \*/**Name** Time No-Op**Description**

This command performs no other function than to increment the command execution counter. The command may be used to verify general aliveness of the Time Services task.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_NOOP**Command Structure**

[CFE\\_TIME\\_NoopCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- The [CFE\\_TIME\\_NOOP\\_EID](#) informational event message will be generated

**Error Conditions**

There are no error conditions for this command. If the Time Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

**Criticality**

None

**See also**

Definition at line 66 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.5 CFE\_TIME\_RESET\_COUNTERS\_CC** #define CFE\_TIME\_RESET\_COUNTERS\_CC 1 /\* reset counters \*/**Name** Time Reset Counters**Description**

This command resets the following counters within the Time Services [Housekeeping Telemetry](#) :

- Command Execution Counter (\$sc\_\$cpu\_TIME\_CMDPC)
- Command Error Counter (\$sc\_\$cpu\_TIME\_CMDEC) This command also resets the following counters within the Time Services [Diagnostic Telemetry](#) :
  - Tone Signal Detected Software Bus Message Counter (\$sc\_\$cpu\_TIME\_DTSDETCNT)
  - Time at the Tone Data Software Bus Message Counter (\$sc\_\$cpu\_TIME\_DTATTCNT)
  - Tone Signal/Data Verify Counter (\$sc\_\$cpu\_TIME\_DVERIFYCNT)
  - Tone Signal/Data Error Counter (\$sc\_\$cpu\_TIME\_DVERIFYER)

- Tone Signal Interrupt Counter (\$sc\_\$cpu\_TIME\_DTsISRCNT)
- Tone Signal Interrupt Error Counter (\$sc\_\$cpu\_TIME\_DTsISRERR)
- Tone Signal Task Counter (\$sc\_\$cpu\_TIME\_DTsTaskCNT)
- Local 1 Hz Interrupt Counter (\$sc\_\$cpu\_TIME\_D1HzISRCNT)
- Local 1 Hz Task Counter (\$sc\_\$cpu\_TIME\_D1HzTaskCNT)
- Reference Time Version Counter (\$sc\_\$cpu\_TIME\_DVersionCNT)

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_ResetCtrs

#### Command Structure

[CFE\\_TIME\\_ResetCountersCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will reset to 0
- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will reset to 0
- The [CFE\\_TIME\\_RESET\\_EID](#) informational event message will be generated

#### Error Conditions

There are no error conditions for this command. If the Time Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is reset unconditionally.

#### Criticality

None

#### See also

Definition at line 111 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.6 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 145 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.7 CFE\_TIME\_SET\_LEAP\_SECONDS\_CC** #define CFE\_TIME\_SET\_LEAP\_SECONDS\_CC 10 /\* set Leap Seconds \*/

**Name** Set Leap Seconds

### Description

This command sets the spacecraft Leap Seconds to the specified value. Leap Seconds may be positive or negative, and there is no limit to the value except, of course, the limit imposed by the 16 bit signed integer data type. The new Leap Seconds value takes effect immediately upon execution of this command.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SetClockLeap

### Command Structure

[CFE\\_TIME\\_SetLeapSecondsCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_TIME\_LeapSecs** - Housekeeping Telemetry point indicating new Leap seconds value
- The [CFE\\_TIME\\_LEAPS\\_EID](#) informational event message will be generated

### Error Conditions

- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDEC** - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_LEAPS\\_CFG\\_EID](#))

### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

### See also

[CFE\\_TIME\\_SET\\_TIME\\_CC](#), [CFE\\_TIME\\_SET\\_MET\\_CC](#), [CFE\\_TIME\\_SET\\_STCF\\_CC](#)

Definition at line 485 of file default\_cfe\_time\_fcncodes.h.

#### 12.132.2.8 CFE\_TIME\_SET\_MET\_CC `#define CFE_TIME_SET_MET_CC 8 /* set MET */`

**Name** Set Mission Elapsed Time

### Description

This command sets the Mission Elapsed Timer (MET) to the specified value.

Note that the MET (as implemented for cFE Time Service) is a logical representation and not a physical timer. Thus, setting the MET is not dependent on whether the hardware supports a MET register that can be written to.

Note also that Time Service "assumes" that during normal operation, the MET is synchronized to the tone signal. Therefore, unless operating in FLYWHEEL mode, the sub-seconds portion of the MET will be set to zero at the next tone signal interrupt.

The new MET takes effect immediately upon execution of this command.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SetClockMET

### Command Structure

[CFE\\_TIME\\_SetMETCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_TIME\_METSecs** - Housekeeping Telemetry point indicating new MET seconds value
- **\$sc\_\$cpu\_TIME\_METSubsecs** - Housekeeping Telemetry point indicating new MET subseconds value
- The [CFE\\_TIME\\_MET\\_EID](#) informational event message will be generated

### Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDEC** - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_MET\\_CFG\\_EID](#) or [CFE\\_TIME\\_MET\\_ERR\\_EID](#))

### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

### See also

[CFE\\_TIME\\_SET\\_TIME\\_CC](#), [CFE\\_TIME\\_SET\\_STCF\\_CC](#), [CFE\\_TIME\\_SET\\_LEAP\\_SECONDS\\_CC](#)

Definition at line 413 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.9 CFE\_TIME\_SET\_SIGNAL\_CC** #define CFE\_TIME\_SET\_SIGNAL\_CC 15 /\* set clock signal (pri vs red) \*/

**Name** Set Tone Signal Source

### Description

This command selects the Time Service tone signal source. Although the list of potential tone signal sources is mission specific, a common choice is the selection of primary or redundant tone signal. The selection may be available to both the Time Server and Time Clients, depending on hardware configuration.

Notes:

- This command is only valid when the [CFE\\_PLATFORM\\_TIME\\_CFG\\_SIGNAL](#) configuration parameter in the cfe\_platform\_cfg.h file has been set to true.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SetSignal

### Command Structure

[CFE\\_TIME\\_SetSignalCmd\\_t](#)

### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_DSignal](#) - Diagnostic Telemetry point will indicate the command specified value
- The [CFE\\_TIME\\_SIGNAL\\_EID](#) informational event message will be generated

### Error Conditions

- Invalid Signal selection (a value other than [CFE\\_TIME\\_ToneSignalSelect\\_PRIMARY](#) or [CFE\\_TIME\\_ToneSignalSelect\\_REDUNDANT](#) was specified)
- Multiple Tone Signal Sources not available on this platform

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - Command Error counter will increment
- Error specific event message (either [CFE\\_TIME\\_SIGNAL\\_CFG\\_EID](#) or [CFE\\_TIME\\_SIGNAL\\_ERR\\_EID](#))

### Criticality

Although tone signal source selection is important, this command is not critical

### See also

[CFE\\_TIME\\_SET\\_STATE\\_CC](#), [CFE\\_TIME\\_SET\\_SOURCE\\_CC](#)

Definition at line 691 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.10 CFE\_TIME\_SET\_SOURCE\_CC** #define CFE\_TIME\_SET\_SOURCE\_CC 3 /\* set clock source (int vs ext) \*/

**Name** Set Time Source

#### Description

This command selects the Time Service clock source. Although the list of potential clock sources is mission specific and defined via configuration parameters, this command provides a common method for switching between the local processor clock and an external source for time data.

When commanded to accept external time data (GPS, MET, spacecraft time, etc.), the Time Server will enable input via an API function specific to the configuration definitions for the particular source. When commanded to use internal time data, the Time Server will ignore the external data. However, the Time Server will continue to use the API function as the trigger to generate a "time at the tone" command packet regardless of the internal/external command selection.

Notes:

- Operating in FLYWHEEL mode is not considered a choice related to clock source, but rather an element of the clock state. See below for a description of the [CFE\\_TIME\\_SET\\_STATE\\_CC](#) command.
- This command is only valid when the [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) configuration parameter in the `cfe_platform_cfg.h` file has been set to true.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SetSource

#### Command Structure

[CFE\\_TIME\\_SetSourceCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_DSource](#) - Diagnostic Telemetry point will indicate the command specified value
- The [CFE\\_TIME\\_SOURCE\\_EID](#) informational event message will be generated

#### Error Conditions

- Invalid Source selection (a value other than [CFE\\_TIME\\_SourceSelect\\_INTERNAL](#) or [CFE\\_TIME\\_SourceSelect\\_EXTERNAL](#) was specified)
- Time source selection not allowed on this platform

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - Command Error counter will increment
- Error specific event message (either [CFE\\_TIME\\_SOURCE\\_CFG\\_EID](#) or [CFE\\_TIME\\_SOURCE\\_ERR\\_EID](#))

#### Criticality

Although clock source selection is important, this command is not critical.

#### See also

[CFE\\_TIME\\_SET\\_STATE\\_CC](#), [CFE\\_TIME\\_SET\\_SIGNAL\\_CC](#)

Definition at line 195 of file `default_cfe_time_fcncodes.h`.

**12.132.2.11 CFE\_TIME\_SET\_STATE\_CC** /\* set clock state \*/

**Name** Set Time State

**Description**

This command indirectly affects the Time Service on-board determination of clock state. Clock state is a combination of factors, most significantly whether the spacecraft time has been accurately set, and whether Time Service is operating in FLYWHEEL mode.

This command may be used to notify the Time Server that spacecraft time is now correct, or that time is no longer correct. This information will be distributed to Time Clients, and in turn, to any interested sub-systems.

Also, this command may be used to force a Time Server or Time Client into FLYWHEEL mode. Use of FLYWHEEL mode is mainly for debug purposes although in extreme circumstances, it may be of value to force Time Service not to rely on normal time updates. Note that when commanded into FLYWHEEL mode, the Time Service will remain so until receipt of another "set state" command setting the state into a mode other than FLYWHEEL.

Note also that setting the clock state to VALID or INVALID on a Time Client that is currently getting time updates from the Time Server will have very limited effect. As soon as the Time Client receives the next time update, the VALID/INVALID selection will be set to that of the Time Server. However, setting a Time Client to FLYWHEEL cannot be overridden by the Time Server since the Time Client will ignore time updates from the Time Server while in FLYWHEEL mode.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SetState

**Command Structure**

[CFE\\_TIME\\_SetStateCmd\\_t](#)

**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_StateFlg](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagSet](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagFly](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagSrc](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagPri](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagSfly](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagCfly](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagAdjd](#), [\\$sc\\_\\$cpu\\_TIME\\_Flag1Hzd](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagClat](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagSorC](#), [\\$sc\\_\\$cpu\\_TIME\\_FlagNIU](#) - Housekeeping Telemetry point "may" indicate the command specified value (see above)
- The [CFE\\_TIME\\_STATE\\_EID](#) informational event message will be generated

**Error Conditions**

- Invalid State selection (a value other than [CFE\\_TIME\\_ClockState\\_INVALID](#), [CFE\\_TIME\\_ClockState\\_VALID](#) or [CFE\\_TIME\\_ClockState\\_FLYWHEEL](#) was specified)
- Time source selection not allowed on this platform

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - Command Error counter will increment
- Error specific event message ([CFE\\_TIME\\_STATE\\_ERR\\_EID](#))

### Criticality

Setting Time Service into FLYWHEEL mode is not particularly hazardous, as the result may be that the calculation of spacecraft time is done using a less than optimal timer. However, inappropriately setting the clock state to V $\leftarrow$  ALID (indicating that spacecraft time is accurate) may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

### See also

[CFE\\_TIME\\_SET\\_SOURCE\\_CC](#), [CFE\\_TIME\\_SET\\_SIGNAL\\_CC](#)

Definition at line 252 of file default\_cfe\_time\_fcncodes.h.

### 12.132.2.12 CFE\_TIME\_SET\_STCF\_CC #define CFE\_TIME\_SET\_STCF\_CC 9 /\* set STCF \*/

**Name** Set Spacecraft Time Correlation Factor

#### Description

This command sets the Spacecraft Time Correlation Factor (STCF) to the specified value. This command differs from the previously described SET CLOCK in the nature of the command argument. This command sets the STCF value directly, rather than extracting the STCF from a value representing the total of MET, STCF and optionally, Leap Seconds. The new STCF takes effect immediately upon execution of this command.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SetClockSTCF

#### Command Structure

[CFE\\_TIME\\_SetSTCFCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_STCFSecs](#) - Housekeeping Telemetry point indicating new STCF seconds value
- [\\$sc\\_\\$cpu\\_TIME\\_STCFSubsecs](#) - Housekeeping Telemetry point indicating new STCF subseconds value
- The [CFE\\_TIME\\_STCF\\_EID](#) informational event message will be generated

#### Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_STCF\\_CFG\\_EID](#) or [CFE\\_TIME\\_STCF\\_ERR\\_EID](#))

### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

### See also

[CFE\\_TIME\\_SET\\_TIME\\_CC](#), [CFE\\_TIME\\_SET\\_MET\\_CC](#), [CFE\\_TIME\\_SET\\_LEAP\\_SECONDS\\_CC](#)

Definition at line 450 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.13 CFE\_TIME\_SET\_TIME\_CC** #define CFE\_TIME\_SET\_TIME\_CC 7 /\* set time \*/**Name** Set Spacecraft Time**Description**

This command sets the spacecraft clock to a new value, regardless of the current setting (time jam). The new time value represents the desired offset from the mission-defined time epoch and takes effect immediately upon execution of this command. Time Service will calculate a new STCF value based on the current MET and the desired new time using one of the following:

If Time Service is configured to compute current time as TAI

- **STCF = (new time) - (current MET)**
- **(current time) = (current MET) + STCF**

If Time Service is configured to compute current time as UTC

- **STCF = ((new time) - (current MET)) + (Leap Seconds)**
- **(current time) = ((current MET) + STCF) - (Leap Seconds)**

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SetClock**Command Structure**[CFE\\_TIME\\_SetTimeCmd\\_t](#)**Command Verification**

Successful execution of this command may be verified with the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_TIME\_STCFSecs** - Housekeeping Telemetry point indicating newly calculated STCF seconds value
- **\$sc\_\$cpu\_TIME\_STCFSubsecs** - Housekeeping Telemetry point indicating newly calculated STCF subseconds value
- The [CFE\\_TIME\\_TIME\\_EID](#) informational event message will be generated

**Error Conditions**

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDEC** - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_TIME\\_CFG\\_EID](#) or [CFE\\_TIME\\_TIME\\_ERR\\_EID](#))

**Criticality**

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

**See also**[CFE\\_TIME\\_SET\\_MET\\_CC](#), [CFE\\_TIME\\_SET\\_STCF\\_CC](#), [CFE\\_TIME\\_SET\\_LEAP\\_SECONDS\\_CC](#)

Definition at line 373 of file default\_cfe\_time\_fcncodes.h.

```
12.132.2.14 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 649 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.15 CFE\_TIME\_SUB\_ADJUST\_CC** #define CFE\_TIME\_SUB\_ADJUST\_CC 12 /\* subtract one time S↔ TCF adjustment \*/

**Name** Subtract Delta from Spacecraft Time Correlation Factor

#### Description

This command adjusts the Spacecraft Time Correlation Factor (STCF) by subtracting the specified value. The new STCF takes effect immediately upon execution of this command.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_SubSTCFAdj

#### Command Structure

[CFE\\_TIME\\_SubAdjustCmd\\_t](#)

#### Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_STCFSecs](#) - Housekeeping Telemetry point indicating new STCF seconds value
- [\\$sc\\_\\$cpu\\_TIME\\_STCFSubsecs](#) - Housekeeping Telemetry point indicating new STCF subseconds value
- The [CFE\\_TIME\\_DELTA\\_EID](#) informational event message will be generated

#### Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_DELTA\\_ERR\\_EID](#) or [CFE\\_TIME\\_DELTA\\_CFG\\_EID](#))

#### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

#### See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_ADD\\_1HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SUB\\_1HZ\\_ADJUSTMENT\\_CC](#)

Definition at line 555 of file default\_cfe\_time\_fcncodes.h.

**12.132.2.16 CFE\_TIME\_SUB\_DELAY\_CC** #define CFE\_TIME\_SUB\_DELAY\_CC 6 /\* sub tone delay value \*/

**Name** Subtract Time from Tone Time Delay

## Description

This command is used to factor out a known, predictable latency between the Time Server and a particular Time Client. The correction is applied (subtracted) to the current time calculation for Time Clients, so this command has no meaning for Time Servers. Each Time Client can have a unique latency setting. The latency value is a positive number of seconds and microseconds that represent the deviation from the time maintained by the Time Server.

Note that it is unimaginable that the seconds value will ever be anything but zero.

## Command Mnemonic(s)

\$sc\_\$cpu\_TIME\_SubClockLat

## Command Structure

[CFE\\_TIME\\_SubDelayCmd\\_t](#)

## Command Verification

Successful execution of this command may be verified with the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_DLatentS](#), [\\$sc\\_\\$cpu\\_TIME\\_DLlatentSs](#) - Housekeeping Telemetry point indicating command specified values
- [\\$sc\\_\\$cpu\\_TIME\\_DLlatentDir](#) - Diagnostic Telemetry point indicating commanded latency direction
- The [CFE\\_TIME\\_DELAY\\_EID](#) informational event message will be generated

## Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Client

Evidence of Failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_DELAY\\_CFG\\_EID](#) or [CFE\\_TIME\\_DELAY\\_ERR\\_EID](#))

## Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

## See also

[CFE\\_TIME\\_ADD\\_DELAY\\_CC](#)

Definition at line 328 of file default\_cfe\_time\_fcncodes.h.

## 12.133 cfe/modules/time/config/default\_cfe\_time\_interface\_cfg.h File Reference

### Macros

- #define [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_TAI](#) true
- #define [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_UTC](#) false
- #define [CFE\\_MISSION\\_TIME\\_CFG\\_FAKE\\_TONE](#) true
- #define [CFE\\_MISSION\\_TIME\\_AT\\_TONE\\_WAS](#) true
- #define [CFE\\_MISSION\\_TIME\\_AT\\_TONE\\_WILL\\_BE](#) false

- #define CFE\_MISSION\_TIME\_MIN\_ELAPSED 0
- #define CFE\_MISSION\_TIME\_MAX\_ELAPSED 200000
- #define CFE\_MISSION\_TIME\_DEF\_MET\_SECS 1000
- #define CFE\_MISSION\_TIME\_DEF\_MET\_SUBS 0
- #define CFE\_MISSION\_TIME\_DEF\_STCF\_SECS 1000000
- #define CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS 0
- #define CFE\_MISSION\_TIME\_DEF\_LEAPS 37
- #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS 0
- #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS 1000
- #define CFE\_MISSION\_TIME\_EPOCH\_YEAR 1980
- #define CFE\_MISSION\_TIME\_EPOCH\_DAY 1
- #define CFE\_MISSION\_TIME\_EPOCH\_HOUR 0
- #define CFE\_MISSION\_TIME\_EPOCH\_MINUTE 0
- #define CFE\_MISSION\_TIME\_EPOCH\_SECOND 0
- #define CFE\_MISSION\_TIME\_EPOCH\_MICROS 0
- #define CFE\_MISSION\_TIME\_FS\_FACTOR 789004800

### 12.133.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.133.2 Macro Definition Documentation

#### 12.133.2.1 CFE\_MISSION\_TIME\_AT\_TONE\_WAS `#define CFE_MISSION_TIME_AT_TONE_WAS true`

**Purpose** Default Time and Tone Order

**Description:**

Time Services may be configured to expect the time at the tone data packet to either precede or follow the tone signal. If the time at the tone data packet follows the tone signal, then the data within the packet describes what the time "was" at the tone. If the time at the tone data packet precedes the tone signal, then the data within the packet describes what the time "will be" at the tone. One, and only one, of the following symbols must be set to true:

- CFE\_MISSION\_TIME\_AT\_TONE\_WAS
- CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE Note: If Time Services is defined as using a simulated tone signal (see [CFE\\_MISSION\\_TIME\\_CFG\\_FAKE\\_TONE](#) above), then the tone data packet must follow the tone signal.

#### Limits

Either CFE\_MISSION\_TIME\_AT\_TONE\_WAS or CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE must be set to true. They may not both be true and they may not both be false.

Definition at line 88 of file `default_cfe_time_interface_cfg.h`.

**12.133.2.2 CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE** #define CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE false  
Definition at line 89 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.3 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI** #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI true

**Purpose** Default Time Format

**Description:**

The following definitions select either UTC or TAI as the default (mission specific) time format. Although it is possible for an application to request time in a specific format, most callers should use [CFE\\_TIME\\_GetTime\(\)](#), which returns time in the default format. This avoids having to modify each individual caller when the default choice is changed.

**Limits**

if CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI is defined as true then CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC must be defined as false. if CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI is defined as false then CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC must be defined as true.

Definition at line 52 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.4 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC** #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC false  
Definition at line 53 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.5 CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE** #define CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE true

**Purpose** Default Time Format

**Description:**

The following definition enables the use of a simulated time at the tone signal using a software bus message.

**Limits**

Not Applicable

Definition at line 65 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.6 CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS 0  
Definition at line 147 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.7 CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS 1000  
Definition at line 148 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.8 CFE\_MISSION\_TIME\_DEF\_LEAPS** #define CFE\_MISSION\_TIME\_DEF\_LEAPS 37  
Definition at line 145 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.9 CFE\_MISSION\_TIME\_DEF\_MET\_SECS** #define CFE\_MISSION\_TIME\_DEF\_MET\_SECS 1000

**Purpose** Default Time Values

**Description:**

Default time values are provided to avoid problems due to time calculations performed after startup but before commands can be processed. For example, if the default time format is UTC then it is important that the sum of MET and STCF always exceed the value of Leap Seconds to prevent the UTC time calculation ( $\text{time} = \text{MET} + \text{STCF} - \text{Leap Seconds}$ ) from resulting in a negative (very large) number.

Some past missions have also created known (albeit wrong) default timestamps. For example, assume the epoch is defined as Jan 1, 1970 and further assume the default time values are set to create a timestamp of Jan 1, 2000. Even though the year 2000 timestamps are wrong, it may be of value to keep the time within some sort of bounds acceptable to the software.

Note: Sub-second units are in micro-seconds (0 to 999,999) and all values must be defined

**Limits**

Not Applicable

Definition at line 139 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.10 CFE\_MISSION\_TIME\_DEF\_MET\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_MET\_SUBS 0

Definition at line 140 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.11 CFE\_MISSION\_TIME\_DEF\_STCF\_SECS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SECS 1000000

Definition at line 142 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.12 CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS 0

Definition at line 143 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.13 CFE\_MISSION\_TIME\_EPOCH\_DAY** #define CFE\_MISSION\_TIME\_EPOCH\_DAY 1

Definition at line 166 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.14 CFE\_MISSION\_TIME\_EPOCH\_HOUR** #define CFE\_MISSION\_TIME\_EPOCH\_HOUR 0

Definition at line 167 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.15 CFE\_MISSION\_TIME\_EPOCH\_MICROS** #define CFE\_MISSION\_TIME\_EPOCH\_MICROS 0

Definition at line 170 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.16 CFE\_MISSION\_TIME\_EPOCH\_MINUTE** #define CFE\_MISSION\_TIME\_EPOCH\_MINUTE 0

Definition at line 168 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.17 CFE\_MISSION\_TIME\_EPOCH\_SECOND** #define CFE\_MISSION\_TIME\_EPOCH\_SECOND 0  
Definition at line 169 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.18 CFE\_MISSION\_TIME\_EPOCH\_YEAR** #define CFE\_MISSION\_TIME\_EPOCH\_YEAR 1980

**Purpose** Default EPOCH Values

**Description:**

Default ground time epoch values Note: these values are used only by the [CFE\\_TIME\\_Print\(\)](#) API function

**Limits**

Year - must be within 136 years Day - Jan 1 = 1, Feb 1 = 32, etc. Hour - 0 to 23 Minute - 0 to 59 Second - 0 to 59  
Micros - 0 to 999999

Definition at line 165 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.19 CFE\_MISSION\_TIME\_FS\_FACTOR** #define CFE\_MISSION\_TIME\_FS\_FACTOR 789004800

**Purpose** Time File System Factor

**Description:**

Define the s/c vs file system time conversion constant...

Note: this value is intended for use only by CFE TIME API functions to convert time values based on the ground system epoch (s/c time) to and from time values based on the file system epoch (fs time).

FS time = S/C time + factor S/C time = FS time - factor

Worksheet:

S/C epoch = Jan 1, 2005 (LRO ground system epoch) FS epoch = Jan 1, 1980 (vxWorks DOS file system epoch)

Delta = 25 years, 0 days, 0 hours, 0 minutes, 0 seconds

Leap years = 1980, 1984, 1988, 1992, 1996, 2000, 2004 (divisible by 4 – except if by 100 – unless also by 400)

1 year = 31,536,000 seconds 1 day = 86,400 seconds 1 hour = 3,600 seconds 1 minute = 60 seconds

25 years = 788,400,000 seconds 7 extra leap days = 604,800 seconds

total delta = 789,004,800 seconds

**Limits**

Not Applicable

Definition at line 208 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.20 CFE\_MISSION\_TIME\_MAX\_ELAPSED** #define CFE\_MISSION\_TIME\_MAX\_ELAPSED 200000

Definition at line 114 of file default\_cfe\_time\_interface\_cfg.h.

**12.133.2.21 CFE\_MISSION\_TIME\_MIN\_ELAPSED** #define CFE\_MISSION\_TIME\_MIN\_ELAPSED 0

**Purpose** Min and Max Time Elapsed

**Description:**

Based on the definition of Time and Tone Order (CFE\_MISSION\_TIME\_AT\_TONE\_WAS/WILL\_BE) either the "time at the tone" signal or data packet will follow the other. This definition sets the valid window of time for the second of the pair to lag behind the first. Time Services will invalidate both the tone and packet if the second does not arrive within this window following the first.

For example, if the data packet follows the tone, it might be valid for the data packet to arrive between zero and 100,000 micro-seconds after the tone. But, if the tone follows the packet, it might be valid only if the packet arrived between 200,000 and 700,000 micro-seconds before the tone.

Note: units are in micro-seconds

**Limits**

0 to 999,999 decimal

Definition at line 113 of file default\_cfe\_time\_interface\_cfg.h.

## **12.134 cfe/modules/time/config/default\_cfe\_time\_internal\_cfg.h File Reference**

**Macros**

- #define CFE\_PLATFORM\_TIME\_CFG\_SERVER true
- #define CFE\_PLATFORM\_TIME\_CFG\_CLIENT false
- #define CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL true
- #define CFE\_PLATFORM\_TIME\_CFG\_SIGNAL false
- #define CFE\_PLATFORM\_TIME\_CFG\_SOURCE false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME false
- #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS 0
- #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS 500000
- #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS 27
- #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS 0
- #define CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT 20000
- #define CFE\_PLATFORM\_TIME\_CFG\_START\_FLY 2
- #define CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY 8
- #define CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY 60
- #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY 25
- #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY 25
- #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE 8192

### **12.134.1 Detailed Description**

#### CFE Time Service (CFE\_TIME) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

**Note**

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.134.2 Macro Definition Documentation

**12.134.2.1 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY 25

Definition at line 222 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.2 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE 8192

Definition at line 241 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.3 CFE\_PLATFORM\_TIME\_CFG\_CLIENT** #define CFE\_PLATFORM\_TIME\_CFG\_CLIENT false

Definition at line 48 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.4 CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY** #define CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY 8

**Purpose** Define Periodic Time to Update Local Clock Tone Latch

### Description:

Define Periodic Time to Update Local Clock Tone Latch. Applies only when in flywheel mode. This define dictates the period at which the simulated 'last tone' time is updated. Units are seconds.

### Limits

Not Applicable

Definition at line 205 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.5 CFE\_PLATFORM\_TIME\_CFG\_SERVER** #define CFE\_PLATFORM\_TIME\_CFG\_SERVER true

**Purpose** Time Server or Time Client Selection

### Description:

This configuration parameter selects whether the Time task functions as a time "server" or "client". A time server generates the "time at the tone" packet which is received by time clients.

### Limits

Enable one, and only one by defining either CFE\_PLATFORM\_TIME\_CFG\_SERVER or CFE\_PLATFORM\_TIME\_CFG\_CLIENT AS true. The other must be defined as false.

Definition at line 47 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.6 CFE\_PLATFORM\_TIME\_CFG\_SIGNAL** #define CFE\_PLATFORM\_TIME\_CFG\_SIGNAL false

**Purpose** Include or Exclude the Primary/Redundant Tone Selection Cmd

**Description:**

Depending on the specific hardware system configuration, it may be possible to switch between a primary and redundant tone signal. If supported by hardware, this definition will enable command interfaces to select the active tone signal. Both Time Clients and Time Servers support this feature. Note: Set the CFE\_PLATFORM\_TIME\_CFG\_SIGNAL define to true to enable tone signal commands.

**Limits**

Not Applicable

Definition at line 95 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.7 CFE\_PLATFORM\_TIME\_CFG\_SOURCE** #define CFE\_PLATFORM\_TIME\_CFG\_SOURCE false

**Purpose** Include or Exclude the Internal/External Time Source Selection Cmd

**Description:**

By default, Time Servers maintain time using an internal MET which may be a h/w register or software counter, depending on available hardware. The following definition enables command interfaces to switch between an internal MET, or external time data received from one of several supported external time sources. Only a Time Server may be configured to use external time data. Note: Set the CFE\_PLATFORM\_TIME\_CFG\_SOURCE define to true to include the Time Source Selection Command (command allows selection between the internal or external time source). Then choose the external source with the CFE\_TIME\_CFG\_SRC\_??? define.

**Limits**

Only applies if [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) is set to true.

Definition at line 115 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.8 CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS** #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS false

Definition at line 132 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.9 CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET** #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET false

**Purpose** Choose the External Time Source for Server only

**Description:**

If [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) is set to true, then one of the following external time source types must also be set to true. Do not set any of the external time source types to true unless [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) is set to true.

**Limits**

1. If [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) is set to true then one and only one of the following three external time sources can and must be set true: [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_MET](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_GPS](#), [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_TIME](#)
2. Only applies if [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) is set to true.

Definition at line 131 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.10 CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME** #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME false  
Definition at line 133 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.11 CFE\_PLATFORM\_TIME\_CFG\_START\_FLY** #define CFE\_PLATFORM\_TIME\_CFG\_START\_FLY 2

**Purpose** Define Time to Start Flywheel Since Last Tone

Description:

Define time to enter flywheel mode (in seconds since last tone data update) Units are microseconds as measured with the local clock.

Limits

Not Applicable

Definition at line 192 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.12 CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT** #define CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT 20000

**Purpose** Define Timing Limits From One Tone To The Next

Description:

Defines limits to the timing of the 1Hz tone signal. A tone signal is valid only if it arrives within one second (plus or minus the tone limit) from the previous tone signal.Units are microseconds as measured with the local clock.

Limits

Not Applicable

Definition at line 180 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.13 CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL** #define CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL true

**Purpose** Time Tone In Big-Endian Order

Description:

If this configuration parameter is defined, the CFE time server will publish time tones with payloads in big-endian order, and time clients will expect the tones to be in big-endian order. This is useful for mixed-endian environments. This will become obsolete once EDS is available and the CFE time tone message is defined.

**Purpose** Local MET or Virtual MET Selection for Time Servers

Description:

Depending on the specific hardware system configuration, it may be possible for Time Servers to read the "local" MET from a h/w register rather than having to track the MET as the count of tone signal interrupts (virtual MET)

Time Clients must be defined as using a virtual MET. Also, a Time Server cannot be defined as having both a h/w MET and an external time source (they both cannot synchronize to the same tone).

Note: "disable" this define (set to false) only for Time Servers with local hardware that supports a h/w MET that is synchronized to the tone signal !!!

Limits

Only applies if [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) is set to true.

Definition at line 80 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.14 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS** #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS 0

**Purpose** Define the Max Delta Limits for Time Servers using an Ext Time Source

Description:

If `CFE_PLATFORM_TIME_CFG_SOURCE` is set to true and one of the external time sources is also set to true, then the delta time limits for range checking is used.

When a new time value is received from an external source, the value is compared against the "expected" time value. If the delta exceeds the following defined amount, then the new time data will be ignored. This range checking is only performed after the clock state has been commanded to "valid". Until then, external time data is accepted unconditionally.

Limits

Applies only if both `CFE_PLATFORM_TIME_CFG_SERVER` and `CFE_PLATFORM_TIME_CFG_SOURCE` are set to true.

Definition at line 152 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.15 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS** #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SU←  
BS 500000

Definition at line 153 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.16 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS** #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS 27

**Purpose** Define the Local Clock Rollover Value in seconds and subseconds

Description:

Specifies the capability of the local clock. Indicates the time at which the local clock rolls over.

Limits

Not Applicable

Definition at line 165 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.17 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS** #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS 0

Definition at line 166 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.18 CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_START\_TASK\_PR←  
ORITY 60

**Purpose** Define TIME Task Priorities

Description:

Defines the cFE\_TIME Task priority. Defines the cFE\_TIME Tone Task priority. Defines the cFE\_TIME 1HZ Task priority.

Limits

There is a lower limit of zero and an upper limit of 255 on these configuration parameters. Remember that the meaning of each task priority is inverted – a "lower" number has a "higher" priority.

Definition at line 220 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.19 CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define TIME Task Stack Sizes

**Description:**

Defines the cFE\_TIME Main Task Stack Size Defines the cFE\_TIME Tone Task Stack Size Defines the cFE\_TIME 1HZ Task Stack Size

**Limits**

There is a lower limit of 2048 on these configuration parameters. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 239 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.20 CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIO\_PRIORITY 25

Definition at line 221 of file default\_cfe\_time\_internal\_cfg.h.

**12.134.2.21 CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096

Definition at line 240 of file default\_cfe\_time\_internal\_cfg.h.

## 12.135 cfe/modules/time/config/default\_cfe\_time\_mission\_cfg.h File Reference

```
#include "cfe_time_interface_cfg.h"
```

### 12.135.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

**Note**

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.136 cfe/modules/time/config/default\_cfe\_time\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_time_msgdefs.h"
#include "cfe_time_msgstruct.h"
```

### 12.136.1 Detailed Description

Specification for the CFE Time Services (CFE\_TIME) command and telemetry message data types.  
This is a compatibility header for the "cfe\_time\_msg.h" file that has traditionally provided the message definitions for cFS apps.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.137 cfe/modules/time/config/default\_cfe\_time\_msgdefs.h File Reference

```
#include "cfe_time_fcncodes.h"
```

#### 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.*

### 12.137.1 Detailed Description

Specification for the CFE Time Services (CFE\_TIME) command and telemetry message constant definitions.  
For CFE\_TIME this is only the function/command code definitions

## 12.138 cfe/modules/time/config/default\_cfe\_time\_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_time_topicids.h"
```

### Macros

- #define CFE\_TIME\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TIME\_CMD\_MSG /\* 0x1805 \*/
- #define CFE\_TIME\_SEND\_HK\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TIME\_SEND\_HK\_MSG /\* 0x180D \*/
- #define CFE\_TIME\_TONE\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TIME\_TONE\_CMD\_MSG /\* 0x1810 \*/
- #define CFE\_TIME\_1HZ\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TIME\_1HZ\_CMD\_MSG /\* 0x1811 \*/
- #define CFE\_TIME\_DATA\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE\_GLOB + CFE\_MISSION\_TIME\_DATA\_CMD\_MSG /\* 0x1860 \*/
- #define CFE\_TIME\_SEND\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE\_GLOB + CFE\_MISSION\_TIME\_SEND\_CMD\_MSG /\* 0x1862 \*/
- #define CFE\_TIME\_HK\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_TIME\_HK\_TLM\_MSG /\* 0x0805 \*/
- #define CFE\_TIME\_DIAG\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_TIME\_DIAG\_TLM\_MSG /\* 0x0806 \*/

### 12.138.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Message IDs

### 12.138.2 Macro Definition Documentation

**12.138.2.1 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 35 of file default\_cfe\_time\_msgids.h.

**12.138.2.2 CFE\_TIME\_CMD\_MID** #define CFE\_TIME\_CMD\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TIME\_CMD\_MSG /\* 0x1805 \*/  
Definition at line 32 of file default\_cfe\_time\_msgids.h.

**12.138.2.3 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 40 of file default\_cfe\_time\_msgids.h.

**12.138.2.4 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 47 of file default\_cfe\_time\_msgids.h.

**12.138.2.5 CFE\_TIME\_HK\_TLM\_MID** #define CFE\_TIME\_HK\_TLM\_MID CFE\_PLATFORM\_TLM\_MID\_BASE + CFE\_MISSION\_TIME\_HK\_TLM\_MID  
/\* 0x0805 \*/  
Definition at line 46 of file default\_cfe\_time\_msgids.h.

**12.138.2.6 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 41 of file default\_cfe\_time\_msgids.h.

**12.138.2.7 CFE\_TIME\_SEND\_HK\_MID** #define CFE\_TIME\_SEND\_HK\_MID CFE\_PLATFORM\_CMD\_MID\_BASE + CFE\_MISSION\_TIME\_SEND\_HK\_MID  
/\* 0x180D \*/  
Definition at line 33 of file default\_cfe\_time\_msgids.h.

**12.138.2.8 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 34 of file default\_cfe\_time\_msgids.h.

## 12.139 cfe/modules/time/config/default\_cfe\_time\_msgstruct.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_msg_hdr.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](#)

## 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 [CFE\\_TIME\\_NoArgsCmd\\_t](#) [CFE\\_TIME\\_SendHkCmd\\_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\\_SetMETCCmd\\_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`

### 12.139.1 Detailed Description

Purpose: cFE Executive Services (TIME) Command and Telemetry packet definition file.

References: Flight Software Branch C Coding Standard Version 1.0a cFE Flight Software Application Developers Guide

Notes:

### 12.139.2 Typedef Documentation

#### 12.139.2.1 `CFE_TIME_1HzCmd_t` `typedef CFE_TIME_NoArgsCmd_t CFE_TIME_1HzCmd_t`

Definition at line 57 of file default\_cfe\_time\_msgstruct.h.

#### 12.139.2.2 `CFE_TIME_Add1HZAdjustmentCmd_t` `typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Add1HZAdjustmentCmd_t`

Definition at line 192 of file default\_cfe\_time\_msgstruct.h.

#### 12.139.2.3 `CFE_TIME_AddAdjustCmd_t` `typedef CFE_TIME_TimeCmd_t CFE_TIME_AddAdjustCmd_t`

Definition at line 165 of file default\_cfe\_time\_msgstruct.h.

#### 12.139.2.4 `CFE_TIME_AddDelayCmd_t` `typedef CFE_TIME_TimeCmd_t CFE_TIME_AddDelayCmd_t`

Definition at line 161 of file default\_cfe\_time\_msgstruct.h.

#### 12.139.2.5 `CFE_TIME_DiagnosticTlm_Payload_t` `typedef struct CFE_TIME_DiagnosticTlm_Payload CFE_TIME_DiagnosticTlm_Payload_t`

**Name** Time Services Diagnostics Packet

#### 12.139.2.6 `CFE_TIME_DiagnosticTlm_t` `typedef struct CFE_TIME_DiagnosticTlm CFE_TIME_DiagnosticTlm_t`

**12.139.2.7 CFE\_TIME\_FakeToneCmd\_t** `typedef CFE_TIME_NoArgsCmd_t CFE_TIME_FakeToneCmd_t`  
Definition at line 59 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.8 CFE\_TIME\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_TIME_HousekeepingTlm_Payload CFE_TIME_HousekeepingTlm_Payload_t`

**Name** Time Services Housekeeping Packet

**12.139.2.9 CFE\_TIME\_HousekeepingTlm\_t** `typedef struct CFE_TIME_HousekeepingTlm CFE_TIME_HousekeepingTlm_t`

**12.139.2.10 CFE\_TIME\_LeapsCmd\_Payload\_t** `typedef struct CFE_TIME_LeapsCmd_Payload CFE_TIME_LeapsCmd_Payload_t`  
Set leap seconds command payload.

**12.139.2.11 CFE\_TIME\_NoArgsCmd\_t** `typedef struct CFE_TIME_NoArgsCmd CFE_TIME_NoArgsCmd_t`  
Generic no argument command.

**12.139.2.12 CFE\_TIME\_NoopCmd\_t** `typedef CFE_TIME_NoArgsCmd_t CFE_TIME_NoopCmd_t`  
Definition at line 54 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.13 CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t** `typedef struct CFE_TIME_OneHzAdjustmentCmd_Payload CFE_TIME_OneHzAdjustmentCmd_Payload_t`  
Generic seconds, subseconds command payload.

**12.139.2.14 CFE\_TIME\_OneHzAdjustmentCmd\_t** `typedef struct CFE_TIME_OneHzAdjustmentCmd CFE_TIME_OneHzAdjustment_t`  
Generic seconds, subseconds adjustment command.

**12.139.2.15 CFE\_TIME\_ResetCountersCmd\_t** `typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ResetCountersCmd_t`  
Definition at line 55 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.16 CFE\_TIME\_SendDiagnosticCmd\_t** `typedef CFE_TIME_NoArgsCmd_t CFE_TIME_SendDiagnosticCmd_t`  
Definition at line 56 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.17 CFE\_TIME\_SendHkCmd\_t** `typedef CFE_TIME_NoArgsCmd_t CFE_TIME_SendHkCmd_t`  
Definition at line 60 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.18 CFE\_TIME\_SetLeapSecondsCmd\_t** `typedef struct CFE_TIME_SetLeapSecondsCmd CFE_TIME_SetLeapSecondsCmd_t`  
Set leap seconds command.

**12.139.2.19 CFE\_TIME\_SetMETCmd\_t** `typedef CFE_TIME_TimeCmd_t CFE_TIME_SetMETCmd_t`  
Definition at line 163 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.20 CFE\_TIME\_SetSignalCmd\_t** `typedef struct CFE_TIME_SetSignalCmd CFE_TIME_SetSignalCmd_t`  
Set tone signal source command.

**12.139.2.21 CFE\_TIME\_SetSourceCmd\_t** `typedef struct CFE_TIME_SetSourceCmd CFE_TIME_SetSourceCmd_t`  
Set time data source command.

**12.139.2.22 CFE\_TIME\_SetStateCmd\_t** `typedef struct CFE_TIME_SetStateCmd CFE_TIME_SetStateCmd_t`  
Set clock state command.

**12.139.2.23 CFE\_TIME\_SetSTCFCmd\_t** `typedef CFE_TIME_TimeCmd_t CFE_TIME_SetSTCFCmd_t`  
Definition at line 164 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.24 CFE\_TIME\_SetTimeCmd\_t** `typedef CFE_TIME_TimeCmd_t CFE_TIME_SetTimeCmd_t`  
Definition at line 167 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.25 CFE\_TIME\_SignalCmd\_Payload\_t** `typedef struct CFE_TIME_SignalCmd_Payload CFE_TIME_SignalCmd_Payload_t`  
Set tone signal source command payload.

**12.139.2.26 CFE\_TIME\_SourceCmd\_Payload\_t** `typedef struct CFE_TIME_SourceCmd_Payload CFE_TIME_SourceCmd_Payload_t`  
Set time data source command payload.

**12.139.2.27 CFE\_TIME\_StateCmd\_Payload\_t** `typedef struct CFE_TIME_StateCmd_Payload CFE_TIME_StateCmd_Payload_t`  
Set clock state command payload.

**12.139.2.28 CFE\_TIME\_Sub1HZAdjustmentCmd\_t** `typedef CFE_TIME_OneHzAdjustmentCmd_t CFE_TIME_Sub1HZAdjustmentCmd_t`  
Definition at line 193 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.29 CFE\_TIME\_SubAdjustCmd\_t** `typedef CFE_TIME_TimeCmd_t CFE_TIME_SubAdjustCmd_t`  
Definition at line 166 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.30 CFE\_TIME\_SubDelayCmd\_t** `typedef CFE_TIME_TimeCmd_t CFE_TIME_SubDelayCmd_t`  
Definition at line 162 of file default\_cfe\_time\_msgstruct.h.

**12.139.2.31 CFE\_TIME\_TimeCmd\_Payload\_t** `typedef struct CFE_TIME_TimeCmd_Payload CFE_TIME_TimeCmd_Payload_t`  
Generic seconds, microseconds command payload.

**12.139.2.32 CFE\_TIME\_TimeCmd\_t** `typedef struct CFE_TIME_TimeCmd CFE_TIME_TimeCmd_t`  
Generic seconds, microseconds argument command.

**12.139.2.33 CFE\_TIME\_ToneDataCmd\_Payload\_t** `typedef struct CFE_TIME_ToneDataCmd_Payload CFE_TIME_ToneDataCmd_Pa`  
Time at tone data command payload.

**12.139.2.34 CFE\_TIME\_ToneDataCmd\_t** `typedef struct CFE_TIME_ToneDataCmd CFE_TIME_ToneDataCmd_t`  
Time at tone data command.

**12.139.2.35 CFE\_TIME\_ToneSignalCmd\_t** `typedef CFE_TIME_NoArgsCmd_t CFE_TIME_ToneSignalCmd_t`  
Definition at line 58 of file default\_cfe\_time\_msgstruct.h.

## 12.140 cfe/modules/time/config/default\_cfe\_time\_platform\_cfg.h File Reference

```
#include "cfe_time_mission_cfg.h"
#include "cfe_time_internal_cfg.h"
```

### 12.140.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Platform Configuration Header File

This is a compatibility header for the "platform\_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.141 cfe/modules/time/config/default\_cfe\_time\_topicids.h File Reference

### Macros

- #define CFE\_MISSION\_TIME\_CMD\_MSG 5
- #define CFE\_MISSION\_TIME\_SEND\_HK\_MSG 13
- #define CFE\_MISSION\_TIME\_TONE\_CMD\_MSG 16
- #define CFE\_MISSION\_TIME\_1HZ\_CMD\_MSG 17
- #define CFE\_MISSION\_TIME\_DATA\_CMD\_MSG 0
- #define CFE\_MISSION\_TIME\_SEND\_CMD\_MSG 2
- #define CFE\_MISSION\_TIME\_HK\_TLM\_MSG 5
- #define CFE\_MISSION\_TIME\_DIAG\_TLM\_MSG 6

### 12.141.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Topic IDs

### 12.141.2 Macro Definition Documentation

**12.141.2.1 CFE\_MISSION\_TIME\_1HZ\_CMD\_MSG** #define CFE\_MISSION\_TIME\_1HZ\_CMD\_MSG 17  
Definition at line 38 of file default\_cfe\_time\_topicids.h.

**12.141.2.2 CFE\_MISSION\_TIME\_CMD\_MSG** #define CFE\_MISSION\_TIME\_CMD\_MSG 5

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE command messages

**Limits**

Not Applicable

Definition at line 35 of file default\_cfe\_time\_topicids.h.

**12.141.2.3 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

**Limits**

Not Applicable

Definition at line 49 of file default\_cfe\_time\_topicids.h.

**12.141.2.4 CFE\_MISSION\_TIME\_DIAG\_TLM\_MSG** #define CFE\_MISSION\_TIME\_DIAG\_TLM\_MSG 6

Definition at line 62 of file default\_cfe\_time\_topicids.h.

**12.141.2.5 CFE\_MISSION\_TIME\_HK\_TLM\_MSG** #define CFE\_MISSION\_TIME\_HK\_TLM\_MSG 5

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE telemetry messages

**Limits**

Not Applicable

Definition at line 61 of file default\_cfe\_time\_topicids.h.

**12.141.2.6 CFE\_MISSION\_TIME\_SEND\_CMD\_MSG** #define CFE\_MISSION\_TIME\_SEND\_CMD\_MSG 2

Definition at line 50 of file default\_cfe\_time\_topicids.h.

**12.141.2.7 CFE\_MISSION\_TIME\_SEND\_HK\_MSG** #define CFE\_MISSION\_TIME\_SEND\_HK\_MSG 13  
Definition at line 36 of file default\_cfe\_time\_topicids.h.

**12.141.2.8 CFE\_MISSION\_TIME\_TONE\_CMD\_MSG** #define CFE\_MISSION\_TIME\_TONE\_CMD\_MSG 16  
Definition at line 37 of file default\_cfe\_time\_topicids.h.

## 12.142 cfe/modules/time/fsw/inc/cfe\_time\_eventids.h File Reference

### Macros

#### TIME event IDs

- #define **CFE\_TIME\_INIT\_EID** 1  
*TIME Initialization Event ID.*
- #define **CFE\_TIME\_NOOP\_EID** 4  
*TIME No-op Command Success Event ID.*
- #define **CFE\_TIME\_RESET\_EID** 5  
*TIME Reset Counters Command Success Event ID.*
- #define **CFE\_TIME\_DIAG\_EID** 6  
*TIME Request Diagnostics Command Success Event ID.*
- #define **CFE\_TIME\_STATE\_EID** 7  
*TIME Set Time State Command Success Event ID.*
- #define **CFE\_TIME\_SOURCE\_EID** 8  
*TIME Set Time Source Command Success Event ID.*
- #define **CFE\_TIME\_SIGNAL\_EID** 9  
*TIME Set Tone Source Command Success Event ID.*
- #define **CFE\_TIME\_DELAY\_EID** 11  
*TIME Add or Subtract Delay Command Success Event ID.*
- #define **CFE\_TIME\_TIME\_EID** 12  
*TIME Set Time Command Success Event ID.*
- #define **CFE\_TIME\_MET\_EID** 13  
*TIME Set Mission Elapsed Time Command Success Event ID.*
- #define **CFE\_TIME\_STCF\_EID** 14  
*TIME Set Spacecraft Time Correlation Factor Command Success Event ID.*
- #define **CFE\_TIME\_DELTA\_EID** 15  
*TIME Add or Subtract Single STCF Adjustment Command Success Event ID.*
- #define **CFE\_TIME\_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.*

### 12.142.1 Detailed Description

cFE Time Services Event IDs

### 12.142.2 Macro Definition Documentation

#### 12.142.2.1 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\_eventids.h.

**12.142.2.2 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\_eventids.h.

**12.142.2.3 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\_eventids.h.

**12.142.2.4 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\_eventids.h.

**12.142.2.5 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\_eventids.h.

**12.142.2.6 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\_eventids.h.

**12.142.2.7 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\_eventids.h.

**12.142.2.8 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\_eventids.h.

**12.142.2.9 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\_eventids.h.

**12.142.2.10 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\_eventids.h.

**12.142.2.11 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\_eventids.h.

**12.142.2.12 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\_eventids.h.

**12.142.2.13 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\_eventids.h.

**12.142.2.14 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\_eventids.h.

**12.142.2.15 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\_eventids.h.

**12.142.2.16 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\_eventids.h.

**12.142.2.17 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\_eventids.h.

**12.142.2.18 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\_eventids.h.

**12.142.2.19 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\_eventids.h.

**12.142.2.20 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\_eventids.h.

**12.142.2.21 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\_eventids.h.

**12.142.2.22 CFE\_TIME\_RESET\_EID** #define CFE\_TIME\_RESET\_EID 5  
TIME Reset Counters Command Success Event ID.

Type: DEBUG

Cause:

[TIME Reset Counters Command](#) success.  
Definition at line 64 of file cfe\_time\_eventids.h.

**12.142.2.23 CFE\_TIME\_SIGNAL\_CFG\_EID** #define CFE\_TIME\_SIGNAL\_CFG\_EID 41  
TIME Set Clock Signal Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Clock Signal Command](#) failure due to being in an incompatible mode.  
Definition at line 351 of file cfe\_time\_eventids.h.

**12.142.2.24 CFE\_TIME\_SIGNAL\_EID** #define CFE\_TIME\_SIGNAL\_EID 9  
TIME Set Tone Source Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Clock Tone Source Command](#) success.  
Definition at line 108 of file cfe\_time\_eventids.h.

**12.142.2.25 CFE\_TIME\_SIGNAL\_ERR\_EID** #define CFE\_TIME\_SIGNAL\_ERR\_EID 32  
TIME Set Clock Tone Source Command Invalid Source Event ID.

Type: ERROR

Cause:

[Set Clock Tone Source Command](#) failed due to invalid source requested.  
Definition at line 265 of file cfe\_time\_eventids.h.

**12.142.2.26 CFE\_TIME\_SOURCE\_CFG\_EID** #define CFE\_TIME\_SOURCE\_CFG\_EID 40  
TIME Set Clock Source Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Clock Source Command](#) failure due to being in an incompatible mode.  
Definition at line 339 of file cfe\_time\_eventids.h.

**12.142.2.27 CFE\_TIME\_SOURCE\_EID** #define CFE\_TIME\_SOURCE\_EID 8  
TIME Set Time Source Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Time Source Command](#) success.  
Definition at line 97 of file cfe\_time\_eventids.h.

**12.142.2.28 CFE\_TIME\_SOURCE\_ERR\_EID** #define CFE\_TIME\_SOURCE\_ERR\_EID 31  
TIME Set Clock Source Command Invalid Source Event ID.

Type: ERROR

Cause:

[TIME Set Clock Source Command](#) failed due to invalid source requested.  
Definition at line 254 of file cfe\_time\_eventids.h.

**12.142.2.29 CFE\_TIME\_STATE\_EID** #define CFE\_TIME\_STATE\_EID 7  
TIME Set Time State Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Time State Command](#) success.  
Definition at line 86 of file cfe\_time\_eventids.h.

**12.142.2.30 CFE\_TIME\_STATE\_ERR\_EID** #define CFE\_TIME\_STATE\_ERR\_EID 30  
TIME Set Clock State Command Invalid State Event ID.

Type: ERROR

Cause:

[TIME Set Clock State Command](#) failed due to invalid state requested.  
Definition at line 243 of file cfe\_time\_eventids.h.

**12.142.2.31 CFE\_TIME\_STCF\_CFG\_EID** #define CFE\_TIME\_STCF\_CFG\_EID 45  
TIME Set Spacecraft Time Correlation Factor Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Spacecraft Time Correlation Factor Command](#) failure due to being in an incompatible mode.  
Definition at line 400 of file cfe\_time\_eventids.h.

**12.142.2.32 CFE\_TIME\_STCF\_EID** #define CFE\_TIME\_STCF\_EID 14  
TIME Set Spacecraft Time Correlation Factor Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Spacecraft Time Correlation Factor Command](#) success.  
Definition at line 153 of file cfe\_time\_eventids.h.

**12.142.2.33 CFE\_TIME\_STCF\_ERR\_EID** #define CFE\_TIME\_STCF\_ERR\_EID 36  
TIME Set Spacecraft Time Correlation Factor Command Invalid Time Value Event ID.

Type: ERROR

Cause:

[TIME Set Spacecraft Time Correlation Factor Command](#) failure due to an invalid time value.  
Definition at line 314 of file cfe\_time\_eventids.h.

**12.142.2.34 CFE\_TIME\_TIME\_CFG\_EID** #define CFE\_TIME\_TIME\_CFG\_EID 43  
TIME Set Spacecraft Time Command Incompatible Mode Event ID.

Type: ERROR

Cause:

TIME Set Spacecraft Time Command failure due to being in an incompatible mode.  
Definition at line 376 of file cfe\_time\_eventids.h.

**12.142.2.35 CFE\_TIME\_TIME\_EID** #define CFE\_TIME\_TIME\_EID 12  
TIME Set Time Command Success Event ID.

Type: INFORMATION

Cause:

TIME Set Time Command success.  
Definition at line 131 of file cfe\_time\_eventids.h.

**12.142.2.36 CFE\_TIME\_TIME\_ERR\_EID** #define CFE\_TIME\_TIME\_ERR\_EID 34  
TIME Set Spacecraft Time Command Invalid Time Value Event ID.

Type: ERROR

Cause:

TIME Set Spacecraft Time Command failure due to an invalid time value.  
Definition at line 290 of file cfe\_time\_eventids.h.

## 12.143 osal/docs/src/osal\_frontpage.dox File Reference

### 12.144 osal/docs/src/osal\_fs.dox File Reference

### 12.145 osal/docs/src/osal\_timer.dox File Reference

### 12.146 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)

### 12.146.1 Detailed Description

Purpose: Unit specification for common types.

Design Notes: Assumes make file has defined processor family

## 12.146.2 Macro Definition Documentation

### 12.146.2.1 `_EXTENSION_` #define \_EXTENSION\_

Definition at line 65 of file common\_types.h.

### 12.146.2.2 `CompileTimeAssert` #define CompileTimeAssert(

*Condition,*

*Message* ) typedef char Message[(*Condition*) ? 1 : -1]

Definition at line 48 of file common\_types.h.

### 12.146.2.3 `OS_PRINTF` #define OS\_PRINTF(

*n,*

*m* )

Definition at line 67 of file common\_types.h.

### 12.146.2.4 `OS_USED` #define OS\_USED

Definition at line 66 of file common\_types.h.

### 12.146.2.5 `OSAL_BLOCKCOUNT_C` #define OSAL\_BLOCKCOUNT\_C(

*X* ) ((*osal\_blockcount\_t*)(*X*))

Definition at line 172 of file common\_types.h.

### 12.146.2.6 `OSAL_INDEX_C` #define OSAL\_INDEX\_C(

*X* ) ((*osal\_index\_t*)(*X*))

Definition at line 173 of file common\_types.h.

### 12.146.2.7 `OSAL_OBJTYPE_C` #define OSAL\_OBJTYPE\_C(

*X* ) ((*osal\_objtype\_t*)(*X*))

Definition at line 174 of file common\_types.h.

### 12.146.2.8 `OSAL_SIZE_C` #define OSAL\_SIZE\_C(

*X* ) ((*size\_t*)(*X*))

Definition at line 171 of file common\_types.h.

### 12.146.2.9 `OSAL_STATUS_C` #define OSAL\_STATUS\_C(

*X* ) ((*osal\_status\_t*)(*X*))

Definition at line 175 of file common\_types.h.

## 12.146.3 Typedef Documentation

**12.146.3.1 cpuaddr** `typedef uintptr_t cpuaddr`

Definition at line 88 of file common\_types.h.

**12.146.3.2 cpudiff** `typedef ptrdiff_t cpudiff`

Definition at line 90 of file common\_types.h.

**12.146.3.3 cpusize** `typedef size_t cpusize`

Definition at line 89 of file common\_types.h.

**12.146.3.4 int16** `typedef int16_t int16`

Definition at line 80 of file common\_types.h.

**12.146.3.5 int32** `typedef int32_t int32`

Definition at line 81 of file common\_types.h.

**12.146.3.6 int64** `typedef int64_t int64`

Definition at line 82 of file common\_types.h.

**12.146.3.7 int8** `typedef int8_t int8`

Definition at line 79 of file common\_types.h.

**12.146.3.8 intptr** `typedef intptr_t intptr`

Definition at line 87 of file common\_types.h.

**12.146.3.9 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.

**12.146.3.10 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.

**12.146.3.11 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.

**12.146.3.12 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.

**12.146.3.13 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.

**12.146.3.14 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.

**12.146.3.15 uint16** `typedef uint16_t uint16`

Definition at line 84 of file common\_types.h.

**12.146.3.16 uint32** `typedef uint32_t uint32`

Definition at line 85 of file common\_types.h.

**12.146.3.17 uint64** `typedef uint64_t uint64`

Definition at line 86 of file common\_types.h.

**12.146.3.18 uint8** `typedef uint8_t uint8`

Definition at line 83 of file common\_types.h.

## 12.146.4 Function Documentation

**12.146.4.1 CompileTimeAssert() [1/9]** `CompileTimeAssert (`

```
    sizeof(cpuaddr) >=sizeof(void *) ,  
    TypePtrWrongSize )
```

**12.146.4.2 CompileTimeAssert() [2/9]** `CompileTimeAssert (`

```
    sizeof(int16) == 2,  
    Typeint16WrongSize )
```

**12.146.4.3 CompileTimeAssert() [3/9]** `CompileTimeAssert (`

```
    sizeof(int32) == 4,  
    Typeint32WrongSize )
```

**12.146.4.4 CompileTimeAssert() [4/9]** `CompileTimeAssert (`  
    `sizeof(int64) = 8,`  
    `Typeint64WrongSize )`

**12.146.4.5 CompileTimeAssert() [5/9]** `CompileTimeAssert (`  
    `sizeof(int8) = 1,`  
    `Typeint8WrongSize )`

**12.146.4.6 CompileTimeAssert() [6/9]** `CompileTimeAssert (`  
    `sizeof(uint16) = 2,`  
    `TypeUint16WrongSize )`

**12.146.4.7 CompileTimeAssert() [7/9]** `CompileTimeAssert (`  
    `sizeof(uint32) = 4,`  
    `TypeUint32WrongSize )`

**12.146.4.8 CompileTimeAssert() [8/9]** `CompileTimeAssert (`  
    `sizeof(uint64) = 8,`  
    `TypeUint64WrongSize )`

**12.146.4.9 CompileTimeAssert() [9/9]** `CompileTimeAssert (`  
    `sizeof(uint8) = 1,`  
    `TypeUint8WrongSize )`

## 12.147 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)

- `int32 OS_BinSemGive (osal_id_t sem_id)`  
*Unblock all tasks pending on the specified semaphore.*
- `int32 OS_BinSemTake (osal_id_t sem_id)`  
*Increment the semaphore value.*
- `int32 OS_BinSemDelete (osal_id_t sem_id)`  
*Decrement the semaphore value.*
- `int32 OS_BinSemTimedWait (osal_id_t sem_id, uint32 msecs)`  
*Deletes the specified Binary Semaphore.*
- `int32 OS_BinSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`  
*Decrement the semaphore value with a timeout.*
- `int32 OS_BinSemGetInfo (osal_id_t sem_id, OS_bin_sem_prop_t *bin_prop)`  
*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.*

### 12.147.1 Detailed Description

Declarations and prototypes for binary semaphores

## 12.148 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)`

### 12.148.1 Detailed Description

Declarations and prototypes for OSAL BSP

## 12.149 osal/src/os/inc/osapi-clock.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Data Structures

- struct `OS_time_t`  
*OSAL time interval structure.*

### Enumerations

- enum { `OS_TIME_TICK_RESOLUTION_NS` = 100, `OS_TIME_TICKS_PER_SECOND` = 1000000000 / `OS_TIME_TICK_RESOLUTION_NS`, `OS_TIME_TICKS_PER_MSEC` = 1000000 / `OS_TIME_TICK_RESOLUTION_NS`, `OS_TIME_TICKS_PER_USEC` = 1000 / `OS_TIME_TICK_RESOLUTION_NS` }

*Multippliers/divisors to convert ticks into standardized units.*

## Functions

- `int32 OS_GetLocalTime (OS_time_t *time_struct)`  
*Get the local time.*
- `int32 OS_SetLocalTime (const OS_time_t *time_struct)`  
*Set the local time.*
- `static int64 OS_TimeGetTotalSeconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to whole number of seconds.*
- `static OS_time_t OS_TimeFromTotalSeconds (int64 tm)`  
*Get an `OS_time_t` interval object from an integer number of seconds.*
- `static int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to millisecond units.*
- `static OS_time_t OS_TimeFromTotalMilliseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of milliseconds.*
- `static int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to microsecond units.*
- `static OS_time_t OS_TimeFromTotalMicroseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of microseconds.*
- `static int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to nanosecond units.*
- `static OS_time_t OS_TimeFromTotalNanoseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of nanoseconds.*
- `static int64 OS_TimeGetFractionalPart (OS_time_t tm)`  
*Get subseconds portion (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetSubsecondsPart (OS_time_t tm)`  
*Get 32-bit normalized subseconds (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetMillisecondsPart (OS_time_t tm)`  
*Get milliseconds portion (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetMicrosecondsPart (OS_time_t tm)`  
*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.*

### 12.149.1 Detailed Description

Declarations and prototypes for osapi-clock module

## 12.149.2 Enumeration Type Documentation

### 12.149.2.1 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.

## 12.150 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.*

### 12.150.1 Detailed Description

Declarations and prototypes for general OSAL functions that are not part of a subsystem

### 12.150.2 Typedef Documentation

**12.150.2.1 OS\_EventHandler\_t** `typedef int32(* OS_EventHandler_t) (OS_Event_t event, osal_id_t object_id, void *data)`  
A callback routine for event handling.

#### Parameters

in	<i>event</i>	The event that occurred
in	<i>object_id</i>	The associated object_id, or 0 if not associated with an object
in, out	<i>data</i>	An abstract data/context object associated with the event, or NULL.

#### Returns

status Execution status, see [OSAL Return Code Defines](#).

Definition at line 98 of file osapi-common.h.

### 12.150.3 Enumeration Type Documentation

**12.150.3.1 OS\_Event\_t** `enum OS_Event_t`

A set of events that can be used with BSP event callback routines.

#### Enumerator

<code>OS_EVENT_RESERVED</code>	no-op/reserved event id value
<code>OS_EVENT_RESOURCE_ALLOCATED</code>	resource/id has been newly allocated but not yet created. This event is invoked from WITHIN the locked region, in the context of the task which is allocating the resource. If the handler returns non-success, the error will be returned to the caller and the creation process is aborted.

**Enumerator**

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.

## 12.151 osal/src/os/inc/osapi-condvar.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

### Data Structures

- struct **OS\_condvar\_prop\_t**  
*OSAL condition variable properties.*

### Functions

- **int32 OS\_CondVarCreate (osal\_id\_t \*var\_id, const char \*var\_name, uint32 options)**  
*Creates a condition variable resource.*
- **int32 OS\_CondVarLock (osal\_id\_t var\_id)**  
*Locks/Acquires the underlying mutex associated with a condition variable.*
- **int32 OS\_CondVarUnlock (osal\_id\_t var\_id)**  
*Unlocks/Releases the underlying mutex associated with a condition variable.*
- **int32 OS\_CondVarSignal (osal\_id\_t var\_id)**  
*Signals the condition variable resource referenced by var\_id.*
- **int32 OS\_CondVarBroadcast (osal\_id\_t var\_id)**  
*Broadcasts the condition variable resource referenced by var\_id.*
- **int32 OS\_CondVarWait (osal\_id\_t var\_id)**  
*Waits on the condition variable object referenced by var\_id.*
- **int32 OS\_CondVarTimedWait (osal\_id\_t var\_id, const OS\_time\_t \*abs\_wakeup\_time)**  
*Time-limited wait on the condition variable object referenced by var\_id.*
- **int32 OS\_CondVarDelete (osal\_id\_t var\_id)**

*Deletes the specified condition variable.*

- `int32 OS_CondVarGetIdByName (osal_id_t *var_id, const char *var_name)`  
*Find an existing condition variable ID by name.*
- `int32 OS_CondVarGetInfo (osal_id_t var_id, OS_condvar_prop_t *condvar_prop)`  
*Fill a property object buffer with details regarding the resource.*

### 12.151.1 Detailed Description

Declarations and prototypes for condition variables

## 12.152 osal/src/os/inc/osapi-constants.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Macros

- `#define OS_PEND (-1)`
- `#define OS_CHECK (0)`
- `#define OS_OBJECT_ID_UNDEFINED ((osal_id_t){0})`  
*Initializer for the osal\_id\_t type which will not match any valid value.*
- `#define OS_OBJECT_CREATOR_ANY OS_OBJECT_ID_UNDEFINED`  
*Constant that may be passed to `OS_ForEachObject()`/`OS_ForEachObjectType()` to match any creator (i.e. get all objects)*
- `#define OS_MAX_LOCAL_PATH_LEN (OS_MAX_PATH_LEN + OS_FS_PHYS_NAME_LEN)`  
*Maximum length of a local/native path name string.*

### 12.152.1 Detailed Description

General constants for OSAL that are shared across subsystems

### 12.152.2 Macro Definition Documentation

#### 12.152.2.1 OS\_CHECK `#define OS_CHECK (0)`

Definition at line 35 of file osapi-constants.h.

#### 12.152.2.2 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.

#### 12.152.2.3 OS\_OBJECT\_CREATOR\_ANY `#define OS_OBJECT_CREATOR_ANY OS_OBJECT_ID_UNDEFINED`

Constant that may be passed to `OS_ForEachObject()`/`OS_ForEachObjectType()` to match any creator (i.e. get all objects)

Definition at line 46 of file osapi-constants.h.

**12.152.2.4 OS\_OBJECT\_ID\_UNDEFINED** #define OS\_OBJECT\_ID\_UNDEFINED ((osal\_id\_t) {0})  
 Initializer for the osal\_id\_t type which will not match any valid value.  
 Definition at line 40 of file osapi-constants.h.

**12.152.2.5 OS\_PEND** #define OS\_PEND (-1)  
 Definition at line 34 of file osapi-constants.h.

## 12.153 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.*

### 12.153.1 Detailed Description

Declarations and prototypes for counting semaphores

## 12.154 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.*

### 12.154.1 Detailed Description

Declarations and prototypes for directories

### 12.154.2 Macro Definition Documentation

#### 12.154.2.1 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.

## 12.155 osal/src/os/inc/osapi-error.h File Reference

```
#include "common_types.h"
```

## Macros

- `#define OS_ERROR_NAME_LENGTH 35`  
*Error string name length.*
- `#define OS_STATUS_STRING_LENGTH 12`  
*Status converted to string length limit.*
- `#define OS_SUCCESS (0)`  
*Successful execution.*
- `#define OS_ERROR (-1)`  
*Failed execution.*
- `#define OS_INVALID_POINTER (-2)`  
*Invalid pointer.*
- `#define OS_ERROR_ADDRESS_MISALIGNED (-3)`

- `#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.*
- typedef char os\_status\_string\_t[OS\_STATUS\_STRING\_LENGTH]  
*For the `OS_StatusToString()` function, to ensure everyone is making an array of the same length.*

## Functions

- static long OS\_StatusToInteger (osal\_status\_t Status)  
*Convert a status code to a native "long" type.*
- int32 OS\_GetErrorName (int32 error\_num, os\_err\_name\_t \*err\_name)  
*Convert an error number to a string.*
- char \* OS\_StatusToString (osal\_status\_t status, os\_status\_string\_t \*status\_string)  
*Convert status to a string.*

### 12.155.1 Detailed Description

OSAL error code definitions

## 12.155.2 Macro Definition Documentation

### 12.155.2.1 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.

### 12.155.2.2 OS\_STATUS\_STRING\_LENGTH `#define OS_STATUS_STRING_LENGTH 12`

Status converted to string length limit.

Used for sizing `os_status_string_t` intended for use in printing `osal_status_t` values Sized to fit `LONG_MIN` including NULL termination

Definition at line 55 of file osapi-error.h.

## 12.155.3 Typedef Documentation

### 12.155.3.1 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.

### 12.155.3.2 os\_status\_string\_t `typedef char os_status_string_t[OS_STATUS_STRING_LENGTH]`

For the `OS_StatusToString()` function, to ensure everyone is making an array of the same length.

Definition at line 61 of file osapi-error.h.

## 12.156 osal/src/os/inc/osapi-file.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

## Data Structures

- struct `OS_file_prop_t`

*OSAL file properties.*

- struct `os_fstat_t`

*File system status.*

## Macros

- `#define OS_READ_ONLY 0`
- `#define OS_WRITE_ONLY 1`
- `#define OS_READ_WRITE 2`
- `#define OS_SEEK_SET 0`

- #define OS\_SEEK\_CUR 1
- #define OS\_SEEK\_END 2
- #define OS\_FILESTAT\_MODE(x) ((x). FileModeBits)  
*Access file stat mode bits.*
- #define OS\_FILESTAT\_ISDIR(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_DIR)  
*File stat is directory logical.*
- #define OS\_FILESTAT\_EXEC(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_EXEC)  
*File stat is executable logical.*
- #define OS\_FILESTAT\_WRITE(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_WRITE)  
*File stat is write enabled logical.*
- #define OS\_FILESTAT\_READ(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_READ)  
*File stat is read enabled logical.*
- #define OS\_FILESTAT\_SIZE(x) ((x). FileSize)  
*Access file stat size field.*
- #define OS\_FILESTAT\_TIME(x) (OS\_TimeGetTotalSeconds((x). FileTime))  
*Access file stat time field as a whole number of seconds.*

## Enumerations

- enum { OS\_FILESTAT\_MODE\_EXEC = 0x00001, OS\_FILESTAT\_MODE\_WRITE = 0x00002, OS\_FILESTAT\_MODE\_READ = 0x00004, OS\_FILESTAT\_MODE\_DIR = 0x10000 }  
*File stat mode bits.*
- enum OS\_file\_flag\_t { OS\_FILE\_FLAG\_NONE = 0x00, OS\_FILE\_FLAG\_CREATE = 0x01, OS\_FILE\_FLAG\_TRUNCATE = 0x02 }  
*Flags that can be used with opening of a file (bitmask)*

## Functions

- int32 OS\_OpenCreate (osal\_id\_t \*filedes, const char \*path, int32 flags, int32 access\_mode)  
*Open or create a file.*
- int32 OS\_close (osal\_id\_t filedes)  
*Closes an open file handle.*
- int32 OS\_read (osal\_id\_t filedes, void \*buffer, size\_t nbytes)  
*Read from a file handle.*
- int32 OS\_write (osal\_id\_t filedes, const void \*buffer, size\_t nbytes)  
*Write to a file handle.*
- int32 OS\_TimedRead (osal\_id\_t filedes, void \*buffer, size\_t nbytes, int32 timeout)  
*File/Stream input read with a timeout.*
- int32 OS\_TimedWrite (osal\_id\_t filedes, const void \*buffer, size\_t nbytes, int32 timeout)  
*File/Stream output write with a timeout.*
- int32 OS\_chmod (const char \*path, uint32 access\_mode)  
*Changes the permissions of a file.*
- int32 OS\_stat (const char \*path, os\_fstat\_t \*filestats)  
*Obtain information about a file or directory.*
- int32 OS\_lseek (osal\_id\_t filedes, int32 offset, uint32 whence)  
*Seeks to the specified position of an open file.*
- int32 OS\_remove (const char \*path)  
*Removes a file from the file system.*
- int32 OS\_rename (const char \*old\_filename, const char \*new\_filename)

*Renames a file.*

- [int32 OS\\_cp \(const char \\*src, const char \\*dest\)](#)  
*Copies a single file from src to dest.*
- [int32 OS\\_mv \(const char \\*src, const char \\*dest\)](#)  
*Move a single file from src to dest.*
- [int32 OS\\_FDGetInfo \(osal\\_id\\_t filedes, OS\\_file\\_prop\\_t \\*fd\\_prop\)](#)  
*Obtain information about an open file.*
- [int32 OS\\_FileOpenCheck \(const char \\*Filename\)](#)  
*Checks to see if a file is open.*
- [int32 OS\\_CloseAllFiles \(void\)](#)  
*Close all open files.*
- [int32 OS\\_CloseFileByName \(const char \\*Filename\)](#)  
*Close a file by filename.*

### 12.156.1 Detailed Description

Declarations and prototypes for file objects

### 12.156.2 Macro Definition Documentation

**12.156.2.1 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.

**12.156.2.2 OS\_FILESTAT\_ISDIR** `#define OS_FILESTAT_ISDIR (x) ((x). FileModeBits & OS_FILESTAT_MODE_DIR)`

File stat is directory logical.

Definition at line 90 of file osapi-file.h.

**12.156.2.3 OS\_FILESTAT\_MODE** `#define OS_FILESTAT_MODE (x) ((x). FileModeBits)`

Access file stat mode bits.

Definition at line 88 of file osapi-file.h.

**12.156.2.4 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.

**12.156.2.5 OS\_FILESTAT\_SIZE** `#define OS_FILESTAT_SIZE (x) ((x).FileSize)`

Access file stat size field.

Definition at line 98 of file osapi-file.h.

**12.156.2.6 OS\_FILESTAT\_TIME** #define OS\_FILESTAT\_TIME (x) (OS\_TimeGetTotalSeconds((x).FileTime))

Access file stat time field as a whole number of seconds.

Definition at line 100 of file osapi-file.h.

**12.156.2.7 OS\_FILESTAT\_WRITE** #define OS\_FILESTAT\_WRITE ((x). FileModeBits & OS\_FILESTAT\_MODE\_WRITE)

File stat is write enabled logical.

Definition at line 94 of file osapi-file.h.

### 12.156.3 Enumeration Type Documentation

**12.156.3.1 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.

**12.156.3.2 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.

## 12.157 osal/src/os/inc/osapi-filesystem.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Data Structures

- struct os\_fsinfo\_t

*OSAL file system info.*

- struct [OS\\_statvfs\\_t](#)

## Macros

- #define [OS\\_CHK\\_ONLY](#) 0
- #define [OS\\_REPAIR](#) 1

## Functions

- [int32 OS\\_FileSysAddFixedMap \(osal\\_id\\_t \\*filesystems\\_id, const char \\*phys\\_path, const char \\*virt\\_path\)](#)  
*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 \(os\\_fsinfo\\_t \\*filesystems\\_info\)](#)  
*Returns information about the file system.*

### 12.157.1 Detailed Description

Declarations and prototypes for file systems

### 12.157.2 Macro Definition Documentation

#### 12.157.2.1 OS\_CHK\_ONLY #define OS\_CHK\_ONLY 0

Unused, API takes bool

Definition at line 31 of file osapi-filesystems.h.

#### 12.157.2.2 OS\_REPAIR #define OS\_REPAIR 1

Unused, API takes bool

Definition at line 32 of file osapi-filesystems.h.

## 12.158 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.*

#### 12.158.1 Detailed Description

Declarations and prototypes for heap functions

## 12.159 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`

- `#define OS_OBJECT_TYPE_OS_MODULE 0x0A`  
*Object module type.*
- `#define OS_OBJECT_TYPE_OS_FILESYS 0x0B`  
*Object file system type.*
- `#define OS_OBJECT_TYPE_OS_CONSOLE 0x0C`  
*Object console type.*
- `#define OS_OBJECT_TYPE_OS_CONDVAR 0x0D`  
*Object condition variable type.*
- `#define OS_OBJECT_TYPE_USER 0x10`  
*Object user type.*

## Functions

- `static unsigned long OS_ObjectIdToInteger (osal_id_t object_id)`  
*Obtain an integer value corresponding to an object ID.*
- `static osal_id_t OS_ObjectIdFromInteger (unsigned long value)`  
*Obtain an osal ID corresponding to an integer value.*
- `static bool OS_ObjectIdEqual (osal_id_t object_id1, osal_id_t object_id2)`  
*Check two OSAL object ID values for equality.*
- `static bool OS_ObjectIdDefined (osal_id_t object_id)`  
*Check if an object ID is defined.*
- `int32 OS_GetResourceName (osal_id_t object_id, char *buffer, size_t buffer_size)`  
*Obtain the name of an object given an arbitrary object ID.*
- `osal_objtype_t OS_IdentifyObject (osal_id_t object_id)`  
*Obtain the type of an object given an arbitrary object ID.*
- `int32 OS_ConvertToArrayIndex (osal_id_t object_id, osal_index_t *ArrayIndex)`  
*Converts an abstract ID into a number suitable for use as an array index.*
- `int32 OS_ObjectIdToArrayIndex (osal_objtype_t idtype, osal_id_t object_id, osal_index_t *ArrayIndex)`  
*Converts an abstract ID into a number suitable for use as an array index.*
- `void OS_ForEachObject (osal_id_t creator_id, OS_ArgCallback_t callback_ptr, void *callback_arg)`  
*call the supplied callback function for all valid object IDs*
- `void OS_ForEachObjectOfType (osal_objtype_t objtype, osal_id_t creator_id, OS_ArgCallback_t callback_ptr, void *callback_arg)`  
*call the supplied callback function for valid object IDs of a specific type*

### 12.159.1 Detailed Description

Declarations and prototypes for object IDs

### 12.159.2 Macro Definition Documentation

#### 12.159.2.1 OS\_OBJECT\_INDEX\_MASK `#define OS_OBJECT_INDEX_MASK 0xFFFF`

Object index mask.

Definition at line 32 of file osapi-idmap.h.

**12.159.2.2 OS\_OBJECT\_TYPE\_SHIFT** #define OS\_OBJECT\_TYPE\_SHIFT 16  
Object type shift.  
Definition at line 33 of file osapi-idmap.h.

## 12.160 osal/src/os/inc/osapi-macros.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "osconfig.h"
#include "common_types.h"
#include "osapi-printf.h"
```

### Macros

- #define **BUGREPORT**(...) OS\_printf(\_\_VA\_ARGS\_\_)
- #define **BUGCHECK**(cond, errcode)  
*Basic Bug-Checking macro.*
- #define **ARGCHECK**(cond, errcode)  
*Generic argument checking macro for non-critical values.*
- #define **LENGTHCHECK**(str, len, errcode) **ARGCHECK**(memchr(str, '\0', len), errcode)  
*String length limit check macro.*
- #define **BUGCHECK\_VOID**(cond) **BUGCHECK**(cond, )  
*Bug-Check macro for void functions.*

### 12.160.1 Detailed Description

Macro definitions that are used across all OSAL subsystems

### 12.160.2 Macro Definition Documentation

**12.160.2.1 ARGCHECK** #define ARGCHECK(  
    cond,  
    errcode )

**Value:**

```
if (! (cond)) \
{ \
    return errcode; \
}
```

Generic argument checking macro for non-critical values.

This macro checks a conditional that is expected to be true, and return a value if it evaluates false.

ARGCHECK can be used to check for out of range or other invalid argument conditions which may (validly) occur at runtime and do not necessarily indicate bugs in the application.

These argument checks are NOT considered fatal errors. The application continues to run normally. This does not report the error on the console.

As such, ARGCHECK actions are always compiled in - not selectable at compile-time.

**See also**

[BUGCHECK](#) for checking critical values that indicate bugs

Definition at line 131 of file osapi-macros.h.

```
12.160.2.2 BUGCHECK #define BUGCHECK(
    cond,
    errcode )
```

**Value:**

```
if (! (cond))
{
    BUGREPORT("\n**BUG** %s():%d:check \'%s\' FAILED --> %s\n\n", __func__, __LINE__, #cond, #errcode); \
    return errcode;
}
```

Basic Bug-Checking macro.

This macro checks a conditional, and if it is FALSE, then it generates a report - which may in turn contain additional actions.

BUGCHECK should only be used for conditions which are critical and must always be true. If such a condition is ever false then it indicates a bug in the application which must be resolved. It may or may not be possible to continue operation if a bugcheck fails.

**See also**

[ARGCHECK](#) for checking non-critical values

Definition at line 105 of file osapi-macros.h.

```
12.160.2.3 BUGCHECK_VOID #define BUGCHECK_VOID(
    cond ) BUGCHECK(cond, )
```

Bug-Check macro for void functions.

The basic BUGCHECK macro returns a value, which needs to be empty for functions that do not have a return value. In this case the second argument (errcode) is intentionally left blank.

Definition at line 155 of file osapi-macros.h.

```
12.160.2.4 BUGREPORT #define BUGREPORT(
    ... ) OS_printf(__VA_ARGS__)
```

Definition at line 88 of file osapi-macros.h.

```
12.160.2.5 LENGTHCHECK #define LENGTHCHECK(
    str,
    len,
    errcode ) ARGCHECK(memchr(str, '\0', len), errcode)
```

String length limit check macro.

This macro is a specialized version of ARGCHECK that confirms a string will fit into a buffer of the specified length, and return an error code if it will not.

**Note**

this uses ARGCHECK, thus treating a string too long as a normal runtime (i.e. non-bug) error condition with a typical error return to the caller.

Definition at line 146 of file osapi-macros.h.

## 12.161 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.*

### 12.161.1 Detailed Description

Declarations and prototypes for module subsystem

### 12.161.2 Macro Definition Documentation

#### 12.161.2.1 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.

**12.161.2.2 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.

**12.162 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.*

**12.162.1 Detailed Description**

Declarations and prototypes for mutexes

**12.163 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.*

### 12.163.1 Detailed Description

Declarations and prototypes for network subsystem

## 12.164 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.*

### 12.164.1 Detailed Description

Declarations and prototypes for printf/console output

## 12.165 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.*

### 12.165.1 Detailed Description

Declarations and prototypes for queue subsystem

## 12.166 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,  
`OS_STREAM_STATE_LISTENING` = 0x10 }
- For the `OS_SelectSingle()` function's in/out `StateFlags` parameter, the state(s) of the stream and the result of the select is a combination of one or more of these states.*

### Functions

- `int32 OS_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.*

### 12.166.1 Detailed Description

Declarations and prototypes for select abstraction

### 12.166.2 Enumeration Type Documentation

**12.166.2.1 OS\_StreamState\_t enum OS\_StreamState\_t**

For the [OS\\_SelectSingle\(\)](#) function's in/out StateFlags parameter, the state(s) of the stream and the result of the select is a combination of one or more of these states.

See also

[OS\\_SelectSingle\(\)](#)

Enumerator

OS_STREAM_STATE_BOUND	whether the stream is bound
OS_STREAM_STATE_CONNECTED	whether the stream is connected
OS_STREAM_STATE_READABLE	whether the stream is readable
OS_STREAM_STATE_WRITABLE	whether the stream is writable
OS_STREAM_STATE_LISTENING	whether the stream is listening

Definition at line 55 of file osapi-select.h.

**12.167 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.*

**12.167.1 Detailed Description**

Declarations and prototypes for shell abstraction

**12.168 osal/src/os/inc/osapi-sockets.h File Reference**

```
#include "osconfig.h"
#include "common_types.h"
```

Data Structures

- union [OS\\_SockAddrData\\_t](#)  
*Storage buffer for generic network address.*
- struct [OS\\_SockAddr\\_t](#)  
*Encapsulates a generic network address.*
- struct [OS\\_socket\\_prop\\_t](#)  
*Encapsulates socket properties.*

Macros

- [#define OS SOCKADDR MAX LEN 28](#)

## Enumerations

- enum `OS_SocketDomain_t` { `OS_SocketDomain_INVALID`, `OS_SocketDomain_INET`, `OS_SocketDomain_INET6`, `OS_SocketDomain_MAX` }  
*Socket domain.*
- enum `OS_SocketType_t` { `OS_SocketType_INVALID`, `OS_SocketType_DATAGRAM`, `OS_SocketType_STREAM`, `OS_SocketType_MAX` }  
*Socket type.*
- enum `OS_SocketShutdownMode_t` { `OS_SocketShutdownMode_NONE` = 0, `OS_SocketShutdownMode_SHUT_READ` = 1, `OS_SocketShutdownMode_SHUT_WRITE` = 2, `OS_SocketShutdownMode_SHUT_RDWR` = 3 }  
*Shutdown Mode.*

## Functions

- int32 `OS_SocketAddrInit` (`OS_SockAddr_t` \*`Addr`, `OS_SocketDomain_t` `Domain`)  
*Initialize a socket address structure to hold an address of the given family.*
- int32 `OS_SocketAddrToString` (`char` \*`buffer`, `size_t` `buflen`, const `OS_SockAddr_t` \*`Addr`)  
*Get a string representation of a network host address.*
- int32 `OS_SocketAddrFromString` (`OS_SockAddr_t` \*`Addr`, const `char` \*`string`)  
*Set a network host address from a string representation.*
- int32 `OS_SocketAddrGetPort` (`uint16` \*`PortNum`, const `OS_SockAddr_t` \*`Addr`)  
*Get the port number of a network address.*
- int32 `OS_SocketAddrSetPort` (`OS_SockAddr_t` \*`Addr`, `uint16` `PortNum`)  
*Set the port number of a network address.*
- int32 `OS_SocketOpen` (`osal_id_t` \*`sock_id`, `OS_SocketDomain_t` `Domain`, `OS_SocketType_t` `Type`)  
*Opens a socket.*
- int32 `OS_SocketBind` (`osal_id_t` `sock_id`, const `OS_SockAddr_t` \*`Addr`)  
*Binds a socket to a given local address and enter listening (server) mode.*
- int32 `OS_SocketListen` (`osal_id_t` `sock_id`)  
*Places the specified socket into a listening state.*
- int32 `OS_SocketBindAddress` (`osal_id_t` `sock_id`, const `OS_SockAddr_t` \*`Addr`)  
*Binds a socket to a given local address.*
- int32 `OS_SocketConnect` (`osal_id_t` `sock_id`, const `OS_SockAddr_t` \*`Addr`, int32 `timeout`)  
*Connects a socket to a given remote address.*
- int32 `OS_SocketShutdown` (`osal_id_t` `sock_id`, `OS_SocketShutdownMode_t` `Mode`)  
*Implement graceful shutdown of a stream socket.*
- int32 `OS_SocketAccept` (`osal_id_t` `sock_id`, `osal_id_t` \*`connsock_id`, `OS_SockAddr_t` \*`Addr`, int32 `timeout`)  
*Waits for and accept the next incoming connection on the given socket.*
- int32 `OS_SocketRecvFrom` (`osal_id_t` `sock_id`, void \*`buffer`, `size_t` `buflen`, `OS_SockAddr_t` \*`RemoteAddr`, int32 `timeout`)  
*Reads data from a message-oriented (datagram) socket.*
- int32 `OS_SocketSendTo` (`osal_id_t` `sock_id`, const void \*`buffer`, `size_t` `buflen`, const `OS_SockAddr_t` \*`RemoteAddr`)  
*Sends data to a message-oriented (datagram) socket.*
- int32 `OS_SocketGetIdByName` (`osal_id_t` \*`sock_id`, const `char` \*`sock_name`)  
*Gets an OSAL ID from a given name.*
- int32 `OS_SocketGetInfo` (`osal_id_t` `sock_id`, `OS_socket_prop_t` \*`sock_prop`)  
*Gets information about an OSAL Socket ID.*

### 12.168.1 Detailed Description

Declarations and prototypes for sockets abstraction

### 12.168.2 Macro Definition Documentation

#### 12.168.2.1 OS\_SOCKADDR\_MAX\_LEN `#define OS_SOCKADDR_MAX_LEN 28`

Definition at line 45 of file osapi-sockets.h.

### 12.168.3 Enumeration Type Documentation

#### 12.168.3.1 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.

#### 12.168.3.2 OS\_SocketShutdownMode\_t `enum OS_SocketShutdownMode_t`

Shutdown Mode.

Enumerator

OS_SocketShutdownMode_NONE	Reserved value, no effect.
OS_SocketShutdownMode_SHUT_READ	Disable future reading.
OS_SocketShutdownMode_SHUT_WRITE	Disable future writing.
OS_SocketShutdownMode_SHUT_RDWR	Disable future reading or writing.

Definition at line 79 of file osapi-sockets.h.

#### 12.168.3.3 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.

## 12.169 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.*

### 12.169.1 Detailed Description

Declarations and prototypes for task abstraction

### 12.169.2 Macro Definition Documentation

#### 12.169.2.1 OS\_FP\_ENABLED #define OS\_FP\_ENABLED 1

Floating point enabled state for a task.

Definition at line 35 of file osapi-task.h.

#### 12.169.2.2 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.

#### 12.169.2.3 OSAL\_PRIORITY\_C #define OSAL\_PRIORITY\_C( X ) ((osal\_priority\_t) {X})

Definition at line 46 of file osapi-task.h.

#### 12.169.2.4 OSAL\_STACKPTR\_C #define OSAL\_STACKPTR\_C( X ) ((osal\_stackptr\_t) {X})

Definition at line 53 of file osapi-task.h.

#### 12.169.2.5 OSAL\_TASK\_STACK\_ALLOCATE #define OSAL\_TASK\_STACK\_ALLOCATE OSAL\_STACKPTR\_C(NULL)

Definition at line 54 of file osapi-task.h.

### 12.169.3 Typedef Documentation

#### 12.169.3.1 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.

**12.169.3.2 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.

**12.169.3.3 osal\_task** `typedef void osal_task`

For task entry point.

Definition at line 68 of file osapi-task.h.

**12.169.4 Function Documentation****12.169.4.1 osal\_task()** `typedef osal_task (`  
    `(*) (void) osal_task_entry )`

For task entry point.

**12.170 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.*

**12.170.1 Detailed Description**

Declarations and prototypes for timebase abstraction

## 12.170.2 Typedef Documentation

### 12.170.2.1 OS\_TimerSync\_t `typedef uint32 (* OS_TimerSync_t) (osal_id_t timer_id)`

Timer sync.

Definition at line 34 of file osapi-timebase.h.

## 12.171 osal/src/os/inc/osapi-timer.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Data Structures

- struct `OS_timer_prop_t`

*Timer properties.*

### Typedefs

- `typedef void(* OS_TimerCallback_t) (osal_id_t timer_id)`

*Timer callback.*

### Functions

- `int32 OS_TimerCreate (osal_id_t *timer_id, const char *timer_name, uint32 *clock_accuracy, OS_TimerCallback_t callback_ptr)`  
*Create a timer object.*
- `int32 OS_TimerAdd (osal_id_t *timer_id, const char *timer_name, osal_id_t timebase_id, OS_ArgCallback_t callback_ptr, void *callback_arg)`  
*Add a timer object based on an existing TimeBase resource.*
- `int32 OS_TimerSet (osal_id_t timer_id, uint32 start_time, uint32 interval_time)`  
*Configures a periodic or one shot timer.*
- `int32 OS_TimerDelete (osal_id_t timer_id)`  
*Deletes a timer resource.*
- `int32 OS_TimerGetIdByName (osal_id_t *timer_id, const char *timer_name)`  
*Locate an existing timer resource by name.*
- `int32 OS_TimerGetInfo (osal_id_t timer_id, OS_timer_prop_t *timer_prop)`  
*Gets information about an existing timer.*

## 12.171.1 Detailed Description

Declarations and prototypes for timer abstraction (app callbacks)

## 12.171.2 Typedef Documentation

### 12.171.2.1 OS\_TimerCallback\_t `typedef void(* OS_TimerCallback_t) (osal_id_t timer_id)`

Timer callback.

Definition at line 34 of file osapi-timer.h.

## 12.172 osal/src/os/inc/osapi-version.h File Reference

```
#include "common_types.h"
```

### Macros

- #define OS\_BUILD\_NUMBER 229
  - Major version number.
- #define OS\_BUILD\_BASELINE "v6.0.0-rc4"
- #define OS\_MAJOR\_VERSION 5
  - Minor version number.
- #define OS\_MINOR\_VERSION 0
  - Revision version number. Value of 99 indicates a development version.
- #define OS\_MISSION\_REV 0xFF
  - Mission revision.
- #define OS\_STR\_HELPER(x) #x
  - Helper function to concatenate strings from integer.
- #define OS\_STR(x) OS\_STR\_HELPER(x)
  - Helper function to concatenate strings from integer.
- #define OS\_VERSION OS\_BUILD\_BASELINE "+dev" OS\_STR(OS\_BUILD\_NUMBER)
  - Development Build Version Number.
- #define OS\_VERSION\_CODENAME "Draco"
  - Version code name All modular components which are tested/validated together should share the same code name.
- #define OS\_VERSION\_STRING
  - Development Build Version String.
- #define OSAL\_API\_VERSION ((OS\_MAJOR\_VERSION \* 10000) + (OS\_MINOR\_VERSION \* 100) + OS\_REVISION)
  - Combines the revision components into a single value.

### Functions

- const char \* OS\_GetVersionString (void)
- const char \* OS\_GetVersionCodeName (void)
- void OS\_GetVersionNumber (uint8 VersionNumbers[4])
  - Obtain the OSAL numeric version number.
- uint32 OS\_GetBuildNumber (void)
  - Obtain the OSAL library numeric build number.

### 12.172.1 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.

### 12.172.2 Macro Definition Documentation

**12.172.2.1 OS\_BUILD\_BASELINE** #define OS\_BUILD\_BASELINE "v6.0.0-rc4"  
Definition at line 38 of file osapi-version.h.

**12.172.2.2 OS\_BUILD\_NUMBER** #define OS\_BUILD\_NUMBER 229  
Definition at line 37 of file osapi-version.h.

**12.172.2.3 OS\_MAJOR\_VERSION** #define OS\_MAJOR\_VERSION 5  
Major version number.  
Definition at line 43 of file osapi-version.h.

**12.172.2.4 OS\_MINOR\_VERSION** #define OS\_MINOR\_VERSION 0  
Minor version number.  
Definition at line 44 of file osapi-version.h.

**12.172.2.5 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.

**12.172.2.6 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.

**12.172.2.7 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.

**12.172.2.8 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.

**12.172.2.9 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.

**12.172.2.10 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.

**12.172.2.11 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.

**12.172.2.12 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.

### 12.172.3 Function Documentation

**12.172.3.1 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

**12.172.3.2 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.

**12.172.3.3 OS\_GetVersionNumber()** `void OS_GetVersionNumber (`  
    `uint8 VersionNumbers[4] )`

Obtain the OSAL numeric version number.

This retrieves the numeric OSAL version identifier as an array of 4 uint8 values.

The array of numeric values is in order of precedence: [0] = Major Number [1] = Minor Number [2] = Revision Number [3] = Mission Revision

The "Mission Revision" (last output) also indicates whether this is an official release, a patched release, or a development version. 0 indicates an official release 1-254 local patch level (reserved for mission use) 255 indicates a development build

#### Parameters

out	<i>VersionNumbers</i>	A fixed-size array to be filled with the version numbers
-----	-----------------------	--

**12.172.3.4 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.

## 12.173 osal/src/os/inc/osapi.h File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "common_types.h"
#include "osapi-version.h"
#include "osconfig.h"
#include "osapi-binsem.h"
#include "osapi-clock.h"
#include "osapi-common.h"
#include "osapi-condvar.h"
#include "osapi-constants.h"
#include "osapi-countsem.h"
#include "osapi-dir.h"
#include "osapi-error.h"
#include "osapi-file.h"
#include "osapi-fs.h"
#include "osapi-heap.h"
#include "osapi-macros.h"
#include "osapi-idmap.h"
#include "osapi-module.h"
#include "osapi-mutex.h"
#include "osapi-network.h"
#include "osapi-printf.h"
#include "osapi-queue.h"
#include "osapi-select.h"
#include "osapi-shell.h"
#include "osapi-sockets.h"
```

```
#include "osapi-task.h"
#include "osapi-timebase.h"
#include "osapi-timer.h"
#include "osapi-bsp.h"
```

### 12.173.1 Detailed Description

Purpose: Contains functions prototype definitions and variables declarations for the OS Abstraction Layer, Core OS module

## 12.174 psp/fsw/inc/cfe\_psp.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

### Macros

- #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.*

## 12.174.1 Macro Definition Documentation

### 12.174.1.1 BUFF\_SIZE #define BUFF\_SIZE 256

Definition at line 62 of file cfe\_psp.h.

**12.174.1.2 CFE\_PSP\_MEM\_ANY** #define CFE\_PSP\_MEM\_ANY 3  
Definition at line 72 of file cfe\_psp.h.

**12.174.1.3 CFE\_PSP\_MEM\_ATTR\_READ** #define CFE\_PSP\_MEM\_ATTR\_READ 0x02  
Definition at line 79 of file cfe\_psp.h.

**12.174.1.4 CFE\_PSP\_MEM\_ATTR\_READWRITE** #define CFE\_PSP\_MEM\_ATTR\_READWRITE 0x03  
Definition at line 80 of file cfe\_psp.h.

**12.174.1.5 CFE\_PSP\_MEM\_ATTR\_WRITE** #define CFE\_PSP\_MEM\_ATTR\_WRITE 0x01  
Definition at line 78 of file cfe\_psp.h.

**12.174.1.6 CFE\_PSP\_MEM\_EEPROM** #define CFE\_PSP\_MEM\_EEPROM 2  
Definition at line 71 of file cfe\_psp.h.

**12.174.1.7 CFE\_PSP\_MEM\_INVALID** #define CFE\_PSP\_MEM\_INVALID 4  
Definition at line 73 of file cfe\_psp.h.

**12.174.1.8 CFE\_PSP\_MEM\_RAM** #define CFE\_PSP\_MEM\_RAM 1  
Definition at line 70 of file cfe\_psp.h.

**12.174.1.9 CFE\_PSP\_MEM\_SIZE\_BYTE** #define CFE\_PSP\_MEM\_SIZE\_BYTE 0x01  
Definition at line 85 of file cfe\_psp.h.

**12.174.1.10 CFE\_PSP\_MEM\_SIZE\_DWORD** #define CFE\_PSP\_MEM\_SIZE\_DWORD 0x04  
Definition at line 87 of file cfe\_psp.h.

**12.174.1.11 CFE\_PSP\_MEM\_SIZE\_WORD** #define CFE\_PSP\_MEM\_SIZE\_WORD 0x02  
Definition at line 86 of file cfe\_psp.h.

**12.174.1.12 CFE\_PSP\_PANIC\_CORE\_APP** #define CFE\_PSP\_PANIC\_CORE\_APP 6  
Definition at line 56 of file cfe\_psp.h.

**12.174.1.13 CFE\_PSP\_PANIC\_GENERAL\_FAILURE** #define CFE\_PSP\_PANIC\_GENERAL\_FAILURE 7  
Definition at line 57 of file cfe\_psp.h.

**12.174.1.14 CFE\_PSP\_PANIC\_MEMORY\_ALLOC** #define CFE\_PSP\_PANIC\_MEMORY\_ALLOC 3  
Definition at line 53 of file cfe\_psp.h.

**12.174.1.15 CFE\_PSP\_PANIC\_NONVOL\_DISK** #define CFE\_PSP\_PANIC\_NONVOL\_DISK 4  
Definition at line 54 of file cfe\_psp.h.

**12.174.1.16 CFE\_PSP\_PANIC\_STARTUP** #define CFE\_PSP\_PANIC\_STARTUP 1  
Definition at line 51 of file cfe\_psp.h.

**12.174.1.17 CFE\_PSP\_PANIC\_STARTUP\_SEM** #define CFE\_PSP\_PANIC\_STARTUP\_SEM 5  
Definition at line 55 of file cfe\_psp.h.

**12.174.1.18 CFE\_PSP\_PANIC\_VOLATILE\_DISK** #define CFE\_PSP\_PANIC\_VOLATILE\_DISK 2  
Definition at line 52 of file cfe\_psp.h.

**12.174.1.19 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 122 of file cfe\_psp.h.

**12.174.1.20 CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION** #define CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION 6  
Reset was caused by a Processor Exception.  
Definition at line 116 of file cfe\_psp.h.

**12.174.1.21 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 110 of file cfe\_psp.h.

**12.174.1.22 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 112 of file cfe\_psp.h.

**12.174.1.23 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 120 of file cfe\_psp.h.

**12.174.1.24 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 124 of file cfe\_psp.h.

**12.174.1.25 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 106 of file cfe\_psp.h.

**12.174.1.26 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 108 of file cfe\_psp.h.

**12.174.1.27 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 114 of file cfe\_psp.h.

**12.174.1.28 CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET** #define CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET 7  
Reset was caused in an unknown manner.  
Definition at line 118 of file cfe\_psp.h.

**12.174.1.29 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 96 of file cfe\_psp.h.

**12.174.1.30 CFE\_PSP\_RST\_TYPE\_POWERON** #define CFE\_PSP\_RST\_TYPE\_POWERON 2  
All memory has been cleared  
Definition at line 95 of file cfe\_psp.h.

**12.174.1.31 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 94 of file cfe\_psp.h.

**12.174.1.32 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 132 of file cfe\_psp.h.

**12.174.1.33 SIZE\_BYTE** #define SIZE\_BYTE 1  
Definition at line 63 of file cfe\_psp.h.

**12.174.1.34 SIZE\_HALF** #define SIZE\_HALF 2  
Definition at line 64 of file cfe\_psp.h.

**12.174.1.35 SIZE\_WORD** #define SIZE\_WORD 3  
Definition at line 65 of file cfe\_psp.h.

## 12.174.2 Function Documentation

**12.174.2.1 CFE\_PSP\_AttachExceptions()** `void CFE_PSP_AttachExceptions ( void )`

**12.174.2.2 CFE\_PSP\_Decompress()** `int32 CFE_PSP_Decompress ( char * srcFileName, char * dstFileName )`

**12.174.2.3 CFE\_PSP\_EepromPowerDown()** `int32 CFE_PSP_EepromPowerDown ( uint32 Bank )`

**12.174.2.4 CFE\_PSP\_EepromPowerUp()** `int32 CFE_PSP_EepromPowerUp ( uint32 Bank )`

**12.174.2.5 CFE\_PSP\_EepromWrite16()** `int32 CFE_PSP_EepromWrite16 ( cpuaddr MemoryAddress, uint16 uint16Value )`

**12.174.2.6 CFE\_PSP\_EepromWrite32()** `int32 CFE_PSP_EepromWrite32 ( cpuaddr MemoryAddress, uint32 uint32Value )`

**12.174.2.7 CFE\_PSP\_EepromWrite8()** `int32 CFE_PSP_EepromWrite8 ( cpuaddr MemoryAddress, uint8 ByteValue )`

**12.174.2.8 CFE\_PSP\_EepromWriteDisable()** `int32 CFE_PSP_EepromWriteDisable ( uint32 Bank )`

**12.174.2.9 CFE\_PSP\_EepromWriteEnable()** `int32 CFE_PSP_EepromWriteEnable ( uint32 Bank )`

**12.174.2.10 CFE\_PSP\_Exception\_CopyContext()** `int32 CFE_PSP_Exception_CopyContext ( uint32 ContextLogId, void * ContextBuf, uint32 ContextSize )`

**12.174.2.11 CFE\_PSP\_Exception\_GetCount()** `uint32 CFE_PSP_Exception_GetCount ( void )`

```
12.174.2.12 CFE_PSP_Exception_GetSummary() int32 CFE_PSP_Exception_GetSummary (
    uint32 * ContextLogId,
    osal_id_t * TaskId,
    char * ReasonBuf,
    uint32 ReasonSize )
```

```
12.174.2.13 CFE_PSP_FlushCaches() void CFE_PSP_FlushCaches (
    uint32 type,
    void * address,
    uint32 size )
```

```
12.174.2.14 CFE_PSP_Get_Dec() uint32 CFE_PSP_Get_Dec (
    void )
```

```
12.174.2.15 CFE_PSP_Get_Timebase() void CFE_PSP_Get_Timebase (
    uint32 * Tbu,
    uint32 * Tbl )
```

Sample/Read a monotonic platform clock without normalization.

This is defined as a free-running, monotonically-increasing tick counter. The epoch is not defined, but typically is the system boot time, and the value increases indefinitely as the system runs. The tick period/rate is also not defined.

Rollover events - where the range of representable values is exceeded - are theoretically possible, but would take many years of continuous uptime to occur (typically hundreds of years, if not thousands). System designers should ensure that the actual tick rate and resulting timebase range is sufficiently large to ensure that rollover is not a concern.

#### Note

This is a "raw" value from the underlying platform with minimal/no conversions or normalization applied. Neither the epoch nor the resolution of this tick counter is specified, and it may vary from platform to platform. Use the [CFE\\_PSP\\_GetTime\(\)](#) function to sample the timebase and also convert the units into a normalized/more consistent form.

#### See also

[CFE\\_PSP\\_GetTime\(\)](#)

#### Parameters

out	<i>Tbu</i>	Buffer to hold the upper 32 bits of a 64-bit tick counter
out	<i>Tbl</i>	Buffer to hold the lower 32 bits of a 64-bit tick counter

```
12.174.2.16 CFE_PSP_Get_Timer_Tick() uint32 CFE_PSP_Get_Timer_Tick (
    void )
```

```
12.174.2.17 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

**12.174.2.18 CFE\_PSP\_GetCDSSize()** `int32 CFE_PSP_GetCDSSize ( uint32 * SizeOfCDS )`

**12.174.2.19 CFE\_PSP\_GetCFETextSegmentInfo()** `int32 CFE_PSP_GetCFETextSegmentInfo ( cpuaddr * PtrToCFEsegment, uint32 * SizeOfCFEsegment )`

Referenced by CS\_InitSegments().

**12.174.2.20 CFE\_PSP\_GetKernelTextSegmentInfo()** `int32 CFE_PSP_GetKernelTextSegmentInfo ( cpuaddr * PtrToKernelSegment, uint32 * SizeOfKernelSegment )`

Referenced by CS\_InitSegments().

**12.174.2.21 CFE\_PSP\_GetProcessorId()** `uint32 CFE_PSP_GetProcessorId ( void )`

**12.174.2.22 CFE\_PSP\_GetProcessorName()** `const char* CFE_PSP_GetProcessorName ( void )`

**12.174.2.23 CFE\_PSP\_GetResetArea()** `int32 CFE_PSP_GetResetArea ( cpuaddr * PtrToResetArea, uint32 * SizeOfResetArea )`

**12.174.2.24 CFE\_PSP\_GetRestartType()** `uint32 CFE_PSP_GetRestartType ( uint32 * restartSubType )`

**12.174.2.25 CFE\_PSP\_GetSpacecraftId()** `uint32 CFE_PSP_GetSpacecraftId ( void )`

**12.174.2.26 CFE\_PSP\_GetTime()** `void CFE_PSP_GetTime ( OS_time_t * LocalTime )`

Sample/Read a monotonic platform clock with normalization.

Outputs an `OS_time_t` value indicating the time elapsed since an epoch. The epoch is not defined, but typically represents the system boot time. The value increases continuously over time and cannot be reset by software.

This is similar to the [CFE\\_PSP\\_Get\\_Timebase\(\)](#), but additionally it normalizes the output value to an [OS\\_time\\_t](#), thereby providing consistent units to the calling application. Any OSAL-provided routine accepts [OS\\_time\\_t](#) inputs may be used to convert this value into other standardized time units.

#### Note

This should refer to the same time domain as [CFE\\_PSP\\_Get\\_Timebase\(\)](#), the primary difference being the format and units of the output value.

#### See also

[CFE\\_PSP\\_Get\\_Timebase\(\)](#)

#### Parameters

<code>out</code>	<code>LocalTime</code>	Value of PSP tick counter as <a href="#">OS_time_t</a>
------------------	------------------------	--

**12.174.2.27 CFE\_PSP\_GetTimerLow32Rollover()** `uint32` CFE\_PSP\_GetTimerLow32Rollover (   
 `void` )

**12.174.2.28 CFE\_PSP\_GetTimerTicksPerSecond()** `uint32` CFE\_PSP\_GetTimerTicksPerSecond (   
 `void` )

**12.174.2.29 CFE\_PSP.GetUserReservedArea()** `int32` CFE\_PSP.GetUserReservedArea (   
 `cpuaddr` \* *PtrToUserArea*,   
 `uint32` \* *SizeOfUserArea* )

**12.174.2.30 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.

**12.174.2.31 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	<i>VersionNumbers</i>	A fixed-size array to be filled with the version numbers
-----	-----------------------	--

**12.174.2.32 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.

**12.174.2.33 CFE\_PSP\_GetVolatileDiskMem()** `int32 CFE_PSP_GetVolatileDiskMem ( cpuaddr * PtrToVolDisk, uint32 * SizeOfVolDisk )`

**12.174.2.34 CFE\_PSP\_InitSSR()** `int32 CFE_PSP_InitSSR ( uint32 bus, uint32 device, char * DeviceName )`

**12.174.2.35 CFE\_PSP\_Main()** `void CFE_PSP_Main ( void )`

**12.174.2.36 CFE\_PSP\_MemCpy()** `int32 CFE_PSP_MemCpy ( void * dest, const void * src, uint32 n )`

**12.174.2.37 CFE\_PSP\_MemRangeGet()** `int32 CFE_PSP_MemRangeGet ( uint32 RangeNum, uint32 * MemoryType, cpuaddr * StartAddr, size_t * Size, size_t * WordSize, uint32 * Attributes )`

**12.174.2.38 CFE\_PSP\_MemRanges()** `uint32 CFE_PSP_MemRanges ( void )`

**12.174.2.39 CFE\_PSP\_MemRangeSet()** `int32 CFE_PSP_MemRangeSet (`  
    `uint32 RangeNum,`  
    `uint32 MemoryType,`  
    `cpuaddr StartAddr,`  
    `size_t Size,`  
    `size_t WordSize,`  
    `uint32 Attributes )`

**12.174.2.40 CFE\_PSP\_MemRead16()** `int32 CFE_PSP_MemRead16 (`  
    `cpuaddr MemoryAddress,`  
    `uint16 * uint16Value )`

**12.174.2.41 CFE\_PSP\_MemRead32()** `int32 CFE_PSP_MemRead32 (`  
    `cpuaddr MemoryAddress,`  
    `uint32 * uint32Value )`

**12.174.2.42 CFE\_PSP\_MemRead8()** `int32 CFE_PSP_MemRead8 (`  
    `cpuaddr MemoryAddress,`  
    `uint8 * ByteValue )`

**12.174.2.43 CFE\_PSP\_MemSet()** `int32 CFE_PSP_MemSet (`  
    `void * dest,`  
    `uint8 value,`  
    `uint32 n )`

**12.174.2.44 CFE\_PSP\_MemValidateRange()** `int32 CFE_PSP_MemValidateRange (`  
    `cpuaddr Address,`  
    `size_t Size,`  
    `uint32 MemoryType )`

Referenced by `CS_OneShotCmd()`, `CS_ValidateEepromChecksumDefinitionTable()`, and `CS_ValidateMemoryChecksumDefinitionTable()`.

**12.174.2.45 CFE\_PSP\_MemWrite16()** `int32 CFE_PSP_MemWrite16 (`  
    `cpuaddr MemoryAddress,`  
    `uint16 uint16Value )`

**12.174.2.46 CFE\_PSP\_MemWrite32()** `int32 CFE_PSP_MemWrite32 (`  
    `cpuaddr MemoryAddress,`  
    `uint32 uint32Value )`

**12.174.2.47 CFE\_PSP\_MemWrite8()** `int32 CFE_PSP_MemWrite8 (`  
    `cpuaddr MemoryAddress,`  
    `uint8 ByteValue )`

**12.174.2.48 CFE\_PSP\_Panic()** void CFE\_PSP\_Panic (   
   int32 ErrorCode )

**12.174.2.49 CFE\_PSP\_PortRead16()** int32 CFE\_PSP\_PortRead16 (   
   cpuaddr PortAddress,   
   uint16 \* uint16Value )

**12.174.2.50 CFE\_PSP\_PortRead32()** int32 CFE\_PSP\_PortRead32 (   
   cpuaddr PortAddress,   
   uint32 \* uint32Value )

**12.174.2.51 CFE\_PSP\_PortRead8()** int32 CFE\_PSP\_PortRead8 (   
   cpuaddr PortAddress,   
   uint8 \* ByteValue )

**12.174.2.52 CFE\_PSP\_PortWrite16()** int32 CFE\_PSP\_PortWrite16 (   
   cpuaddr PortAddress,   
   uint16 uint16Value )

**12.174.2.53 CFE\_PSP\_PortWrite32()** int32 CFE\_PSP\_PortWrite32 (   
   cpuaddr PortAddress,   
   uint32 uint32Value )

**12.174.2.54 CFE\_PSP\_PortWrite8()** int32 CFE\_PSP\_PortWrite8 (   
   cpuaddr PortAddress,   
   uint8 ByteValue )

**12.174.2.55 CFE\_PSP\_ReadFromCDS()** int32 CFE\_PSP\_ReadFromCDS (   
   void \* PtrToDataToRead,   
   uint32 CDSOffset,   
   uint32 NumBytes )

**12.174.2.56 CFE\_PSP\_Restart()** void CFE\_PSP\_Restart (   
   uint32 resetType )

**12.174.2.57 CFE\_PSP\_SetDefaultExceptionEnvironment()** void CFE\_PSP\_SetDefaultExceptionEnvironment (   
   void )

**12.174.2.58 CFE\_PSP\_WatchdogDisable()** void CFE\_PSP\_WatchdogDisable (   
   void )

**12.174.2.59 CFE\_PSP\_WatchdogEnable()** void CFE\_PSP\_WatchdogEnable ( void )

**12.174.2.60 CFE\_PSP\_WatchdogGet()** uint32 CFE\_PSP\_WatchdogGet ( void )

**12.174.2.61 CFE\_PSP\_WatchdogInit()** void CFE\_PSP\_WatchdogInit ( void )

**12.174.2.62 CFE\_PSP\_WatchdogService()** void CFE\_PSP\_WatchdogService ( void )

**12.174.2.63 CFE\_PSP\_WatchdogSet()** void CFE\_PSP\_WatchdogSet ( uint32 WatchdogValue )

**12.174.2.64 CFE\_PSP\_WriteToCDS()** int32 CFE\_PSP\_WriteToCDS ( const void \* PtrToDataToWrite, uint32 CDSOffset, uint32 NumBytes )

## 12.175 psp/fsw/inc/cfe\_psp\_error.h File Reference

cFE PSP Error header

```
#include "common_types.h"
```

### Macros

- #define CFE\_PSP\_STATUS\_C(X) ((CFE\_PSP\_Status\_t)(X))  
*PSP Status macro for literal.*
- #define CFE\_PSP\_STATUS\_STRING\_LENGTH 12  
*PSP Status converted to string length limit.*
- #define CFE\_PSP\_SUCCESS (CFE\_PSP\_STATUS\_C(0))
- #define CFE\_PSP\_ERROR (CFE\_PSP\_STATUS\_C(-1))
- #define CFE\_PSP\_INVALID\_POINTER (CFE\_PSP\_STATUS\_C(-2))
- #define CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED (CFE\_PSP\_STATUS\_C(-3))
- #define CFE\_PSP\_ERROR\_TIMEOUT (CFE\_PSP\_STATUS\_C(-4))
- #define CFE\_PSP\_INVALID\_INT\_NUM (CFE\_PSP\_STATUS\_C(-5))
- #define CFE\_PSP\_INVALID\_MEM\_ADDR (CFE\_PSP\_STATUS\_C(-21))
- #define CFE\_PSP\_INVALID\_MEM\_TYPE (CFE\_PSP\_STATUS\_C(-22))
- #define CFE\_PSP\_INVALID\_MEM\_RANGE (CFE\_PSP\_STATUS\_C(-23))
- #define CFE\_PSP\_INVALID\_MEM\_WORDSIZE (CFE\_PSP\_STATUS\_C(-24))
- #define CFE\_PSP\_INVALID\_MEM\_SIZE (CFE\_PSP\_STATUS\_C(-25))
- #define CFE\_PSP\_INVALID\_MEM\_ATTR (CFE\_PSP\_STATUS\_C(-26))
- #define CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED (CFE\_PSP\_STATUS\_C(-27))
- #define CFE\_PSP\_INVALID\_MODULE\_NAME (CFE\_PSP\_STATUS\_C(-28))
- #define CFE\_PSP\_INVALID\_MODULE\_ID (CFE\_PSP\_STATUS\_C(-29))
- #define CFE\_PSP\_NO\_EXCEPTION\_DATA (CFE\_PSP\_STATUS\_C(-30))

## Typedefs

- `typedef int32 CFE_PSP_Status_t`  
*PSP Status type for readability and potentially type safety.*
- `typedef char CFE_PSP_StatusString_t[CFE_PSP_STATUS_STRING_LENGTH]`  
*For the [CFE\\_PSP\\_StatusToString\(\)](#) function, to ensure everyone is making an array of the same length.*

## Functions

- `char * CFE_PSP_StatusToString (CFE_PSP_Status_t status, CFE_PSP_StatusString_t *status_string)`  
*Convert status to a string.*

### 12.175.1 Detailed Description

cFE PSP Error header

### 12.175.2 Macro Definition Documentation

**12.175.2.1 CFE\_PSP\_ERROR** `#define CFE_PSP_ERROR (CFE_PSP_STATUS_C(-1))`  
Definition at line 66 of file cfe\_psp\_error.h.

**12.175.2.2 CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED** `#define CFE_PSP_ERROR_ADDRESS_MISALIGNED (CFE_PSP_STATUS_C(-3))`  
Definition at line 68 of file cfe\_psp\_error.h.

**12.175.2.3 CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED** `#define CFE_PSP_ERROR_NOT_IMPLEMENTED (CFE_PSP_STATUS_C(-27))`  
Definition at line 77 of file cfe\_psp\_error.h.

**12.175.2.4 CFE\_PSP\_ERROR\_TIMEOUT** `#define CFE_PSP_ERROR_TIMEOUT (CFE_PSP_STATUS_C(-4))`  
Definition at line 69 of file cfe\_psp\_error.h.

**12.175.2.5 CFE\_PSP\_INVALID\_INT\_NUM** `#define CFE_PSP_INVALID_INT_NUM (CFE_PSP_STATUS_C(-5))`  
Definition at line 70 of file cfe\_psp\_error.h.

**12.175.2.6 CFE\_PSP\_INVALID\_MEM\_ADDR** `#define CFE_PSP_INVALID_MEM_ADDR (CFE_PSP_STATUS_C(-21))`  
Definition at line 71 of file cfe\_psp\_error.h.

**12.175.2.7 CFE\_PSP\_INVALID\_MEM\_ATTR** `#define CFE_PSP_INVALID_MEM_ATTR (CFE_PSP_STATUS_C(-26))`  
Definition at line 76 of file cfe\_psp\_error.h.

**12.175.2.8 CFE\_PSP\_INVALID\_MEM\_RANGE** `#define CFE_PSP_INVALID_MEM_RANGE (CFE_PSP_STATUS_C(-23))`  
Definition at line 73 of file cfe\_psp\_error.h.

**12.175.2.9 CFE\_PSP\_INVALID\_MEM\_SIZE** #define CFE\_PSP\_INVALID\_MEM\_SIZE (CFE\_PSP\_STATUS\_C(-25))  
Definition at line 75 of file cfe\_psp\_error.h.

**12.175.2.10 CFE\_PSP\_INVALID\_MEM\_TYPE** #define CFE\_PSP\_INVALID\_MEM\_TYPE (CFE\_PSP\_STATUS\_C(-22))  
Definition at line 72 of file cfe\_psp\_error.h.

**12.175.2.11 CFE\_PSP\_INVALID\_MEM\_WORDSIZE** #define CFE\_PSP\_INVALID\_MEM\_WORDSIZE (CFE\_PSP\_STATUS\_C(-24))  
Definition at line 74 of file cfe\_psp\_error.h.

**12.175.2.12 CFE\_PSP\_INVALID\_MODULE\_ID** #define CFE\_PSP\_INVALID\_MODULE\_ID (CFE\_PSP\_STATUS\_C(-29))  
Definition at line 79 of file cfe\_psp\_error.h.

**12.175.2.13 CFE\_PSP\_INVALID\_MODULE\_NAME** #define CFE\_PSP\_INVALID\_MODULE\_NAME (CFE\_PSP\_STATUS\_C(-28))  
Definition at line 78 of file cfe\_psp\_error.h.

**12.175.2.14 CFE\_PSP\_INVALID\_POINTER** #define CFE\_PSP\_INVALID\_POINTER (CFE\_PSP\_STATUS\_C(-2))  
Definition at line 67 of file cfe\_psp\_error.h.

**12.175.2.15 CFE\_PSP\_NO\_EXCEPTION\_DATA** #define CFE\_PSP\_NO\_EXCEPTION\_DATA (CFE\_PSP\_STATUS\_C(-30))  
Definition at line 80 of file cfe\_psp\_error.h.

**12.175.2.16 CFE\_PSP\_STATUS\_C** #define CFE\_PSP\_STATUS\_C(  
    X ) ((CFE\_PSP\_Status\_t)(X))

PSP Status macro for literal.

Definition at line 36 of file cfe\_psp\_error.h.

**12.175.2.17 CFE\_PSP\_STATUS\_STRING\_LENGTH** #define CFE\_PSP\_STATUS\_STRING\_LENGTH 12

PSP Status converted to string length limit.

Used for sizing CFE\_PSP\_StatusString\_t intended for use in printing CFE\_PSP\_Status\_t values Sized for Id (LONG←\_MIN) including NULL

Definition at line 44 of file cfe\_psp\_error.h.

**12.175.2.18 CFE\_PSP\_SUCCESS** #define CFE\_PSP\_SUCCESS (CFE\_PSP\_STATUS\_C(0))  
Definition at line 65 of file cfe\_psp\_error.h.

## 12.175.3 Typedef Documentation

**12.175.3.1 CFE\_PSP\_Status\_t** typedef int32 CFE\_PSP\_Status\_t

PSP Status type for readability and potentially type safety.

Definition at line 31 of file cfe\_psp\_error.h.

**12.175.3.2 CFE\_PSP\_StatusString\_t** `typedef char CFE_PSP_StatusString_t[CFE_PSP_STATUS_STRING_LENGTH]`  
For the `CFE_PSP_StatusToString()` function, to ensure everyone is making an array of the same length.  
Definition at line 50 of file `cfe_psp_error.h`.

## 12.175.4 Function Documentation

**12.175.4.1 CFE\_PSP\_StatusToString()** `char* CFE_PSP_StatusToString (`  
`CFE_PSP_Status_t status,`  
`CFE_PSP_StatusString_t * status_string )`

Convert status to a string.

### Parameters

in	<i>status</i>	Status value to convert
out	<i>status_string</i>	Buffer to store status converted to string

### Returns

Passed in string pointer



## Index

\_EXTENSION\_  
    common\_types.h, 1317

accuracy  
    OS\_timebase\_prop\_t, 697  
    OS\_timer\_prop\_t, 697

ActiveBuffer  
    CFE\_TBL\_HousekeepingTlm\_Payload, 619

ActiveBufferAddr  
    CFE\_TBL\_TblRegPacket\_Payload, 629

ActiveTableFlag  
    CFE\_TBL\_DumpCmd\_Payload, 613  
    CFE\_TBL\_ValidateCmd\_Payload, 632

ActualLength  
    OS\_SockAddr\_t, 692

addr  
    OS\_module\_prop\_t, 690

AddrData  
    OS\_SockAddr\_t, 692

Address  
    CS\_GetEntryIDCmd\_Payload\_t, 667  
    CS\_OneShotCmd\_Payload\_t, 675  
    OS\_static\_symbol\_record\_t, 694

AddressesAreValid  
    CFE\_ES\_AppInfo, 527

AlignPtr  
    OS\_SockAddrData\_t, 693

AlignU32  
    OS\_SockAddrData\_t, 693

AppCSErrCounter  
    CS\_HkPacket\_Payload\_t, 669

AppCSState  
    CS\_HkPacket\_Payload\_t, 669

AppData  
    CFE\_EVS\_HousekeepingTlm\_Payload, 574

AppDataFilename  
    CFE\_EVS\_AppDataCmd\_Payload, 564

AppEnableStatus  
    CFE\_EVS\_AppTlmData, 570

AppEntryPoint  
    CFE\_ES\_StartAppCmd\_Payload, 559

AppFileName  
    CFE\_ES\_AppReloadCmd\_Payload, 531  
    CFE\_ES\_StartAppCmd\_Payload, 559

AppID  
    CFE\_EVS\_AppTlmData, 570

Appld  
    CFE\_ES\_TaskInfo, 563  
    CFE\_SB\_PipeInfoEntry, 597

AppInfo  
    CFE\_ES\_OneAppTlm\_Payload, 549

Application  
    CFE\_ES\_AppNameCmd\_Payload, 531  
    CFE\_ES\_AppReloadCmd\_Payload, 532  
    CFE\_ES\_SendMemPoolStatsCmd\_Payload, 554  
    CFE\_ES\_StartAppCmd\_Payload, 559

ApplicationID  
    CFE\_FS\_Header, 586

AppMessageSentCounter  
    CFE\_EVS\_AppTlmData, 570

AppMessageSquelchedCounter  
    CFE\_EVS\_AppTlmData, 570

AppName  
    CFE\_ES\_TaskInfo, 563  
    CFE\_EVS\_AppNameBitMaskCmd\_Payload, 565  
    CFE\_EVS\_AppNameCmd\_Payload, 566  
    CFE\_EVS\_AppNameEventIDCmd\_Payload, 568  
    CFE\_EVS\_AppNameEventIDMaskCmd\_Payload, 569  
    CFE\_EVS\_PacketID, 579  
    CFE\_SB\_PipeInfoEntry, 597  
    CFE\_SB\_RoutingFileEntry, 600

AppResTablesTblPtr  
    CS\_AppData\_t, 656

apps/cs/docs/dox\_src/cfs\_cs.dox, 698

apps/cs/fsw/inc/cs\_events.h, 698

apps/cs/fsw/inc/cs\_mission\_cfg.h, 705

apps/cs/fsw/inc/cs\_msg.h, 705

apps/cs/fsw/inc/cs\_msgdefs.h, 706

apps/cs/fsw/inc/cs\_msghids.h, 710

apps/cs/fsw/inc/cs\_perfids.h, 710

apps/cs/fsw/inc/cs\_platform\_cfg.h, 710

apps/cs/fsw/inc/cs\_tbldefs.h, 712

apps/cs/fsw/src/cs\_app.c, 724

apps/cs/fsw/src/cs\_app.h, 736

apps/cs/fsw/src/cs\_app\_cmds.c, 749

apps/cs/fsw/src/cs\_app\_cmds.h, 755

apps/cs/fsw/src/cs\_cmds.c, 761

apps/cs/fsw/src/cs\_cmds.h, 776

apps/cs/fsw/src/cs\_compute.c, 791

apps/cs/fsw/src/cs\_compute.h, 799

apps/cs/fsw/src/cs\_eeprom\_cmds.c, 807

apps/cs/fsw/src/cs\_eeprom\_cmds.h, 814

apps/cs/fsw/src/cs\_init.c, 820

apps/cs/fsw/src/cs\_init.h, 824

apps/cs/fsw/src/cs\_memory\_cmds.c, 828

apps/cs/fsw/src/cs\_memory\_cmds.h, 835

apps/cs/fsw/src/cs\_table\_cmds.c, 842

apps/cs/fsw/src/cs\_table\_cmds.h, 848

apps/cs/fsw/src/cs\_table\_processing.c, 854

apps/cs/fsw/src/cs\_utils.c, 865

apps/cs/fsw/src/cs\_utils.h, 884

apps/cs/fsw/src/cs\_verify.h, 904  
 apps/cs/fsw/src/cs\_version.h, 905  
 apps/cs/fsw/tables/cs\_apptbl.c, 905  
 apps/cs/fsw/tables/cs\_eepromtbl.c, 906  
 apps/cs/fsw/tables/cs\_memorytbl.c, 906  
 apps/cs/fsw/tables/cs\_tablestbl.c, 907  
**ARGCHECK**  
     osapi-macros.h, 1338  
**AtToneDelay**  
     CFE\_TIME\_DiagnosticTlm\_Payload, 635  
**AtToneLatch**  
     CFE\_TIME\_DiagnosticTlm\_Payload, 636  
**AtToneLeapSeconds**  
     CFE\_TIME\_DiagnosticTlm\_Payload, 636  
     CFE\_TIME\_ToneDataCmd\_Payload, 654  
**AtToneMET**  
     CFE\_TIME\_DiagnosticTlm\_Payload, 636  
     CFE\_TIME\_ToneDataCmd\_Payload, 654  
**AtToneState**  
     CFE\_TIME\_ToneDataCmd\_Payload, 654  
**AtToneSTCF**  
     CFE\_TIME\_DiagnosticTlm\_Payload, 636  
     CFE\_TIME\_ToneDataCmd\_Payload, 654  
  
**BitMask**  
     CFE\_EVS\_AppNameBitMaskCmd\_Payload, 565  
     CFE\_EVS\_BitMaskCmd\_Payload, 572  
**block\_size**  
     OS\_statvfs\_t, 695  
**blocks\_free**  
     OS\_statvfs\_t, 695  
**BlockSize**  
     CFE\_ES\_BlockStats, 532  
**BlockStats**  
     CFE\_ES\_MemPoolStats, 546  
**BootSource**  
     CFE\_ES\_HousekeepingTlm\_Payload, 540  
**bss\_address**  
     OS\_module\_address\_t, 689  
**bss\_size**  
     OS\_module\_address\_t, 689  
**BSSAddress**  
     CFE\_ES\_AppInfo, 527  
**BSSSize**  
     CFE\_ES\_AppInfo, 527  
**BUFF\_SIZE**  
     cfe\_psp.h, 1357  
**Buffer**  
     OS\_SockAddrData\_t, 693  
**BUGCHECK**  
     osapi-macros.h, 1338  
**BUGCHECK\_VOID**  
     osapi-macros.h, 1339  
**BUGREPORT**  
     osapi-macros.h, 1339  
  
     osapi-macros.h, 1339  
     buildosal\_public\_api/inc/osconfig.h, 907  
**ByteAlign4**  
     CFE\_TBL\_TblRegPacket\_Payload, 629  
**ByteAlignPad1**  
     CFE\_TBL\_HousekeepingTlm\_Payload, 619  
**ByteAlignSpare**  
     CFE\_ES\_CDSRegDumpRec, 533  
**ByteOffset**  
     CS\_Res\_App\_Table\_Entry\_t, 676  
     CS\_Res\_EepromMemory\_Table\_Entry\_t, 678  
     CS\_Res\_Tables\_Table\_Entry\_t, 680  
  
**CCSDS\_ExtendedHeader**, 525  
     Subsystem, 525  
     SystemId, 525  
**CCSDS\_ExtendedHeader\_t**  
     ccsds\_hdr.h, 1170  
**ccsds\_hdr.h**  
     CCSDS\_ExtendedHeader\_t, 1170  
     CCSDS\_PrimaryHeader\_t, 1171  
**CCSDS\_PrimaryHeader**, 525  
     Length, 526  
     Sequence, 526  
     StreamId, 526  
**CCSDS\_PrimaryHeader\_t**  
     ccsds\_hdr.h, 1171  
**CdsName**  
     CFE\_ES\_DeleteCDSCmd\_Payload, 535  
**cFE Access Table Content APIs**, 368  
     CFE\_TBL\_GetAddress, 368  
     CFE\_TBL\_GetAddresses, 369  
     CFE\_TBL\_ReleaseAddress, 370  
     CFE\_TBL\_ReleaseAddresses, 371  
**cFE Application Behavior APIs**, 249  
     CFE\_ES\_ExitApp, 249  
     CFE\_ES\_IncrementTaskCounter, 249  
     CFE\_ES\_RunLoop, 250  
     CFE\_ES\_WaitForStartupSync, 251  
     CFE\_ES\_WaitForSystemState, 251  
**cFE Application Control APIs**, 246  
     CFE\_ES\_DeleteApp, 246  
     CFE\_ES\_ReloadApp, 246  
     CFE\_ES\_RestartApp, 247  
**cFE Child Task APIs**, 262  
     CFE\_ES\_CreateChildTask, 262  
     CFE\_ES\_DeleteChildTask, 263  
     CFE\_ES\_ExitChildTask, 264  
     CFE\_ES\_GetTaskIDByName, 264  
     CFE\_ES\_GetTaskName, 265  
**cFE Clock State Flag Defines**, 399  
     CFE\_TIME\_FLAG\_ADD1HZ, 399  
     CFE\_TIME\_FLAG\_ADDADJ, 399  
     CFE\_TIME\_FLAG\_ADDTCL, 399

CFE\_TIME\_FLAG\_CLKSET, 399  
CFE\_TIME\_FLAG\_CMDFLY, 400  
CFE\_TIME\_FLAG\_FLYING, 400  
CFE\_TIME\_FLAG\_GDTONE, 400  
CFE\_TIME\_FLAG\_REFERR, 400  
CFE\_TIME\_FLAG\_SERVER, 400  
CFE\_TIME\_FLAG\_SIGPRI, 400  
CFE\_TIME\_FLAG\_SRCINT, 400  
CFE\_TIME\_FLAG\_SRVFLY, 400  
CFE\_TIME\_FLAG\_UNUSED, 400

cFE Critical Data Store APIs, 270  
CFE\_ES\_CopyToCDS, 270  
CFE\_ES\_GetCDSBlockIDByName, 271  
CFE\_ES\_GetCDSBlockName, 271  
CFE\_ES\_RegisterCDS, 272  
CFE\_ES\_RestoreFromCDS, 273

cFE Entry/Exit APIs, 244  
CFE\_ES\_Main, 244  
CFE\_ES\_ResetCFE, 244

cFE External Time Source APIs, 389  
CFE\_TIME\_ExternalGPS, 389  
CFE\_TIME\_ExternalMET, 390  
CFE\_TIME\_ExternalTime, 390  
CFE\_TIME\_ExternalTone, 391  
CFE\_TIME\_RegisterSynchCallback, 391  
CFE\_TIME\_UnregisterSynchCallback, 392

cFE File Header Management APIs, 299  
CFE\_FS\_InitHeader, 299  
CFE\_FS\_ReadHeader, 299  
CFE\_FS\_SetTimestamp, 300  
CFE\_FS\_WriteHeader, 301

cFE File Utility APIs, 303  
CFE\_FS\_BackgroundFileDumpIsPending, 303  
CFE\_FS\_BackgroundFileDumpRequest, 304  
CFE\_FS\_ExtractFilenameFromPath, 304  
CFE\_FS\_GetDefaultExtension, 305  
CFE\_FS\_GetDefaultMountPoint, 305  
CFE\_FS\_ParseInputFileName, 305  
CFE\_FS\_ParseInputFileNameEx, 306

cFE Generic Counter APIs, 284  
CFE\_ES\_DeleteGenCounter, 284  
CFE\_ES\_GetGenCount, 285  
CFE\_ES\_GetGenCounterIDByName, 285  
CFE\_ES\_GetGenCounterName, 286  
CFE\_ES\_IncrementGenCounter, 287  
CFE\_ES\_RegisterGenCounter, 287  
CFE\_ES\_SetGenCount, 288

cFE Generic Message APIs, 308  
CFE\_MSG\_Init, 308  
CFE\_MSG\_UpdateHeader, 308

cFE Get Current Time APIs, 378  
CFE\_TIME\_GetMET, 378  
CFE\_TIME\_GetMETseconds, 378  
CFE\_TIME\_GetMETsubsecs, 379

CFE\_TIME\_GetTAI, 379  
CFE\_TIME\_GetTime, 380  
CFE\_TIME\_GetUTC, 380

cFE Get Table Information APIs, 373  
CFE\_TBL\_GetInfo, 373  
CFE\_TBL\_GetStatus, 374  
CFE\_TBL\_NotifyByMessage, 374

cFE Get Time Information APIs, 381  
CFE\_TIME\_GetClockInfo, 381  
CFE\_TIME\_GetClockState, 381  
CFE\_TIME\_GetLeapSeconds, 382  
CFE\_TIME\_GetSTCF, 382

cFE Information APIs, 253  
CFE\_ES\_GetAppID, 253  
CFE\_ES\_GetAppIDByName, 254  
CFE\_ES\_GetAppInfo, 254  
CFE\_ES\_GetAppName, 255  
CFE\_ES\_GetLibIDByName, 256  
CFE\_ES\_GetLibInfo, 257  
CFE\_ES\_GetLibName, 257  
CFE\_ES\_GetModuleInfo, 258  
CFE\_ES\_GetResetType, 259  
CFE\_ES\_GetTaskID, 260  
CFE\_ES\_GetTaskInfo, 260

cFE Manage Table Content APIs, 362  
CFE\_TBL\_DumpToBuffer, 362  
CFE\_TBL\_Load, 363  
CFE\_TBL\_Manage, 364  
CFE\_TBL\_Modified, 365  
CFE\_TBL\_Update, 366  
CFE\_TBL\_Validate, 366

cFE Memory Manager APIs, 275  
CFE\_ES\_GetMemPoolStats, 275  
CFE\_ES\_GetPoolBuf, 276  
CFE\_ES\_GetPoolBufInfo, 276  
CFE\_ES\_PoolCreate, 277  
CFE\_ES\_PoolCreateEx, 278  
CFE\_ES\_PoolCreateNoSem, 279  
CFE\_ES\_PoolDelete, 280  
CFE\_ES\_PutPoolBuf, 281

cFE Message Characteristics APIs, 349  
CFE\_SB\_GetUserData, 349  
CFE\_SB\_GetUserDataLength, 349  
CFE\_SB\_MessageStringGet, 350  
CFE\_SB\_MessageStringSet, 351  
CFE\_SB\_SetUserDataLength, 352  
CFE\_SB\_TimeStampMsg, 352

cFE Message Checking APIs, 332  
CFE\_MSG\_Verify, 332

cFE Message Extended Header APIs, 319  
CFE\_MSG\_GetEDSVersion, 319  
CFE\_MSG\_GetEndian, 320  
CFE\_MSG\_GetPlaybackFlag, 320  
CFE\_MSG\_GetSubsystem, 321

CFE\_MSG\_GetSystem, 321  
 CFE\_MSG\_SetEDSVersion, 322  
 CFE\_MSG\_SetEndian, 322  
 CFE\_MSG\_SetPlaybackFlag, 323  
 CFE\_MSG\_SetSubsystem, 323  
 CFE\_MSG\_SetSystem, 324  
 cFE Message ID APIs, 354  
 CFE\_SB\_IsValidMsgId, 354  
 CFE\_SB\_MsgId\_Equal, 354  
 CFE\_SB\_MsgIdToValue, 355  
 CFE\_SB\_ValueToMsgId, 355  
 cFE Message Id APIs, 330  
 CFE\_MSG\_GetMsgId, 330  
 CFE\_MSG.GetTypeFromMsgId, 330  
 CFE\_MSG\_SetMsgId, 331  
 cFE Message Primary Header APIs, 310  
 CFE\_MSG\_GetApld, 310  
 CFE\_MSG\_GetHasSecondaryHeader, 311  
 CFE\_MSG\_GetHeaderVersion, 311  
 CFE\_MSG\_GetNextSequenceCount, 312  
 CFE\_MSG\_GetSegmentationFlag, 312  
 CFE\_MSG\_GetSequenceCount, 313  
 CFE\_MSG.GetSize, 313  
 CFE\_MSG.GetType, 314  
 CFE\_MSG\_SetApld, 314  
 CFE\_MSG\_SetHasSecondaryHeader, 315  
 CFE\_MSG\_SetHeaderVersion, 315  
 CFE\_MSG\_SetSegmentationFlag, 316  
 CFE\_MSG\_SetSequenceCount, 316  
 CFE\_MSG\_SetSize, 317  
 CFE\_MSG\_SetType, 317  
 cFE Message Secondary Header APIs, 325  
 CFE\_MSG\_GenerateChecksum, 325  
 CFE\_MSG\_GetFcnCode, 326  
 CFE\_MSG\_GetMsgTime, 326  
 CFE\_MSG\_SetFcnCode, 327  
 CFE\_MSG\_SetMsgTime, 327  
 CFE\_MSG\_ValidateChecksum, 328  
 cFE Message Subscription Control APIs, 338  
 CFE\_SB\_Subscribe, 338  
 CFE\_SB\_SubscribeEx, 339  
 CFE\_SB\_SubscribeLocal, 340  
 CFE\_SB\_Unsubscribe, 340  
 CFE\_SB\_UnsubscribeLocal, 341  
 cFE Miscellaneous APIs, 267  
 CFE\_ES\_BackgroundWakeup, 267  
 CFE\_ES\_CalculateCRC, 267  
 CFE\_ES\_ProcessAsyncEvent, 268  
 CFE\_ES\_WriteToSysLog, 268  
 cFE Miscellaneous Time APIs, 394  
 CFE\_TIME\_Local1HzISR, 394  
 CFE\_TIME\_Print, 394  
 cFE Performance Monitor APIs, 282  
 CFE\_ES\_PerfLogAdd, 283  
 CFE\_ES\_PerfLogEntry, 282  
 CFE\_ES\_PerfLogExit, 282  
 cFE Pipe Management APIs, 333  
 CFE\_SB\_CreatePipe, 333  
 CFE\_SB\_DeletePipe, 334  
 CFE\_SB\_GetPipeIdByName, 334  
 CFE\_SB\_GetPipeName, 335  
 CFE\_SB\_GetPipeOpts, 336  
 CFE\_SB\_PipeId\_ToIndex, 336  
 CFE\_SB\_SetPipeOpts, 337  
 cFE Registration APIs, 290, 357  
 CFE\_EVS\_Register, 290  
 CFE\_TBL\_Register, 357  
 CFE\_TBL\_Share, 359  
 CFE\_TBL\_Unregister, 360  
 cFE Reset Event Filter APIs, 297  
 CFE\_EVS\_ResetAllFilters, 297  
 CFE\_EVS\_ResetFilter, 297  
 cFE Resource ID APIs, 241  
 CFE\_ES\_AppID\_ToIndex, 241  
 CFE\_ES\_CounterID\_ToIndex, 241  
 CFE\_ES\_LibID\_ToIndex, 242  
 CFE\_ES\_TaskID\_ToIndex, 243  
 cFE Resource ID base values, 397  
 CFE\_CONFIGID\_BASE, 398  
 CFE\_ES\_APPID\_BASE, 397  
 CFE\_ES\_CDSBLOCKID\_BASE, 398  
 CFE\_ES\_COUNTID\_BASE, 397  
 CFE\_ES\_LIBID\_BASE, 397  
 CFE\_ES\_POOLID\_BASE, 398  
 CFE\_ES\_TASKID\_BASE, 397  
 CFE\_RESOURCEID\_CONFIGID\_BASE\_OFFSET,  
 397  
 CFE\_RESOURCEID\_ES\_APPID\_BASE\_OFFSET,  
 397  
 CFE\_RESOURCEID\_ES\_CDSBLOCKID\_BASE\_OFFSET,  
 397  
 CFE\_RESOURCEID\_ES\_COUNTID\_BASE\_OFFSET,  
 397  
 CFE\_RESOURCEID\_ES\_LIBID\_BASE\_OFFSET,  
 397  
 CFE\_RESOURCEID\_ES\_POOLID\_BASE\_OFFSET,  
 397  
 CFE\_RESOURCEID\_ES\_TASKID\_BASE\_OFFSET,  
 397  
 CFE\_RESOURCEID\_SB\_PIPEID\_RESOURCE\_BASE\_OFFSET,  
 397  
 CFE\_SB\_PIPEID\_BASE, 398  
 cFE Return Code Defines, 218  
 CFE\_ES\_APP\_CLEANUP\_ERR, 223  
 CFE\_ES\_BAD\_ARGUMENT, 223  
 CFE\_ES\_BIN\_SEM\_DELETE\_ERR, 223  
 CFE\_ES\_BUFFER\_NOT\_IN\_POOL, 223  
 CFE\_ES\_CDS\_ACCESS\_ERROR, 223

CFE\_ES\_CDS\_ALREADY\_EXISTS, 224  
CFE\_ES\_CDS\_BLOCK\_CRC\_ERR, 224  
CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY, 224  
CFE\_ES\_CDS\_INVALID, 224  
CFE\_ES\_CDS\_INVALID\_NAME, 224  
CFE\_ES\_CDS\_INVALID\_SIZE, 224  
CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR, 224  
CFE\_ES\_CDS\_WRONG\_TYPE\_ERR, 224  
CFE\_ES\_COUNT\_SEM\_DELETE\_ERR, 225  
CFE\_ES\_ERR\_APP\_CREATE, 225  
CFE\_ES\_ERR\_APP\_REGISTER, 225  
CFE\_ES\_ERR\_CHILD\_TASK\_CREATE, 225  
CFE\_ES\_ERR\_CHILD\_TASK\_DELETE, 225  
CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK,  
    225  
CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER, 225  
CFE\_ES\_ERR\_DUPLICATE\_NAME, 225  
CFE\_ES\_ERR\_LOAD\_LIB, 226  
CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE, 226  
CFE\_ES\_ERR\_NAME\_NOT\_FOUND, 226  
CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID, 226  
CFE\_ES\_ERR\_SYS\_LOG\_FULL, 226  
CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED, 226  
CFE\_ES\_FILE\_CLOSE\_ERR, 226  
CFE\_ES\_FILE\_IO\_ERR, 226  
CFE\_ES\_LIB\_ALREADY\_LOADED, 227  
CFE\_ES\_MUT\_SEM\_DELETE\_ERR, 227  
CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE, 227  
CFE\_ES\_NOT\_IMPLEMENTED, 227  
CFE\_ES\_OPERATION\_TIMED\_OUT, 227  
CFE\_ES\_POOL\_BLOCK\_INVALID, 227  
CFE\_ES\_QUEUE\_DELETE\_ERR, 227  
CFE\_ES\_RST\_ACCESS\_ERR, 227  
CFE\_ES\_TASK\_DELETE\_ERR, 227  
CFE\_ES\_TIMER\_DELETE\_ERR, 228  
CFE\_EVS\_APP\_FILTER\_OVERLOAD, 228  
CFE\_EVS\_APP\_ILLEGAL\_APP\_ID, 228  
CFE\_EVS\_APP\_NOT\_REGISTERED, 228  
CFE\_EVS\_APP\_SQUELCHED, 228  
CFE\_EVS\_EVT\_NOT\_REGISTERED, 228  
CFE\_EVS\_FILE\_WRITE\_ERROR, 228  
CFE\_EVS\_INVALID\_PARAMETER, 228  
CFE\_EVS\_NOT\_IMPLEMENTED, 229  
CFE\_EVS\_RESET\_AREA\_POINTER, 229  
CFE\_EVS\_UNKNOWN\_FILTER, 229  
CFE\_FS\_BAD\_ARGUMENT, 229  
CFE\_FS\_FNAME\_TOO\_LONG, 229  
CFE\_FS\_INVALID\_PATH, 229  
CFE\_FS\_NOT\_IMPLEMENTED, 229  
CFE\_SB\_BAD\_ARGUMENT, 229  
CFE\_SB\_BUF\_ALOC\_ERR, 229  
CFE\_SB\_BUFFER\_INVALID, 230  
CFE\_SB\_INTERNAL\_ERR, 230  
CFE\_SB\_MAX\_DESTS\_MET, 230  
CFE\_SB\_MAX\_MSGS\_MET, 230  
CFE\_SB\_MAX\_PIPES\_MET, 230  
CFE\_SB\_MSG\_TOO\_BIG, 230  
CFE\_SB\_NO\_MESSAGE, 230  
CFE\_SB\_NOT\_IMPLEMENTED, 231  
CFE\_SB\_PIPE\_CR\_ERR, 231  
CFE\_SB\_PIPE\_RD\_ERR, 231  
CFE\_SB\_TIME\_OUT, 231  
CFE\_SB\_WRONG\_MSG\_TYPE, 231  
CFE\_STATUS\_BAD\_COMMAND\_CODE, 231  
CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL, 231  
CFE\_STATUS\_INCORRECT\_STATE, 232  
CFE\_STATUS\_NO\_COUNTER\_INCREMENT, 232  
CFE\_STATUS\_NOT\_IMPLEMENTED, 232  
CFE\_STATUS\_RANGE\_ERROR, 232  
CFE\_STATUS\_REQUEST\_ALREADY\_PENDING,  
    232  
CFE\_STATUS\_UNKNOWN\_MSG\_ID, 232  
CFE\_STATUS\_VALIDATION\_FAILURE, 232  
CFE\_STATUS\_WRONG\_MSG\_LENGTH, 233  
CFE\_SUCCESS, 233  
CFE\_TBL\_BAD\_ARGUMENT, 233  
CFE\_TBL\_ERR\_ACCESS, 233  
CFE\_TBL\_ERR\_BAD\_CONTENT\_ID, 233  
CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID, 233  
CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID, 233  
CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID, 233  
CFE\_TBL\_ERR\_DUMP\_ONLY, 234  
CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE, 234  
CFE\_TBL\_ERR\_DUPLICATE\_NOT\_OWNED, 234  
CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE, 234  
CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT, 234  
CFE\_TBL\_ERR\_FILE\_TOO\_LARGE, 234  
CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG, 234  
CFE\_TBL\_ERR\_HANDLES\_FULL, 234  
CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE, 235  
CFE\_TBL\_ERR\_INVALID\_HANDLE, 235  
CFE\_TBL\_ERR\_INVALID\_NAME, 235  
CFE\_TBL\_ERR\_INVALID\_OPTIONS, 235  
CFE\_TBL\_ERR\_INVALID\_SIZE, 235  
CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS, 235  
CFE\_TBL\_ERR\_LOAD\_INCOMPLETE, 236  
CFE\_TBL\_ERR\_NEVER\_LOADED, 236  
CFE\_TBL\_ERR\_NO\_ACCESS, 236  
CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL, 236  
CFE\_TBL\_ERR\_NO\_STD\_HEADER, 236  
CFE\_TBL\_ERR\_NO\_TBL\_HEADER, 236  
CFE\_TBL\_ERR\_PARTIAL\_LOAD, 236  
CFE\_TBL\_ERR\_REGISTRY\_FULL, 236  
CFE\_TBL\_ERR\_SHORT\_FILE, 237  
CFE\_TBL\_ERR\_UNREGISTERED, 237  
CFE\_TBL\_INFO\_DUMP\_PENDING, 237  
CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING, 237  
CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING, 237

CFE\_TBL\_INFO\_RECOVERED\_TBL, 237  
 CFE\_TBL\_INFO\_TABLE\_LOCKED, 237  
 CFE\_TBL\_INFO\_UPDATE\_PENDING, 237  
 CFE\_TBL\_INFO\_UPDATED, 238  
 CFE\_TBL\_INFO\_VALIDATION\_PENDING, 238  
 CFE\_TBL\_MESSAGE\_ERROR, 238  
 CFE\_TBL\_NOT\_IMPLEMENTED, 238  
 CFE\_TBL\_WARN\_DUPLICATE, 238  
 CFE\_TBL\_WARN\_NOT\_CRITICAL, 238  
 CFE\_TBL\_WARN\_PARTIAL\_LOAD, 238  
 CFE\_TBL\_WARN\_SHORT\_FILE, 239  
 CFE\_TIME\_BAD\_ARGUMENT, 239  
 CFE\_TIME\_CALLBACK\_NOT\_REGISTERED, 239  
 CFE\_TIME\_INTERNAL\_ONLY, 239  
 CFE\_TIME\_NOT\_IMPLEMENTED, 239  
 CFE\_TIME\_OUT\_OF\_RANGE, 239  
 CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS, 239

cFE SB Pipe options, 356  
 CFE\_SB\_PIPEOPTS\_IGNOREMINE, 356

cFE Send Event APIs, 292  
 CFE\_EVS\_SendEvent, 292  
 CFE\_EVS\_SendEventWithApplID, 293  
 CFE\_EVS\_SendTimedEvent, 295

cFE Send/Receive Message APIs, 343  
 CFE\_SB\_ReceiveBuffer, 343  
 CFE\_SB\_TransmitMsg, 344

cFE Table Type Defines, 376  
 CFE\_TBL\_OPT\_BUFFER\_MSK, 376  
 CFE\_TBL\_OPT\_CRITICAL, 376  
 CFE\_TBL\_OPT\_CRITICAL\_MSK, 376  
 CFE\_TBL\_OPT\_DBL\_BUFFER, 376  
 CFE\_TBL\_OPT\_DEFAULT, 377  
 CFE\_TBL\_OPT\_DUMP\_ONLY, 377  
 CFE\_TBL\_OPT\_LD\_DMP\_MSK, 377  
 CFE\_TBL\_OPT\_LOAD\_DUMP, 377  
 CFE\_TBL\_OPT\_NOT\_CRITICAL, 377  
 CFE\_TBL\_OPT\_NOT\_USR\_DEF, 377  
 CFE\_TBL\_OPT\_SNGL\_BUFFER, 377  
 CFE\_TBL\_OPT\_USR\_DEF\_ADDR, 377  
 CFE\_TBL\_OPT\_USR\_DEF\_MSK, 377

cFE Time Arithmetic APIs, 384  
 CFE\_TIME\_Add, 384  
 CFE\_TIME\_Compare, 384  
 CFE\_TIME\_Subtract, 385

cFE Time Conversion APIs, 387  
 CFE\_TIME\_MET2SCTime, 387  
 CFE\_TIME\_Micro2SubSecs, 387  
 CFE\_TIME\_Sub2MicroSecs, 388

cFE Zero Copy APIs, 346  
 CFE\_SB\_AllocateMessageBuffer, 346  
 CFE\_SB\_ReleaseMessageBuffer, 346  
 CFE\_SB\_TransmitBuffer, 347

cfe/cmake/sample\_defs/example\_mission\_cfg.h, 913

cfe/cmake/sample\_defs/example\_platform\_cfg.h, 924  
 cfe/cmake/sample\_defs/sample\_perfids.h, 968  
 cfe/docs/src/cfe\_api.dox, 971  
 cfe/docs/src/cfe\_es.dox, 971  
 cfe/docs/src/cfe\_evs.dox, 971  
 cfe/docs/src/cfe\_frontpage.dox, 971  
 cfe/docs/src/cfe\_glossary.dox, 971  
 cfe/docs/src/cfe\_sb.dox, 971  
 cfe/docs/src/cfe\_tbl.dox, 971  
 cfe/docs/src/cfe\_time.dox, 971  
 cfe/docs/src/cfe\_xref.dox, 971  
 cfe/docs/src/cfs\_versions.dox, 971  
 cfe/modules/core\_api/config/default\_cfe\_core\_api\_base\_msgids.h, 971  
 cfe/modules/core\_api/config/default\_cfe\_core\_api\_interface\_cfg.h, 972  
 cfe/modules/core\_api/config/default\_cfe\_mission\_cfg.h, 974  
 cfe/modules/core\_api/config/default\_cfe\_msgids.h, 974  
 cfe/modules/core\_api/fsw/inc/cfe.h, 974  
 cfe/modules/core\_api/fsw/inc/cfe\_config.h, 975  
 cfe/modules/core\_api/fsw/inc/cfe\_config\_api\_typedefs.h, 978  
 cfe/modules/core\_api/fsw/inc/cfe\_endian.h, 979  
 cfe/modules/core\_api/fsw/inc/cfe\_error.h, 979  
 cfe/modules/core\_api/fsw/inc/cfe\_es.h, 988  
 cfe/modules/core\_api/fsw/inc/cfe\_es\_api\_typedefs.h, 991  
 cfe/modules/core\_api/fsw/inc/cfe\_evs.h, 996  
 cfe/modules/core\_api/fsw/inc/cfe\_evs\_api\_typedefs.h, 998  
 cfe/modules/core\_api/fsw/inc/cfe\_fs.h, 1000  
 cfe/modules/core\_api/fsw/inc/cfe\_fs\_api\_typedefs.h, 1001  
 cfe/modules/core\_api/fsw/inc/cfe\_msg.h, 1003  
 cfe/modules/core\_api/fsw/inc/cfe\_msg\_api\_typedefs.h, 1005  
 cfe/modules/core\_api/fsw/inc/cfe\_resourceid.h, 1010  
 cfe/modules/core\_api/fsw/inc/cfe\_resourceid\_api\_typedefs.h, 1015  
 cfe/modules/core\_api/fsw/inc/cfe\_sb.h, 1016  
 cfe/modules/core\_api/fsw/inc/cfe\_sb\_api\_typedefs.h, 1018  
 cfe/modules/core\_api/fsw/inc/cfe\_tbl.h, 1021  
 cfe/modules/core\_api/fsw/inc/cfe\_tbl\_api\_typedefs.h, 1022  
 cfe/modules/core\_api/fsw/inc/cfe\_tbl\_filedef.h, 1025  
 cfe/modules/core\_api/fsw/inc/cfe\_time.h, 1026  
 cfe/modules/core\_api/fsw/inc/cfe\_time\_api\_typedefs.h, 1028  
 cfe/modules/core\_api/fsw/inc/cfe\_version.h, 1029  
 cfe/modules/es/config/default\_cfe\_es\_extern\_typedefs.h, 1031  
 cfe/modules/es/config/default\_cfe\_es\_fcncodes.h, 1040  
 cfe/modules/es/config/default\_cfe\_es\_interface\_cfg.h, 1060  
 cfe/modules/es/config/default\_cfe\_es\_internal\_cfg.h,

1063  
cfe/modules/es/config/default\_cfe\_es\_mission\_cfg.h,  
1083  
cfe/modules/es/config/default\_cfe\_es\_msg.h, 1083  
cfe/modules/es/config/default\_cfe\_es\_msgdefs.h, 1084  
cfe/modules/es/config/default\_cfe\_es\_msgids.h, 1084  
cfe/modules/es/config/default\_cfe\_es\_msgstruct.h, 1085  
cfe/modules/es/config/default\_cfe\_es\_platform\_cfg.h,  
1092  
cfe/modules/es/config/default\_cfe\_es\_topicids.h, 1093  
cfe/modules/es/fsw/inc/cfe\_es\_eventids.h, 1094  
cfe/modules/evs/config/default\_cfe\_evs\_extern\_typedefs.h,  
1119  
cfe/modules/evs/config/default\_cfe\_evs\_fcncodes.h, 1122  
cfe/modules/evs/config/default\_cfe\_evs\_interface\_cfg.h,  
1140  
cfe/modules/evs/config/default\_cfe\_evs\_internal\_cfg.h,  
1141  
cfe/modules/evs/config/default\_cfe\_evs\_mission\_cfg.h,  
1145  
cfe/modules/evs/config/default\_cfe\_evs\_msg.h, 1145  
cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h, 1145  
cfe/modules/evs/config/default\_cfe\_evs\_msgids.h, 1146  
cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h,  
1147  
cfe/modules/evs/config/default\_cfe\_evs\_platform\_cfg.h,  
1154  
cfe/modules/evs/config/default\_cfe\_evs\_topicids.h, 1154  
cfe/modules/evs/fsw/inc/cfe\_evs\_eventids.h, 1155  
cfe/modules/fs/config/default\_cfe\_fs\_extern\_typedefs.h,  
1167  
cfe/modules/fs/config/default\_cfe\_fs\_filedef.h, 1167  
cfe/modules/fs/config/default\_cfe\_fs\_interface\_cfg.h, 1169  
cfe/modules/fs/config/default\_cfe\_fs\_mission\_cfg.h, 1170  
cfe/modules/msg/fsw/inc/ccsds\_hdr.h, 1170  
cfe/modules/resourceid/fsw/inc/cfe\_core\_resourceid\_basevalues.h,  
1171  
cfe/modules/resourceid/fsw/inc/cfe\_resourceid\_basevalue.h,  
1171  
cfe/modules/sb/config/default\_cfe\_sb\_extern\_typedefs.h,  
1172  
cfe/modules/sb/config/default\_cfe\_sb\_fcncodes.h, 1174  
cfe/modules/sb/config/default\_cfe\_sb\_interface\_cfg.h,  
1184  
cfe/modules/sb/config/default\_cfe\_sb\_internal\_cfg.h,  
1185  
cfe/modules/sb/config/default\_cfe\_sb\_mission\_cfg.h,  
1193  
cfe/modules/sb/config/default\_cfe\_sb\_msg.h, 1194  
cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h, 1194  
cfe/modules/sb/config/default\_cfe\_sb\_msgids.h, 1194  
cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h, 1195  
cfe/modules/sb/config/default\_cfe\_sb\_platform\_cfg.h,  
1200  
cfe/modules/sb/config/default\_cfe\_sb\_topicids.h, 1200  
cfe/modules/sb/fsw/inc/cfe\_sb\_eventids.h, 1201  
cfe/modules/tbl/config/default\_cfe\_tbl\_extern\_typedefs.h,  
1220  
cfe/modules/tbl/config/default\_cfe\_tbl\_fcncodes.h, 1222  
cfe/modules/tbl/config/default\_cfe\_tbl\_interface\_cfg.h,  
1231  
cfe/modules/tbl/config/default\_cfe\_tbl\_internal\_cfg.h, 1232  
cfe/modules/tbl/config/default\_cfe\_tbl\_mission\_cfg.h,  
1238  
cfe/modules/tbl/config/default\_cfe\_tbl\_msg.h, 1238  
cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h, 1238  
cfe/modules/tbl/config/default\_cfe\_tbl\_msgids.h, 1239  
cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h, 1239  
cfe/modules/tbl/config/default\_cfe\_tbl\_platform\_cfg.h,  
1244  
cfe/modules/tbl/config/default\_cfe\_tbl\_topicids.h, 1244  
cfe/modules/tbl/fsw/inc/cfe\_tbl\_eventids.h, 1245  
cfe/modules/time/config/default\_cfe\_time\_extern\_typedefs.h,  
1265  
cfe/modules/time/config/default\_cfe\_time\_fcncodes.h,  
1270  
cfe/modules/time/config/default\_cfe\_time\_interface\_cfg.h,  
1285  
cfe/modules/time/config/default\_cfe\_time\_internal\_cfg.h,  
1290  
cfe/modules/time/config/default\_cfe\_time\_mission\_cfg.h,  
1295  
cfe/modules/time/config/default\_cfe\_time\_msg.h, 1295  
cfe/modules/time/config/default\_cfe\_time\_msgdefs.h,  
1296  
cfe/modules/time/config/default\_cfe\_time\_msgids.h, 1297  
cfe/modules/time/config/default\_cfe\_time\_msgstruct.h,  
1298  
cfe/modules/time/config/default\_cfe\_time\_platform\_cfg.h,  
1303  
cfe/modules/time/config/default\_cfe\_time\_topicids.h, 1303  
cfe/modules/time/fsw/inc/cfe\_time\_eventids.h, 1305  
CFE\_BIT  
    cfe\_sb.h, 1018  
CFE\_BUILD\_BASELINE  
    cfe\_version.h, 1030  
CFE\_BUILD\_NUMBER  
    cfe\_version.h, 1030  
CFE\_CLR  
    cfe\_sb.h, 1018  
cfe\_config.h  
    CFE\_Config\_GetIdByName, 975  
    CFE\_Config\_GetName, 976  
    CFE\_Config\_GetObjPointer, 976  
    CFE\_Config\_GetString, 977  
    CFE\_Config\_GetValue, 977  
    CFE\_Config\_IterateAll, 977  
cfe\_config\_api\_typedefs.h

CFE\_Config\_Callback\_t, [978](#)  
 CFE\_CONFIGID\_C, [978](#)  
 CFE\_ConfigId\_t, [978](#)  
 CFE\_CONFIGID\_UNDEFINED, [978](#)  
 CFE\_Config\_Callback\_t  
   cfe\_config\_api\_typedefs.h, [978](#)  
 CFE\_Config\_GetIdByName  
   cfe\_config.h, [975](#)  
 CFE\_Config\_GetName  
   cfe\_config.h, [976](#)  
 CFE\_Config\_GetObjPointer  
   cfe\_config.h, [976](#)  
 CFE\_Config\_GetString  
   cfe\_config.h, [977](#)  
 CFE\_Config\_GetValue  
   cfe\_config.h, [977](#)  
 CFE\_Config\_IterateAll  
   cfe\_config.h, [977](#)  
 CFE\_CONFIGID\_BASE  
   cFE Resource ID base values, [398](#)  
 CFE\_CONFIGID\_C  
   cfe\_config\_api\_typedefs.h, [978](#)  
 CFE\_ConfigId\_t  
   cfe\_config\_api\_typedefs.h, [978](#)  
 CFE\_CONFIGID\_UNDEFINED  
   cfe\_config\_api\_typedefs.h, [978](#)  
 cfe\_endian.h  
   CFE\_MAKE\_BIG16, [979](#)  
   CFE\_MAKE\_BIG32, [979](#)  
 cfe\_error.h  
   CFE\_ES\_StatusToString, [987](#)  
   CFE\_EVENTS\_SERVICE, [986](#)  
   CFE\_EXECUTIVE\_SERVICE, [986](#)  
   CFE\_FILE\_SERVICE, [986](#)  
   CFE\_GENERIC\_SERVICE, [986](#)  
   CFE\_SERVICE\_BITMASK, [986](#)  
   CFE\_SEVERITY\_BITMASK, [986](#)  
   CFE\_SEVERITY\_ERROR, [986](#)  
   CFE\_SEVERITY\_INFO, [986](#)  
   CFE\_SEVERITY\_SUCCESS, [986](#)  
   CFE\_SOFTWARE\_BUS\_SERVICE, [987](#)  
   CFE\_STATUS\_C, [987](#)  
   CFE\_STATUS\_STRING\_LENGTH, [987](#)  
   CFE\_Status\_t, [987](#)  
   CFE\_StatusString\_t, [987](#)  
   CFE\_TABLE\_SERVICE, [987](#)  
   CFE\_TIME\_SERVICE, [987](#)  
 cfe\_es.h  
   CFE\_ES\_DBIT, [991](#)  
   CFE\_ES\_DTEST, [991](#)  
   CFE\_ES\_TEST\_LONG\_MASK, [991](#)  
   OS\_PRINTF, [991](#)  
 CFE\_ES\_ALL\_APPS\_EID  
   cfe\_es\_eventids.h, [1097](#)  
 cfe\_es\_api\_typedefs.h  
   CFE\_ES\_APP\_RESTART, [993](#)  
   CFE\_ES\_APPID\_C, [993](#)  
   CFE\_ES\_APPID\_UNDEFINED, [993](#)  
   CFE\_ES\_CDS\_BAD\_HANDLE, [993](#)  
   CFE\_ES\_CDSHANDLE\_C, [993](#)  
   CFE\_ES\_ChildTaskMainFuncPtr\_t, [995](#)  
   CFE\_ES\_COUNTERID\_C, [993](#)  
   CFE\_ES\_COUNTERID\_UNDEFINED, [993](#)  
   CFE\_ES\_CRC\_TYPE\_CRC\_16, [996](#)  
   CFE\_ES\_CRC\_TYPE\_CRC\_32, [996](#)  
   CFE\_ES\_CRC\_TYPE\_CRC\_8, [996](#)  
   CFE\_ES\_CRC\_TYPE\_ENUM, [996](#)  
   CFE\_ES\_CRC\_TYPE\_ENUM\_T, [995](#)  
   CFE\_ES\_LIBID\_C, [993](#)  
   CFE\_ES\_LIBID\_UNDEFINED, [993](#)  
   CFE\_ES\_LibraryEntryFuncPtr\_t, [995](#)  
   CFE\_ES\_MEMHANDLE\_C, [994](#)  
   CFE\_ES\_MEMHANDLE\_UNDEFINED, [994](#)  
   CFE\_ES\_MEMPOOLBUF\_C, [994](#)  
   CFE\_ES\_MemPoolBuf\_t, [995](#)  
   CFE\_ES\_NO\_MUTEX, [994](#)  
   CFE\_ES\_PoolAlign\_t, [995](#)  
   CFE\_ES\_StackPointer\_t, [995](#)  
   CFE\_ES\_STATIC\_POOL\_TYPE, [994](#)  
   CFE\_ES\_TASK\_STACK\_ALLOCATE, [994](#)  
   CFE\_ES\_TaskEntryFuncPtr\_t, [996](#)  
   CFE\_ES\_TASKID\_C, [994](#)  
   CFE\_ES\_TASKID\_UNDEFINED, [995](#)  
   CFE\_ES\_USE\_MUTEX, [995](#)  
   CFE\_ES\_APP\_CLEANUP\_ERR  
     cFE Return Code Defines, [223](#)  
 CFE\_ES\_APP\_RESTART  
   cfe\_es\_api\_typedefs.h, [993](#)  
 CFE\_ES\_APP\_TLM\_MID  
   default\_cfe\_es\_msgids.h, [1084](#)  
 CFE\_ES\_APPID\_BASE  
   cFE Resource ID base values, [397](#)  
 CFE\_ES\_APPID\_C  
   cfe\_es\_api\_typedefs.h, [993](#)  
 CFE\_ES\_AppId\_t  
   default\_cfe\_es\_extern\_typedefs.h, [1034](#)  
 CFE\_ES\_AppID\_ToIndex  
   cFE Resource ID APIs, [241](#)  
 CFE\_ES\_APPID\_UNDEFINED  
   cfe\_es\_api\_typedefs.h, [993](#)  
 CFE\_ES\_AppInfo, [526](#)  
   AddressesAreValid, [527](#)  
   BSSAddress, [527](#)  
   BSSSize, [527](#)  
   CodeAddress, [527](#)  
   CodeSize, [528](#)  
   DataAddress, [528](#)  
   DataSize, [528](#)

EntryPoint, 528  
ExceptionAction, 528  
ExecutionCounter, 528  
FileName, 528  
MainTaskId, 529  
MainTaskName, 529  
Name, 529  
NumOfChildTasks, 529  
Priority, 529  
ResourceId, 529  
StackSize, 529  
StartAddress, 530  
Type, 530  
**CFE\_ES\_AppInfo\_t**  
    default\_cfe\_es\_extern\_typedefs.h, 1034  
**CFE\_ES\_AppNameCmd**, 530  
    CommandHeader, 530  
    Payload, 530  
**CFE\_ES\_AppNameCmd\_Payload**, 531  
    Application, 531  
**CFE\_ES\_AppNameCmd\_Payload\_t**  
    default\_cfe\_es\_msgstruct.h, 1088  
**CFE\_ES\_AppNameCmd\_t**  
    default\_cfe\_es\_msgstruct.h, 1088  
**CFE\_ES\_AppReloadCmd\_Payload**, 531  
    AppFileName, 531  
    Application, 532  
**CFE\_ES\_AppReloadCmd\_Payload\_t**  
    default\_cfe\_es\_msgstruct.h, 1088  
**CFE\_ES\_AppState**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppState\_EARLY\_INIT**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppState\_Enum\_t**  
    default\_cfe\_es\_extern\_typedefs.h, 1034  
**CFE\_ES\_AppState\_LATE\_INIT**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppState\_MAX**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppState\_RUNNING**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppState\_STOPPED**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppState\_UNDEFINED**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppState\_WAITING**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppType**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppType\_CORE**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppType\_Enum\_t**  
    default\_cfe\_es\_extern\_typedefs.h, 1035  
**CFE\_ES\_AppType\_EXTERNAL**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_AppType\_LIBRARY**  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
**CFE\_ES\_BackgroundWakeups**  
    cFE Miscellaneous APIs, 267  
**CFE\_ES\_BAD\_ARGUMENT**  
    cFE Return Code Defines, 223  
**CFE\_ES\_BIN\_SEM\_DELETE\_ERR**  
    cFE Return Code Defines, 223  
**CFE\_ES\_BlockStats**, 532  
    BlockSize, 532  
    NumCreated, 532  
    NumFree, 532  
**CFE\_ES\_BlockStats\_t**  
    default\_cfe\_es\_extern\_typedefs.h, 1035  
**CFE\_ES\_BOOT\_ERR\_EID**  
    cfe\_es\_eventids.h, 1097  
**CFE\_ES\_BUFFER\_NOT\_IN\_POOL**  
    cFE Return Code Defines, 223  
**CFE\_ES\_BUILD\_INF\_EID**  
    cfe\_es\_eventids.h, 1098  
**CFE\_ES\_CalculateCRC**  
    cFE Miscellaneous APIs, 267  
**CFE\_ES\_CC1\_ERR\_EID**  
    cfe\_es\_eventids.h, 1098  
**CFE\_ES\_CDS\_ACCESS\_ERROR**  
    cFE Return Code Defines, 223  
**CFE\_ES\_CDS\_ALREADY\_EXISTS**  
    cFE Return Code Defines, 224  
**CFE\_ES\_CDS\_BAD\_HANDLE**  
    cfe\_es\_api\_typedefs.h, 993  
**CFE\_ES\_CDS\_BLOCK\_CRC\_ERR**  
    cFE Return Code Defines, 224  
**CFE\_ES\_CDS\_DELETE\_ERR\_EID**  
    cfe\_es\_eventids.h, 1098  
**CFE\_ES\_CDS\_DELETE\_TBL\_ERR\_EID**  
    cfe\_es\_eventids.h, 1098  
**CFE\_ES\_CDS\_DELETED\_INFO\_EID**  
    cfe\_es\_eventids.h, 1099  
**CFE\_ES\_CDS\_DUMP\_ERR\_EID**  
    cfe\_es\_eventids.h, 1099  
**CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY**  
    cFE Return Code Defines, 224  
**CFE\_ES\_CDS\_INVALID**  
    cFE Return Code Defines, 224  
**CFE\_ES\_CDS\_INVALID\_NAME**  
    cFE Return Code Defines, 224  
**CFE\_ES\_CDS\_INVALID\_SIZE**  
    cFE Return Code Defines, 224  
**CFE\_ES\_CDS\_NAME\_ERR\_EID**  
    cfe\_es\_eventids.h, 1099  
**CFE\_ES\_CDS\_OWNER\_ACTIVE\_EID**  
    cfe\_es\_eventids.h, 1099  
**CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR**

cFE Return Code Defines, [224](#)  
**CFE\_ES\_CDS\_REG\_DUMP\_INF\_EID**  
 cfe\_es\_eventids.h, [1100](#)  
**CFE\_ES\_CDS\_REGISTER\_ERR\_EID**  
 cfe\_es\_eventids.h, [1100](#)  
**CFE\_ES\_CDS\_WRONG\_TYPE\_ERR**  
 cFE Return Code Defines, [224](#)  
**CFE\_ES\_CDSBLOCKID\_BASE**  
 cFE Resource ID base values, [398](#)  
**CFE\_ES\_CDSHANDLE\_C**  
 cfe\_es\_api\_typedefs.h, [993](#)  
**CFE\_ES\_CDCHandle\_t**  
 default\_cfe\_es\_extern\_typedefs.h, [1035](#)  
**CFE\_ES\_CDSRegDumpRec**, [533](#)  
 ByteAlignSpare, [533](#)  
 Handle, [533](#)  
 Name, [533](#)  
 Size, [533](#)  
 Table, [533](#)  
**CFE\_ES\_CDSRegDumpRec\_t**  
 default\_cfe\_es\_extern\_typedefs.h, [1035](#)  
**CFE\_ES\_ChildTaskMainFuncPtr\_t**  
 cfe\_es\_api\_typedefs.h, [995](#)  
**CFE\_ES\_CLEAR\_ER\_LOG\_CC**  
 default\_cfe\_es\_fcncodes.h, [1040](#)  
**CFE\_ES\_CLEAR\_SYSLOG\_CC**  
 default\_cfe\_es\_fcncodes.h, [1041](#)  
**CFE\_ES\_ClearERLogCmd\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_ClearSysLogCmd\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_CMD\_MID**  
 default\_cfe\_es\_msgids.h, [1084](#)  
**CFE\_ES\_CopyToCDS**  
 cFE Critical Data Store APIs, [270](#)  
**CFE\_ES\_COUNT\_SEM\_DELETE\_ERR**  
 cFE Return Code Defines, [225](#)  
**CFE\_ES\_COUNTERID\_C**  
 cfe\_es\_api\_typedefs.h, [993](#)  
**CFE\_ES\_CounterId\_t**  
 default\_cfe\_es\_extern\_typedefs.h, [1035](#)  
**CFE\_ES\_CounterID\_TolIndex**  
 cFE Resource ID APIs, [241](#)  
**CFE\_ES\_COUNTERID\_UNDEFINED**  
 cfe\_es\_api\_typedefs.h, [993](#)  
**CFE\_ES\_COUNTID\_BASE**  
 cFE Resource ID base values, [397](#)  
**CFE\_ES\_CrcType\_CRC\_16**  
 cfe\_es\_api\_typedefs.h, [996](#)  
**CFE\_ES\_CrcType\_CRC\_32**  
 cfe\_es\_api\_typedefs.h, [996](#)  
**CFE\_ES\_CrcType\_CRC\_8**  
 cfe\_es\_api\_typedefs.h, [996](#)  
**CFE\_ES\_CrcType\_Enum**  
 cfe\_es\_api\_typedefs.h, [996](#)  
**CFE\_ES\_CrcType\_Enum\_t**  
 cfe\_es\_api\_typedefs.h, [995](#)  
**CFE\_ES\_CreateChildTask**  
 cFE Child Task APIs, [262](#)  
**CFE\_ES\_CREATING\_CDS\_DUMP\_ERR\_EID**  
 cfe\_es\_eventids.h, [1100](#)  
**CFE\_ES\_DBIT**  
 cfe\_es.h, [991](#)  
**CFE\_ES\_DELETE\_CDS\_CC**  
 default\_cfe\_es\_fcncodes.h, [1042](#)  
**CFE\_ES\_DeleteApp**  
 cFE Application Control APIs, [246](#)  
**CFE\_ES\_DeleteCDSCmd**, [534](#)  
 CommandHeader, [534](#)  
 Payload, [534](#)  
**CFE\_ES\_DeleteCDSCmd\_Payload**, [534](#)  
 CdsName, [535](#)  
**CFE\_ES\_DeleteCDSCmd\_Payload\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_DeleteCDSCmd\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_DeleteChildTask**  
 cFE Child Task APIs, [263](#)  
**CFE\_ES\_DeleteGenCounter**  
 cFE Generic Counter APIs, [284](#)  
**CFE\_ES\_DTEST**  
 cfe\_es.h, [991](#)  
**CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC**  
 default\_cfe\_es\_fcncodes.h, [1042](#)  
**CFE\_ES\_DumpCDSRegistryCmd**, [535](#)  
 CommandHeader, [535](#)  
 Payload, [535](#)  
**CFE\_ES\_DumpCDSRegistryCmd\_Payload**, [535](#)  
 DumpFilename, [536](#)  
**CFE\_ES\_DumpCDSRegistryCmd\_Payload\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_DumpCDSRegistryCmd\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_ERLOG1\_INF\_EID**  
 cfe\_es\_eventids.h, [1100](#)  
**CFE\_ES\_ERLOG2\_EID**  
 cfe\_es\_eventids.h, [1101](#)  
**CFE\_ES\_ERLOG2\_ERR\_EID**  
 cfe\_es\_eventids.h, [1101](#)  
**CFE\_ES\_ERLOG\_PENDING\_ERR\_EID**  
 cfe\_es\_eventids.h, [1101](#)  
**CFE\_ES\_ERR\_APP\_CREATE**  
 cFE Return Code Defines, [225](#)  
**CFE\_ES\_ERR\_APP\_REGISTER**  
 cFE Return Code Defines, [225](#)  
**CFE\_ES\_ERR\_CHILD\_TASK\_CREATE**  
 cFE Return Code Defines, [225](#)  
**CFE\_ES\_ERR\_CHILD\_TASK\_DELETE**

cFE Return Code Defines, [225](#)  
CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK  
    cFE Return Code Defines, [225](#)  
CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER  
    cFE Return Code Defines, [225](#)  
CFE\_ES\_ERR\_DUPLICATE\_NAME  
    cFE Return Code Defines, [225](#)  
CFE\_ES\_ERR\_LOAD\_LIB  
    cFE Return Code Defines, [226](#)  
CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE  
    cFE Return Code Defines, [226](#)  
CFE\_ES\_ERR\_NAME\_NOT\_FOUND  
    cFE Return Code Defines, [226](#)  
CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID  
    cFE Return Code Defines, [226](#)  
CFE\_ES\_ERR\_SYS\_LOG\_FULL  
    cFE Return Code Defines, [226](#)  
CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED  
    cFE Return Code Defines, [226](#)  
CFE\_ES\_ERR\_SYSLOGMODE\_EID  
    `cfe_es_eventids.h`, [1101](#)  
CFE\_ES\_ERREXIT\_APP\_ERR\_EID  
    `cfe_es_eventids.h`, [1102](#)  
CFE\_ES\_ERREXIT\_APP\_INF\_EID  
    `cfe_es_eventids.h`, [1102](#)  
`cfe_es_eventids.h`  
    CFE\_ES\_ALL\_APPS\_EID, [1097](#)  
    CFE\_ES\_BOOT\_ERR\_EID, [1097](#)  
    CFE\_ES\_BUILD\_INF\_EID, [1098](#)  
    CFE\_ES\_CC1\_ERR\_EID, [1098](#)  
    CFE\_ES\_CDS\_DELETE\_ERR\_EID, [1098](#)  
    CFE\_ES\_CDS\_DELETE\_TBL\_ERR\_EID, [1098](#)  
    CFE\_ES\_CDS\_DELETED\_INFO\_EID, [1099](#)  
    CFE\_ES\_CDS\_DUMP\_ERR\_EID, [1099](#)  
    CFE\_ES\_CDS\_NAME\_ERR\_EID, [1099](#)  
    CFE\_ES\_CDS\_OWNER\_ACTIVE\_EID, [1099](#)  
    CFE\_ES\_CDS\_REG\_DUMP\_INF\_EID, [1100](#)  
    CFE\_ES\_CDS\_REGISTER\_ERR\_EID, [1100](#)  
    CFE\_ES\_CREATING\_CDS\_DUMP\_ERR\_EID, [1100](#)  
    CFE\_ES\_ERLOG1\_INF\_EID, [1100](#)  
    CFE\_ES\_ERLOG2\_EID, [1101](#)  
    CFE\_ES\_ERLOG2\_ERR\_EID, [1101](#)  
    CFE\_ES\_ERLOG\_PENDING\_ERR\_EID, [1101](#)  
    CFE\_ES\_ERR\_SYSLOGMODE\_EID, [1101](#)  
    CFE\_ES\_ERREXIT\_APP\_ERR\_EID, [1102](#)  
    CFE\_ES\_ERREXIT\_APP\_INF\_EID, [1102](#)  
    CFE\_ES\_EXIT\_APP\_ERR\_EID, [1102](#)  
    CFE\_ES\_EXIT\_APP\_INF\_EID, [1102](#)  
    CFE\_ES\_FILEWRITE\_ERR\_EID, [1103](#)  
    CFE\_ES\_INIT\_INF\_EID, [1103](#)  
    CFE\_ES\_INITSTATS\_INF\_EID, [1103](#)  
    CFE\_ES\_INVALID\_POOL\_HANDLE\_ERR\_EID,  
        [1103](#)  
    CFE\_ES\_LEN\_ERR\_EID, [1104](#)  
CFE\_ES\_MID\_ERR\_EID, [1104](#)  
CFE\_ES\_NOOP\_INF\_EID, [1104](#)  
CFE\_ES\_ONE\_APP\_EID, [1104](#)  
CFE\_ES\_ONE\_APPID\_ERR\_EID, [1105](#)  
CFE\_ES\_ONE\_ERR\_EID, [1105](#)  
CFE\_ES\_OSCREATE\_ERR\_EID, [1105](#)  
CFE\_ES\_PCR\_ERR1\_EID, [1105](#)  
CFE\_ES\_PCR\_ERR2\_EID, [1106](#)  
CFE\_ES\_PERF\_DATAWRITTEN\_EID, [1106](#)  
CFE\_ES\_PERF\_FILTMSKCMD\_EID, [1106](#)  
CFE\_ES\_PERF\_FILTMSKERR\_EID, [1106](#)  
CFE\_ES\_PERF\_LOG\_ERR\_EID, [1107](#)  
CFE\_ES\_PERF\_STARTCMD\_EID, [1107](#)  
CFE\_ES\_PERF\_STARTCMD\_ERR\_EID, [1107](#)  
CFE\_ES\_PERF\_STARTCMD\_TRIG\_ERR\_EID,  
    [1107](#)  
CFE\_ES\_PERF\_STOPCMD\_EID, [1108](#)  
CFE\_ES\_PERF\_STOPCMD\_ERR2\_EID, [1108](#)  
CFE\_ES\_PERF\_TRIGMSKCMD\_EID, [1108](#)  
CFE\_ES\_PERF\_TRIGMSKERR\_EID, [1108](#)  
CFE\_ES\_RELOAD\_APP\_DBG\_EID, [1109](#)  
CFE\_ES\_RELOAD\_APP\_ERR1\_EID, [1109](#)  
CFE\_ES\_RELOAD\_APP\_ERR2\_EID, [1109](#)  
CFE\_ES\_RELOAD\_APP\_ERR3\_EID, [1109](#)  
CFE\_ES\_RELOAD\_APP\_ERR4\_EID, [1110](#)  
CFE\_ES\_RELOAD\_APP\_INF\_EID, [1110](#)  
CFE\_ES\_RESET\_INF\_EID, [1110](#)  
CFE\_ES\_RESET\_PR\_COUNT\_EID, [1110](#)  
CFE\_ES\_RESTART\_APP\_DBG\_EID, [1111](#)  
CFE\_ES\_RESTART\_APP\_ERR1\_EID, [1111](#)  
CFE\_ES\_RESTART\_APP\_ERR2\_EID, [1111](#)  
CFE\_ES\_RESTART\_APP\_ERR3\_EID, [1111](#)  
CFE\_ES\_RESTART\_APP\_ERR4\_EID, [1112](#)  
CFE\_ES\_RESTART\_APP\_INF\_EID, [1112](#)  
CFE\_ES\_SET\_MAX\_PR\_COUNT\_EID, [1112](#)  
CFE\_ES\_START\_ERR\_EID, [1112](#)  
CFE\_ES\_START\_EXC\_ACTION\_ERR\_EID, [1113](#)  
CFE\_ES\_START\_INF\_EID, [1113](#)  
CFE\_ES\_START\_INVALID\_ENTRY\_POINT\_ERR\_EID,  
    [1113](#)  
CFE\_ES\_START\_INVALID\_FILENAME\_ERR\_EID,  
    [1113](#)  
CFE\_ES\_START\_NULL\_APP\_NAME\_ERR\_EID,  
    [1114](#)  
CFE\_ES\_START\_PRIORITY\_ERR\_EID, [1114](#)  
CFE\_ES\_STOP\_DBG\_EID, [1114](#)  
CFE\_ES\_STOP\_ERR1\_EID, [1114](#)  
CFE\_ES\_STOP\_ERR2\_EID, [1115](#)  
CFE\_ES\_STOP\_ERR3\_EID, [1115](#)  
CFE\_ES\_STOP\_INF\_EID, [1115](#)  
CFE\_ES\_SYSLOG1\_INF\_EID, [1115](#)  
CFE\_ES\_SYSLOG2\_EID, [1116](#)  
CFE\_ES\_SYSLOG2\_ERR\_EID, [1116](#)  
CFE\_ES\_SYSLOGMODE\_EID, [1116](#)

CFE\_ES\_TASKINFO\_EID, [1116](#)  
 CFE\_ES\_TASKINFO\_OSCREATE\_ERR\_EID, [1117](#)  
 CFE\_ES\_TASKINFO\_WR\_ERR\_EID, [1117](#)  
 CFE\_ES\_TASKINFO\_WRHDR\_ERR\_EID, [1117](#)  
 CFE\_ES\_TASKWR\_ERR\_EID, [1117](#)  
 CFE\_ES\_TLM\_POOL\_STATS\_INFO\_EID, [1118](#)  
 CFE\_ES\_VERSION\_INF\_EID, [1118](#)  
 CFE\_ES\_WRHDR\_ERR\_EID, [1118](#)  
 CFE\_ES\_WRITE\_CFE\_HDR\_ERR\_EID, [1118](#)  
**CFE\_ES\_ExceptionAction**  
 default\_cfe\_es\_extern\_typedefs.h, [1038](#)  
**CFE\_ES\_ExceptionAction\_Enum\_t**  
 default\_cfe\_es\_extern\_typedefs.h, [1035](#)  
**CFE\_ES\_ExceptionAction\_PROC\_RESTART**  
 default\_cfe\_es\_extern\_typedefs.h, [1038](#)  
**CFE\_ES\_ExceptionAction\_RESTART\_APP**  
 default\_cfe\_es\_extern\_typedefs.h, [1038](#)  
**CFE\_ES\_EXIT\_APP\_ERR\_EID**  
 cfe\_es\_eventids.h, [1102](#)  
**CFE\_ES\_EXIT\_APP\_INF\_EID**  
 cfe\_es\_eventids.h, [1102](#)  
**CFE\_ES\_ExitApp**  
 cFE Application Behavior APIs, [249](#)  
**CFE\_ES\_ExitChildTask**  
 cFE Child Task APIs, [264](#)  
**CFE\_ES\_FILE\_CLOSE\_ERR**  
 cFE Return Code Defines, [226](#)  
**CFE\_ES\_FILE\_IO\_ERR**  
 cFE Return Code Defines, [226](#)  
**CFE\_ES\_FileNameCmd**, [536](#)  
 CommandHeader, [536](#)  
 Payload, [536](#)  
**CFE\_ES\_FileNameCmd\_Payload**, [536](#)  
 FileName, [537](#)  
**CFE\_ES\_FileNameCmd\_Payload\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_FileNameCmd\_t**  
 default\_cfe\_es\_msgstruct.h, [1089](#)  
**CFE\_ES\_FILEWRITE\_ERR\_EID**  
 cfe\_es\_eventids.h, [1103](#)  
**CFE\_ES\_GetAppID**  
 cFE Information APIs, [253](#)  
**CFE\_ES\_GetAppIDByName**  
 cFE Information APIs, [254](#)  
**CFE\_ES\_GetAppInfo**  
 cFE Information APIs, [254](#)  
**CFE\_ES\_GetAppName**  
 cFE Information APIs, [255](#)  
**CFE\_ES\_GetCDSBlockIDByName**  
 cFE Critical Data Store APIs, [271](#)  
**CFE\_ES\_GetCDSBlockName**  
 cFE Critical Data Store APIs, [271](#)  
**CFE\_ES\_GetGenCount**  
 cFE Generic Counter APIs, [285](#)  
**CFE\_ES\_GetGenCounterIDByName**  
 cFE Generic Counter APIs, [285](#)  
**CFE\_ES\_GetGenCounterName**  
 cFE Generic Counter APIs, [286](#)  
**CFE\_ES\_GetLibIDByName**  
 cFE Information APIs, [256](#)  
**CFE\_ES\_GetLibInfo**  
 cFE Information APIs, [257](#)  
**CFE\_ES\_GetLibName**  
 cFE Information APIs, [257](#)  
**CFE\_ES\_GetMemPoolStats**  
 cFE Memory Manager APIs, [275](#)  
**CFE\_ES\_GetModuleInfo**  
 cFE Information APIs, [258](#)  
**CFE\_ES\_GetPoolBuf**  
 cFE Memory Manager APIs, [276](#)  
**CFE\_ES\_GetPoolBufInfo**  
 cFE Memory Manager APIs, [276](#)  
**CFE\_ES\_GetResetType**  
 cFE Information APIs, [259](#)  
**CFE\_ES\_GetTaskID**  
 cFE Information APIs, [260](#)  
**CFE\_ES\_GetTaskIDByName**  
 cFE Child Task APIs, [264](#)  
**CFE\_ES\_GetTaskInfo**  
 cFE Information APIs, [260](#)  
**CFE\_ES\_GetTaskName**  
 cFE Child Task APIs, [265](#)  
**CFE\_ES\_HK\_TLM\_MID**  
 default\_cfe\_es\_msgids.h, [1085](#)  
**CFE\_ES\_HousekeepingTlm**, [537](#)  
 Payload, [537](#)  
 TelemetryHeader, [537](#)  
**CFE\_ES\_HousekeepingTlm\_Payload**, [538](#)  
 BootSource, [540](#)  
 CFECOREChecksum, [540](#)  
 CFEMajorVersion, [540](#)  
 CFEMinorVersion, [540](#)  
 CFEMissionRevision, [540](#)  
 CFERevision, [540](#)  
 CommandCounter, [540](#)  
 CommandErrorCounter, [540](#)  
 ERLogEntries, [541](#)  
 ERLogIndex, [541](#)  
 HeapBlocksFree, [541](#)  
 HeapBytesFree, [541](#)  
 HeapMaxBlockSize, [541](#)  
 MaxProcessorResets, [541](#)  
 OSALMajorVersion, [541](#)  
 OSALMinorVersion, [542](#)  
 OSALMissionRevision, [542](#)  
 OSALRevision, [542](#)  
 PerfDataCount, [542](#)  
 PerfDataEnd, [542](#)

PerfDataStart, 542  
PerfDataToWrite, 542  
PerfFilterMask, 543  
PerfMode, 543  
PerfState, 543  
PerfTriggerCount, 543  
PerfTriggerMask, 543  
ProcessorResets, 543  
PSPMajorVersion, 543  
PSPMinorVersion, 544  
PSPMissionRevision, 544  
PSPRevision, 544  
RegisteredCoreApps, 544  
RegisteredExternalApps, 544  
RegisteredLibs, 544  
RegisteredTasks, 544  
ResetSubtype, 545  
ResetType, 545  
SysLogBytesUsed, 545  
SysLogEntries, 545  
SysLogMode, 545  
SysLogSize, 545  
CFE\_ES\_HousekeepingTlm\_Payload\_t  
    default\_cfe\_es\_msgstruct.h, 1089  
CFE\_ES\_HousekeepingTlm\_t  
    default\_cfe\_es\_msgstruct.h, 1089  
CFE\_ES\_IncrementGenCounter  
    cFE Generic Counter APIs, 287  
CFE\_ES\_IncrementTaskCounter  
    cFE Application Behavior APIs, 249  
CFE\_ES\_INIT\_INF\_EID  
    cfe\_es\_eventids.h, 1103  
CFE\_ES\_INITSTATS\_INF\_EID  
    cfe\_es\_eventids.h, 1103  
CFE\_ES\_INVALID\_POOL\_HANDLE\_ERR\_EID  
    cfe\_es\_eventids.h, 1103  
CFE\_ES\_LEN\_ERR\_EID  
    cfe\_es\_eventids.h, 1104  
CFE\_ES\_LIB\_ALREADY\_LOADED  
    cFE Return Code Defines, 227  
CFE\_ES\_LIBID\_BASE  
    cFE Resource ID base values, 397  
CFE\_ES\_LIBID\_C  
    cfe\_es\_api\_typedefs.h, 993  
CFE\_ES\_LibId\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1035  
CFE\_ES\_LibID\_TolIndex  
    cFE Resource ID APIs, 242  
CFE\_ES\_LIBID\_UNDEFINED  
    cfe\_es\_api\_typedefs.h, 993  
CFE\_ES\_LibraryEntryFuncPtr\_t  
    cfe\_es\_api\_typedefs.h, 995  
CFE\_ES\_LogEntryType  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
CFE\_ES\_LogEntryType\_APPLICATION  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
CFE\_ES\_LogEntryType\_CORE  
    default\_cfe\_es\_extern\_typedefs.h, 1038  
CFE\_ES\_LogEntryType\_Enum\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1036  
CFE\_ES\_LogMode  
    default\_cfe\_es\_extern\_typedefs.h, 1039  
CFE\_ES\_LogMode\_DISCARD  
    default\_cfe\_es\_extern\_typedefs.h, 1039  
CFE\_ES\_LogMode\_Enum\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1036  
CFE\_ES\_LogMode\_OVERWRITE  
    default\_cfe\_es\_extern\_typedefs.h, 1039  
CFE\_ES\_Main  
    cFE Entry/Exit APIs, 244  
CFE\_ES\_MEMADDRESS\_C  
    default\_cfe\_es\_extern\_typedefs.h, 1034  
CFE\_ES\_MemAddress\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1036  
CFE\_ES\_MEMADDRESS\_TO\_PTR  
    default\_cfe\_es\_extern\_typedefs.h, 1034  
CFE\_ES\_MEMHANDLE\_C  
    cfe\_es\_api\_typedefs.h, 994  
CFE\_ES\_MemHandle\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1036  
CFE\_ES\_MEMHANDLE\_UNDEFINED  
    cfe\_es\_api\_typedefs.h, 994  
CFE\_ES\_MEMOFFSET\_C  
    default\_cfe\_es\_extern\_typedefs.h, 1034  
CFE\_ES\_MemOffset\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1036  
CFE\_ES\_MEMOFFSET\_TO\_SIZE\_T  
    default\_cfe\_es\_extern\_typedefs.h, 1034  
CFE\_ES\_MEMPOOLBUF\_C  
    cfe\_es\_api\_typedefs.h, 994  
CFE\_ES\_MemPoolBuf\_t  
    cfe\_es\_api\_typedefs.h, 995  
CFE\_ES\_MemPoolStats, 546  
    BlockStats, 546  
    CheckErrCtr, 546  
    NumBlocksRequested, 546  
    NumFreeBytes, 546  
    PoolSize, 547  
CFE\_ES\_MemPoolStats\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1036  
CFE\_ES\_MEMSTATS\_TLM\_MID  
    default\_cfe\_es\_msgids.h, 1085  
CFE\_ES\_MemStatsTlm, 547  
    Payload, 547  
    TelemetryHeader, 547  
CFE\_ES\_MemStatsTlm\_t  
    default\_cfe\_es\_msgstruct.h, 1089  
CFE\_ES\_MID\_ERR\_EID

cfe\_es\_eventids.h, 1104  
**CFE\_ES\_MUT\_SEM\_DELETE\_ERR**  
 cFE Return Code Defines, 227  
**CFE\_ES\_NO\_MUTEX**  
 cfe\_es\_api\_typedefs.h, 994  
**CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE**  
 cFE Return Code Defines, 227  
**CFE\_ES\_NoArgsCmd**, 547  
 CommandHeader, 548  
**CFE\_ES\_NoArgsCmd\_t**  
 default\_cfe\_es\_msgstruct.h, 1089  
**CFE\_ES\_NOOP\_CC**  
 default\_cfe\_es\_fcncodes.h, 1043  
**CFE\_ES\_NOOP\_INF\_EID**  
 cfe\_es\_eventids.h, 1104  
**CFE\_ES\_NoopCmd\_t**  
 default\_cfe\_es\_msgstruct.h, 1090  
**CFE\_ES\_NOT\_IMPLEMENTED**  
 cFE Return Code Defines, 227  
**CFE\_ES\_ONE\_APP\_EID**  
 cfe\_es\_eventids.h, 1104  
**CFE\_ES\_ONE\_APPID\_ERR\_EID**  
 cfe\_es\_eventids.h, 1105  
**CFE\_ES\_ONE\_ERR\_EID**  
 cfe\_es\_eventids.h, 1105  
**CFE\_ES\_OneAppTlm**, 548  
 Payload, 548  
 TelemetryHeader, 548  
**CFE\_ES\_OneAppTlm\_Payload**, 549  
 AppInfo, 549  
**CFE\_ES\_OneAppTlm\_Payload\_t**  
 default\_cfe\_es\_msgstruct.h, 1090  
**CFE\_ES\_OneAppTlm\_t**  
 default\_cfe\_es\_msgstruct.h, 1090  
**CFE\_ES\_OPERATION\_TIMED\_OUT**  
 cFE Return Code Defines, 227  
**CFE\_ES\_OSCREATE\_ERR\_EID**  
 cfe\_es\_eventids.h, 1105  
**CFE\_ES\_OVER\_WRITE\_SYSLOG\_CC**  
 default\_cfe\_es\_fcncodes.h, 1044  
**CFE\_ES\_OverWriteSysLogCmd**, 549  
 CommandHeader, 549  
 Payload, 550  
**CFE\_ES\_OverWriteSysLogCmd\_Payload**, 550  
 Mode, 550  
**CFE\_ES\_OverWriteSysLogCmd\_Payload\_t**  
 default\_cfe\_es\_msgstruct.h, 1090  
**CFE\_ES\_OverWriteSysLogCmd\_t**  
 default\_cfe\_es\_msgstruct.h, 1090  
**CFE\_ES\_PCR\_ERR1\_EID**  
 cfe\_es\_eventids.h, 1105  
**CFE\_ES\_PCR\_ERR2\_EID**  
 cfe\_es\_eventids.h, 1106  
**CFE\_ES\_PERF\_DATAWRITTEN\_EID**  
 cfe\_es\_eventids.h, 1106  
**CFE\_ES\_PERF\_FILTMSKCMD\_EID**  
 cfe\_es\_eventids.h, 1106  
**CFE\_ES\_PERF\_FILTMSKERR\_EID**  
 cfe\_es\_eventids.h, 1106  
**CFE\_ES\_PERF\_LOG\_ERR\_EID**  
 cfe\_es\_eventids.h, 1107  
**CFE\_ES\_PERF\_STARTCMD\_EID**  
 cfe\_es\_eventids.h, 1107  
**CFE\_ES\_PERF\_STARTCMD\_ERR\_EID**  
 cfe\_es\_eventids.h, 1107  
**CFE\_ES\_PERF\_STARTCMD\_TRIG\_ERR\_EID**  
 cfe\_es\_eventids.h, 1107  
**CFE\_ES\_PERF\_STOPCMD\_EID**  
 cfe\_es\_eventids.h, 1108  
**CFE\_ES\_PERF\_STOPCMD\_ERR2\_EID**  
 cfe\_es\_eventids.h, 1108  
**CFE\_ES\_PERF\_TRIGMSKCMD\_EID**  
 cfe\_es\_eventids.h, 1108  
**CFE\_ES\_PERF\_TRIGMSKERR\_EID**  
 cfe\_es\_eventids.h, 1108  
**CFE\_ES\_PerfLogAdd**  
 cFE Performance Monitor APIs, 283  
**CFE\_ES\_PerfLogEntry**  
 cFE Performance Monitor APIs, 282  
**CFE\_ES\_PerfLogExit**  
 cFE Performance Monitor APIs, 282  
**CFE\_ES\_POOL\_BLOCK\_INVALID**  
 cFE Return Code Defines, 227  
**CFE\_ES\_PoolAlign**, 550  
 LongDouble, 551  
 LongInt, 551  
 Ptr, 551  
**CFE\_ES\_PoolAlign\_t**  
 cfe\_es\_api\_typedefs.h, 995  
**CFE\_ES\_PoolCreate**  
 cFE Memory Manager APIs, 277  
**CFE\_ES\_PoolCreateEx**  
 cFE Memory Manager APIs, 278  
**CFE\_ES\_PoolCreateNoSem**  
 cFE Memory Manager APIs, 279  
**CFE\_ES\_PoolDelete**  
 cFE Memory Manager APIs, 280  
**CFE\_ES\_POOLID\_BASE**  
 cFE Resource ID base values, 398  
**CFE\_ES\_PoolStatsTlm\_Payload**, 551  
 PoolHandle, 551  
 PoolStats, 551  
**CFE\_ES\_PoolStatsTlm\_Payload\_t**  
 default\_cfe\_es\_msgstruct.h, 1090  
**CFE\_ES\_ProcessAsyncEvent**  
 cFE Miscellaneous APIs, 268  
**CFE\_ES\_PutPoolBuf**  
 cFE Memory Manager APIs, 281

CFE\_ES\_QUERY\_ALL\_CC  
  default\_cfe\_es\_fcncodes.h, 1045

CFE\_ES\_QUERY\_ALL\_TASKS\_CC  
  default\_cfe\_es\_fcncodes.h, 1046

CFE\_ES\_QUERY\_ONE\_CC  
  default\_cfe\_es\_fcncodes.h, 1046

CFE\_ES\_QueryAllCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1090

CFE\_ES\_QueryAllTasksCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1090

CFE\_ES\_QueryOneCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1090

CFE\_ES\_QUEUE\_DELETE\_ERR  
  cFE Return Code Defines, 227

CFE\_ES\_RegisterCDS  
  cFE Critical Data Store APIs, 272

CFE\_ES\_RegisterGenCounter  
  cFE Generic Counter APIs, 287

CFE\_ES\_RELOAD\_APP\_CC  
  default\_cfe\_es\_fcncodes.h, 1047

CFE\_ES\_RELOAD\_APP\_DBG\_EID  
  cfe\_es\_eventids.h, 1109

CFE\_ES\_RELOAD\_APP\_ERR1\_EID  
  cfe\_es\_eventids.h, 1109

CFE\_ES\_RELOAD\_APP\_ERR2\_EID  
  cfe\_es\_eventids.h, 1109

CFE\_ES\_RELOAD\_APP\_ERR3\_EID  
  cfe\_es\_eventids.h, 1109

CFE\_ES\_RELOAD\_APP\_ERR4\_EID  
  cfe\_es\_eventids.h, 1110

CFE\_ES\_RELOAD\_APP\_INF\_EID  
  cfe\_es\_eventids.h, 1110

CFE\_ES\_ReloadApp  
  cFE Application Control APIs, 246

CFE\_ES\_ReloadAppCmd, 552  
  CommandHeader, 552  
  Payload, 552

CFE\_ES\_ReloadAppCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1090

CFE\_ES\_RESET\_COUNTERS\_CC  
  default\_cfe\_es\_fcncodes.h, 1048

CFE\_ES\_RESET\_INF\_EID  
  cfe\_es\_eventids.h, 1110

CFE\_ES\_RESET\_PR\_COUNT\_CC  
  default\_cfe\_es\_fcncodes.h, 1049

CFE\_ES\_RESET\_PR\_COUNT\_EID  
  cfe\_es\_eventids.h, 1110

CFE\_ES\_ResetCFE  
  cFE Entry/Exit APIs, 244

CFE\_ES\_ResetCountersCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1090

CFE\_ES\_ResetPRCountCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_RESTART\_APP\_CC  
  default\_cfe\_es\_fcncodes.h, 1049

CFE\_ES\_RESTART\_APP\_DBG\_EID  
  cfe\_es\_eventids.h, 1111

CFE\_ES\_RESTART\_APP\_ERR1\_EID  
  cfe\_es\_eventids.h, 1111

CFE\_ES\_RESTART\_APP\_ERR2\_EID  
  cfe\_es\_eventids.h, 1111

CFE\_ES\_RESTART\_APP\_ERR3\_EID  
  cfe\_es\_eventids.h, 1111

CFE\_ES\_RESTART\_APP\_ERR4\_EID  
  cfe\_es\_eventids.h, 1112

CFE\_ES\_RESTART\_APP\_INF\_EID  
  cfe\_es\_eventids.h, 1112

CFE\_ES\_RESTART\_CC  
  default\_cfe\_es\_fcncodes.h, 1050

CFE\_ES\_RestartApp  
  cFE Application Control APIs, 247

CFE\_ES\_RestartAppCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_RestartCmd, 552  
  CommandHeader, 553  
  Payload, 553

CFE\_ES\_RestartCmd\_Payload, 553  
  RestartType, 553

CFE\_ES\_RestartCmd\_Payload\_t  
  default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_RestartCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_RestoreFromCDS  
  cFE Critical Data Store APIs, 273

CFE\_ES\_RST\_ACCESS\_ERR  
  cFE Return Code Defines, 227

CFE\_ES\_RunLoop  
  cFE Application Behavior APIs, 250

CFE\_ES\_RunStatus  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_APP\_ERROR  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_APP\_EXIT  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_APP\_RUN  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_CORE\_APP\_INIT\_ERROR  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_CORE\_APP\_RUNTIME\_ERROR  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_Enum\_t  
  default\_cfe\_es\_extern\_typedefs.h, 1037

CFE\_ES\_RunStatus\_MAX  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_SYS\_DELETE  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_SYS\_EXCEPTION  
  default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_SYS\_RELOAD  
     default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_SYS\_RESTART  
     default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_RunStatus\_UNDEFINED  
     default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SEND\_HK\_MID  
     default\_cfe\_es\_msgids.h, 1085

CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC  
     default\_cfe\_es\_fcncodes.h, 1051

CFE\_ES\_SendHkCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SendMemPoolStatsCmd, 553  
     CommandHeader, 554  
     Payload, 554

CFE\_ES\_SendMemPoolStatsCmd\_Payload, 554  
     Application, 554  
     PoolHandle, 554

CFE\_ES\_SendMemPoolStatsCmd\_Payload\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SendMemPoolStatsCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC  
     default\_cfe\_es\_fcncodes.h, 1052

CFE\_ES\_SET\_MAX\_PR\_COUNT\_EID  
     cfe\_es\_eventids.h, 1112

CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC  
     default\_cfe\_es\_fcncodes.h, 1053

CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC  
     default\_cfe\_es\_fcncodes.h, 1054

CFE\_ES\_SetGenCount  
     cFE Generic Counter APIs, 288

CFE\_ES\_SetMaxPRCountCmd, 555  
     CommandHeader, 555  
     Payload, 555

CFE\_ES\_SetMaxPRCountCmd\_Payload, 555  
     MaxPRCount, 556

CFE\_ES\_SetMaxPRCountCmd\_Payload\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SetMaxPRCountCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SetPerfFilterMaskCmd, 556  
     CommandHeader, 556  
     Payload, 556

CFE\_ES\_SetPerfFilterMaskCmd\_Payload, 556  
     FilterMask, 557  
     FilterMaskNum, 557

CFE\_ES\_SetPerfFilterMaskCmd\_Payload\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SetPerfFilterMaskCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SetPerfTriggerMaskCmd, 557  
     CommandHeader, 557  
     Payload, 557

CFE\_ES\_SetPerfTriggerMaskCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1091

CFE\_ES\_SetPerfTrigMaskCmd\_Payload, 558  
     TriggerMask, 558  
     TriggerMaskNum, 558

CFE\_ES\_SetPerfTrigMaskCmd\_Payload\_t  
     default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_StackPointer\_t  
     cfe\_es\_api\_typedefs.h, 995

CFE\_ES\_START\_APP\_CC  
     default\_cfe\_es\_fcncodes.h, 1054

CFE\_ES\_START\_ERR\_EID  
     cfe\_es\_eventids.h, 1112

CFE\_ES\_START\_EXC\_ACTION\_ERR\_EID  
     cfe\_es\_eventids.h, 1113

CFE\_ES\_START\_INF\_EID  
     cfe\_es\_eventids.h, 1113

CFE\_ES\_START\_INVALID\_ENTRY\_POINT\_ERR\_EID  
     cfe\_es\_eventids.h, 1113

CFE\_ES\_START\_INVALID\_FILENAME\_ERR\_EID  
     cfe\_es\_eventids.h, 1113

CFE\_ES\_START\_NULL\_APP\_NAME\_ERR\_EID  
     cfe\_es\_eventids.h, 1114

CFE\_ES\_START\_PERF\_DATA\_CC  
     default\_cfe\_es\_fcncodes.h, 1055

CFE\_ES\_START\_PRIORITY\_ERR\_EID  
     cfe\_es\_eventids.h, 1114

CFE\_ES\_StartApp, 558  
     CommandHeader, 558  
     Payload, 559

CFE\_ES\_StartAppCmd\_Payload, 559  
     AppEntryPoint, 559  
     AppFileName, 559  
     Application, 559  
     ExceptionAction, 560  
     Priority, 560  
     StackSize, 560

CFE\_ES\_StartAppCmd\_Payload\_t  
     default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_StartAppCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_StartPerfCmd\_Payload, 560  
     TriggerMode, 560

CFE\_ES\_StartPerfCmd\_Payload\_t  
     default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_StartPerfDataCmd, 560  
     CommandHeader, 561  
     Payload, 561

CFE\_ES\_StartPerfDataCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_STATIC\_POOL\_TYPE  
     cfe\_es\_api\_typedefs.h, 994

CFE\_ES\_StatusToString  
     cfe\_error.h, 987

CFE\_ES\_STOP\_APP\_CC  
    default\_cfe\_es\_fcncodes.h, 1056

CFE\_ES\_STOP\_DBG\_EID  
    cfe\_es\_eventids.h, 1114

CFE\_ES\_STOP\_ERR1\_EID  
    cfe\_es\_eventids.h, 1114

CFE\_ES\_STOP\_ERR2\_EID  
    cfe\_es\_eventids.h, 1115

CFE\_ES\_STOP\_ERR3\_EID  
    cfe\_es\_eventids.h, 1115

CFE\_ES\_STOP\_INF\_EID  
    cfe\_es\_eventids.h, 1115

CFE\_ES\_STOP\_PERF\_DATA\_CC  
    default\_cfe\_es\_fcncodes.h, 1057

CFE\_ES\_StopAppCmd\_t  
    default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_StopPerfCmd\_Payload, 561  
    DataFileName, 561

CFE\_ES\_StopPerfCmd\_Payload\_t  
    default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_StopPerfDataCmd, 562  
    CommandHeader, 562  
    Payload, 562

CFE\_ES\_StopPerfDataCmd\_t  
    default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_SYSLOG1\_INF\_EID  
    cfe\_es\_eventids.h, 1115

CFE\_ES\_SYSLOG2\_EID  
    cfe\_es\_eventids.h, 1116

CFE\_ES\_SYSLOG2\_ERR\_EID  
    cfe\_es\_eventids.h, 1116

CFE\_ES\_SYSLOGMODE\_EID  
    cfe\_es\_eventids.h, 1116

CFE\_ES\_SystemState  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_APPS\_INIT  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_CORE\_READY  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_CORE\_STARTUP  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_EARLY\_INIT  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_Enum\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1037

CFE\_ES\_SystemState\_MAX  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_OPERATIONAL  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_SHUTDOWN  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_SystemState\_UNDEFINED  
    default\_cfe\_es\_extern\_typedefs.h, 1039

CFE\_ES\_TASK\_DELETE\_ERR  
    cFE Return Code Defines, 227

CFE\_ES\_TASK\_STACK\_ALLOCATE  
    cfe\_es\_api\_typedefs.h, 994

CFE\_ES\_TaskEntryFuncPtr\_t  
    cfe\_es\_api\_typedefs.h, 996

CFE\_ES\_TASKID\_BASE  
    cFE Resource ID base values, 397

CFE\_ES\_TASKID\_C  
    cfe\_es\_api\_typedefs.h, 994

CFE\_ES\_TaskId\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1037

CFE\_ES\_TaskID\_ToIndex  
    cFE Resource ID APIs, 243

CFE\_ES\_TASKID\_UNDEFINED  
    cfe\_es\_api\_typedefs.h, 995

CFE\_ES\_TaskInfo, 562  
    ApplId, 563  
    AppName, 563  
    ExecutionCounter, 563  
    Priority, 563  
    Spare, 563  
    StackSize, 563  
    TaskId, 563  
    TaskName, 563

CFE\_ES\_TASKINFO\_EID  
    cfe\_es\_eventids.h, 1116

CFE\_ES\_TASKINFO\_OSCREATE\_ERR\_EID  
    cfe\_es\_eventids.h, 1117

CFE\_ES\_TaskInfo\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1037

CFE\_ES\_TASKINFO\_WR\_ERR\_EID  
    cfe\_es\_eventids.h, 1117

CFE\_ES\_TASKINFO\_WRHDR\_ERR\_EID  
    cfe\_es\_eventids.h, 1117

CFE\_ES\_TaskPriority\_Atom\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1037

CFE\_ES\_TASKWR\_ERR\_EID  
    cfe\_es\_eventids.h, 1117

CFE\_ES\_TEST\_LONG\_MASK  
    cfe\_es.h, 991

CFE\_ES\_TIMER\_DELETE\_ERR  
    cFE Return Code Defines, 228

CFE\_ES\_TLM\_POOL\_STATS\_INFO\_EID  
    cfe\_es\_eventids.h, 1118

CFE\_ES\_USE\_MUTEX  
    cfe\_es\_api\_typedefs.h, 995

CFE\_ES\_VERSION\_INF\_EID  
    cfe\_es\_eventids.h, 1118

CFE\_ES\_WaitForStartupSync  
    cFE Application Behavior APIs, 251

CFE\_ES\_WaitForSystemState  
    cFE Application Behavior APIs, 251

CFE\_ES\_WRHDR\_ERR\_EID  
    cfe\_es\_eventids.h, 1118

CFE\_ES\_WRITE\_CFE\_HDR\_ERR\_EID  
     cfe\_es\_eventids.h, 1118

CFE\_ES\_WRITE\_ER\_LOG\_CC  
     default\_cfe\_es\_fcncodes.h, 1058

CFE\_ES\_WRITE\_SYSLOG\_CC  
     default\_cfe\_es\_fcncodes.h, 1059

CFE\_ES\_WriteERLogCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_WriteSysLogCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1092

CFE\_ES\_WriteToSysLog  
     cFE Miscellaneous APIs, 268

CFE\_EVENTS\_SERVICE  
     cfe\_error.h, 986

cfe\_evs.h  
     CFE\_EVS\_Send, 997  
     CFE\_EVS\_SendCrit, 997  
     CFE\_EVS\_SendDbg, 997  
     CFE\_EVS\_SendErr, 997  
     CFE\_EVS\_SendInfo, 997

CFE\_EVS\_ADD\_EVENT\_FILTER\_CC  
     default\_cfe\_evs\_fcncodes.h, 1122

CFE\_EVS\_AddEventFilterCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_ADDFILTER\_EID  
     cfe\_evs\_eventids.h, 1157

cfe\_evs\_api\_typedefs.h  
     CFE\_EVS\_BinFilter\_t, 1000  
     CFE\_EVS\_EVERY\_FOURTH\_ONE, 998  
     CFE\_EVS\_EVERY\_OTHER\_ONE, 999  
     CFE\_EVS\_EVERY\_OTHER\_TWO, 999  
     CFE\_EVS\_FIRST\_16\_STOP, 999  
     CFE\_EVS\_FIRST\_32\_STOP, 999  
     CFE\_EVS\_FIRST\_4\_STOP, 999  
     CFE\_EVS\_FIRST\_64\_STOP, 999  
     CFE\_EVS\_FIRST\_8\_STOP, 999  
     CFE\_EVS\_FIRST\_ONE\_STOP, 999  
     CFE\_EVS\_FIRST\_TWO\_STOP, 999  
     CFE\_EVS\_NO\_FILTER, 999

CFE\_EVS\_APP\_FILTER\_OVERLOAD  
     cFE Return Code Defines, 228

CFE\_EVS\_APP\_ILLEGAL\_APP\_ID  
     cFE Return Code Defines, 228

CFE\_EVS\_APP\_NOT\_REGISTERED  
     cFE Return Code Defines, 228

CFE\_EVS\_APP\_SQUELCHED  
     cFE Return Code Defines, 228

CFE\_EVS\_AppDataCmd\_Payload, 564  
     AppDataFilename, 564

CFE\_EVS\_AppDataCmd\_Payload\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameBitMaskCmd, 564  
     CommandHeader, 564  
     Payload, 565

CFE\_EVS\_AppNameBitMaskCmd\_Payload, 565  
     AppName, 565  
     BitMask, 565  
     Spare, 565

CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameBitMaskCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameCmd, 566  
     CommandHeader, 566  
     Payload, 566

CFE\_EVS\_AppNameCmd\_Payload, 566  
     AppName, 566

CFE\_EVS\_AppNameCmd\_Payload\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameEventIDCmd, 567  
     CommandHeader, 567  
     Payload, 567

CFE\_EVS\_AppNameEventIDCmd\_Payload, 567  
     AppName, 568  
     EventID, 568

CFE\_EVS\_AppNameEventIDCmd\_Payload\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameEventIDCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameEventIDMaskCmd, 568  
     CommandHeader, 568  
     Payload, 568

CFE\_EVS\_AppNameEventIDMaskCmd\_Payload, 569  
     AppName, 569  
     EventID, 569  
     Mask, 569

CFE\_EVS\_AppNameEventIDMaskCmd\_Payload\_t  
     default\_cfe\_evs\_msgstruct.h, 1150

CFE\_EVS\_AppNameEventIDMaskCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1151

CFE\_EVS\_AppTlmData, 569  
     AppEnableStatus, 570  
     AppID, 570  
     AppMessageSentCounter, 570  
     AppMessageSquelchedCounter, 570

CFE\_EVS\_AppTlmData\_t  
     default\_cfe\_evs\_msgstruct.h, 1151

CFE\_EVS\_BinFilter, 570  
     EventID, 571  
     Mask, 571

CFE\_EVS\_BinFilter\_t  
     cfe\_evs\_api\_typedefs.h, 1000

CFE\_EVS\_BitMaskCmd, 571  
     CommandHeader, 571  
     Payload, 571

CFE\_EVS\_BitMaskCmd\_Payload, 572

BitMask, 572  
Spare, 572  
CFE\_EVS\_BitMaskCmd\_Payload\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_BitMaskCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_CLEAR\_LOG\_CC  
  default\_cfe\_evs\_fcncodes.h, 1123  
CFE\_EVS\_ClearLogCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_CMD\_MID  
  default\_cfe\_evs\_msgids.h, 1147  
CFE\_EVS\_CRITICAL\_BIT  
  default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_DEBUG\_BIT  
  default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC  
  default\_cfe\_evs\_fcncodes.h, 1124  
CFE\_EVS\_DeleteEventFilterCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_DELFILTER\_EID  
  cfe\_evs\_eventids.h, 1157  
CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC  
  default\_cfe\_evs\_fcncodes.h, 1124  
CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC  
  default\_cfe\_evs\_fcncodes.h, 1125  
CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC  
  default\_cfe\_evs\_fcncodes.h, 1126  
CFE\_EVS\_DISABLE\_PORTS\_CC  
  default\_cfe\_evs\_fcncodes.h, 1127  
CFE\_EVS\_DisableAppEventsCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_DisableAppEventTypeCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_DisablePortsCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_DISAPPENTTYPE\_EID  
  cfe\_evs\_eventids.h, 1157  
CFE\_EVS\_DISAPPEVT\_EID  
  cfe\_evs\_eventids.h, 1157  
CFE\_EVS\_DISEVTTYPE\_EID  
  cfe\_evs\_eventids.h, 1158  
CFE\_EVS\_DISPRT\_EID  
  cfe\_evs\_eventids.h, 1158  
CFE\_EVS\_ENAAPPEVT\_EID  
  cfe\_evs\_eventids.h, 1158  
CFE\_EVS\_ENAAPPEVTTYPE\_EID  
  cfe\_evs\_eventids.h, 1158  
CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC  
  default\_cfe\_evs\_fcncodes.h, 1128  
CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC  
  default\_cfe\_evs\_fcncodes.h, 1129  
CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC  
  default\_cfe\_evs\_fcncodes.h, 1130  
CFE\_EVS\_ENABLE\_PORTS\_CC  
  default\_cfe\_evs\_fcncodes.h, 1131  
CFE\_EVS\_EnableAppEventsCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_EnableAppEventTypeCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1151  
CFE\_EVS\_EnableEventTypeCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_EnablePortsCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_ENAEVTTYPE\_EID  
  cfe\_evs\_eventids.h, 1159  
CFE\_EVS\_ENAPORT\_EID  
  cfe\_evs\_eventids.h, 1159  
CFE\_EVS\_ERR\_APPNOREGS\_EID  
  cfe\_evs\_eventids.h, 1159  
CFE\_EVS\_ERR\_CC\_EID  
  cfe\_evs\_eventids.h, 1159  
CFE\_EVS\_ERR\_CRDATFILE\_EID  
  cfe\_evs\_eventids.h, 1160  
CFE\_EVS\_ERR\_CRLOGFILE\_EID  
  cfe\_evs\_eventids.h, 1160  
CFE\_EVS\_ERR\_EVTIDNOREGS\_EID  
  cfe\_evs\_eventids.h, 1160  
CFE\_EVS\_ERR\_ILLAPPIDRANGE\_EID  
  cfe\_evs\_eventids.h, 1160  
CFE\_EVS\_ERR\_ILLEGALFMTMOD\_EID  
  cfe\_evs\_eventids.h, 1161  
CFE\_EVS\_ERR\_INVALID\_BITMASK\_EID  
  cfe\_evs\_eventids.h, 1161  
CFE\_EVS\_ERR\_LOGMODE\_EID  
  cfe\_evs\_eventids.h, 1161  
CFE\_EVS\_ERR\_MAXREGSFILTER\_EID  
  cfe\_evs\_eventids.h, 1161  
CFE\_EVS\_ERR\_MSGID\_EID  
  cfe\_evs\_eventids.h, 1162  
CFE\_EVS\_ERR\_NOAPPIDFOUND\_EID  
  cfe\_evs\_eventids.h, 1162  
CFE\_EVS\_ERR\_UNREGISTERED\_EVS\_APP  
  cfe\_evs\_eventids.h, 1162  
CFE\_EVS\_ERR\_WRDATFILE\_EID  
  cfe\_evs\_eventids.h, 1162  
CFE\_EVS\_ERR\_WRLogFile\_EID  
  cfe\_evs\_eventids.h, 1163  
CFE\_EVS\_ERROR\_BIT  
  default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_EventFilter  
  default\_cfe\_evs\_extern\_typedefs.h, 1120  
CFE\_EVS\_EventFilter\_BINARY  
  default\_cfe\_evs\_extern\_typedefs.h, 1120  
CFE\_EVS\_EventFilter\_Enum\_t  
  default\_cfe\_evs\_extern\_typedefs.h, 1119

cfe\_evs\_eventids.h  
 CFE\_EVS\_ADDFILTER\_EID, 1157  
 CFE\_EVS\_DELFILTER\_EID, 1157  
 CFE\_EVS\_DISAPPENTTYPE\_EID, 1157  
 CFE\_EVS\_DISAPPEVT\_EID, 1157  
 CFE\_EVS\_DISEVTTYPE\_EID, 1158  
 CFE\_EVS\_DISPRT\_EID, 1158  
 CFE\_EVS\_ENAAPPEVT\_EID, 1158  
 CFE\_EVS\_ENAAPPEVTTYPE\_EID, 1158  
 CFE\_EVS\_ENAEVTTYPE\_EID, 1159  
 CFE\_EVS\_ENAPORT\_EID, 1159  
 CFE\_EVS\_ERR\_APPNOREGS\_EID, 1159  
 CFE\_EVS\_ERR\_CC\_EID, 1159  
 CFE\_EVS\_ERR\_CRDATFILE\_EID, 1160  
 CFE\_EVS\_ERR\_CRLOGFILE\_EID, 1160  
 CFE\_EVS\_ERR\_EVTIDNOREGS\_EID, 1160  
 CFE\_EVS\_ERR\_ILLAPPIDRANGE\_EID, 1160  
 CFE\_EVS\_ERR\_ILLEGALFMTMOD\_EID, 1161  
 CFE\_EVS\_ERR\_INVALID\_BITMASK\_EID, 1161  
 CFE\_EVS\_ERR\_LOGMODE\_EID, 1161  
 CFE\_EVS\_ERR\_MAXREGSFILTER\_EID, 1161  
 CFE\_EVS\_ERR\_MSGID\_EID, 1162  
 CFE\_EVS\_ERR\_NOAPPIDFOUND\_EID, 1162  
 CFE\_EVS\_ERR\_UNREGISTERED\_EVS\_APP,  
     1162  
 CFE\_EVS\_ERR\_WRDATFILE\_EID, 1162  
 CFE\_EVS\_ERR\_WRLOGFILE\_EID, 1163  
 CFE\_EVS\_EVT\_FILTERED\_EID, 1163  
 CFE\_EVS\_FILTER\_MAX\_EID, 1163  
 CFE\_EVS\_LEN\_ERR\_EID, 1163  
 CFE\_EVS\_LOGMODE\_EID, 1164  
 CFE\_EVS\_NOOP\_EID, 1164  
 CFE\_EVS\_RSTALLFILTER\_EID, 1164  
 CFE\_EVS\_RSTCNT\_EID, 1164  
 CFE\_EVS\_RSTEVCNT\_EID, 1165  
 CFE\_EVS\_RSTFILTER\_EID, 1165  
 CFE\_EVS\_SETEVTFMTMOD\_EID, 1165  
 CFE\_EVS\_SETFILTERMSK\_EID, 1165  
 CFE\_EVS\_SQUELCHED\_ERR\_EID, 1166  
 CFE\_EVS\_STARTUP\_EID, 1166  
 CFE\_EVS\_WRDAT\_EID, 1166  
 CFE\_EVS\_WRITE\_HEADER\_ERR\_EID, 1166  
 CFE\_EVS\_WRLOG\_EID, 1167  
 CFE\_EVS\_EventOutput  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventOutput\_Enum\_t  
     default\_cfe\_evs\_extern\_typedefs.h, 1120  
 CFE\_EVS\_EventOutput\_PORT1  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventOutput\_PORT2  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventOutput\_PORT3  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventOutput\_PORT4  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventType  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventType\_CRITICAL  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventType\_DEBUG  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventType\_Enum\_t  
     default\_cfe\_evs\_extern\_typedefs.h, 1120  
 CFE\_EVS\_EventType\_ERROR  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EventType\_INFORMATION  
     default\_cfe\_evs\_extern\_typedefs.h, 1121  
 CFE\_EVS\_EVERY\_FOURTH\_ONE  
     cfe\_evs\_api\_typedefs.h, 998  
 CFE\_EVS\_EVERY\_OTHER\_ONE  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_EVERY\_OTHER\_TWO  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_EVT\_FILTERED\_EID  
     cfe\_evs\_eventids.h, 1163  
 CFE\_EVS\_EVT\_NOT\_REGISTERED  
     cFE Return Code Defines, 228  
 CFE\_EVS\_FILE\_WRITE\_ERROR  
     cFE Return Code Defines, 228  
 CFE\_EVS\_FILTER\_MAX\_EID  
     cfe\_evs\_eventids.h, 1163  
 CFE\_EVS\_FIRST\_16\_STOP  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_FIRST\_32\_STOP  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_FIRST\_4\_STOP  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_FIRST\_64\_STOP  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_FIRST\_8\_STOP  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_FIRST\_ONE\_STOP  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_FIRST\_TWO\_STOP  
     cfe\_evs\_api\_typedefs.h, 999  
 CFE\_EVS\_HK\_TLM\_MID  
     default\_cfe\_evs\_msgids.h, 1147  
 CFE\_EVS\_HousekeepingTlm, 572  
     Payload, 572  
     TelemetryHeader, 573  
 CFE\_EVS\_HousekeepingTlm\_Payload, 573  
     AppData, 574  
     CommandCounter, 574  
     CommandErrorCounter, 574  
     LogEnabled, 574  
     LogFullFlag, 574  
     LogMode, 574  
     LogOverflowCounter, 574

MessageFormatMode, 575  
MessageSendCounter, 575  
MessageTruncCounter, 575  
OutputPort, 575  
Spare1, 575  
Spare2, 575  
Spare3, 575  
UnregisteredAppCounter, 576  
CFE\_EVS\_HousekeepingTlm\_Payload\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_HousekeepingTlm\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_INFORMATION\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_INVALID\_PARAMETER  
    cFE Return Code Defines, 228  
CFE\_EVS\_LEN\_ERR\_EID  
    cfe\_evs\_eventids.h, 1163  
CFE\_EVS\_LogFileCmd\_Payload, 576  
    LogFilename, 576  
CFE\_EVS\_LogFileCmd\_Payload\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_LogMode  
    default\_cfe\_evs\_extern\_typedefs.h, 1121  
CFE\_EVS\_LogMode\_DISCARD  
    default\_cfe\_evs\_extern\_typedefs.h, 1121  
CFE\_EVS\_LOGMODE\_EID  
    cfe\_evs\_eventids.h, 1164  
CFE\_EVS\_LogMode\_Enum\_t  
    default\_cfe\_evs\_extern\_typedefs.h, 1120  
CFE\_EVS\_LogMode\_OVERWRITE  
    default\_cfe\_evs\_extern\_typedefs.h, 1121  
CFE\_EVS\_LONG\_EVENT\_MSG\_MID  
    default\_cfe\_evs\_msgids.h, 1147  
CFE\_EVS\_LongEventTlm, 576  
    Payload, 577  
    TelemetryHeader, 577  
CFE\_EVS\_LongEventTlm\_Payload, 577  
    Message, 577  
    PacketID, 577  
    Spare1, 578  
    Spare2, 578  
CFE\_EVS\_LongEventTlm\_Payload\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_LongEventTlm\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_MsgFormat  
    default\_cfe\_evs\_extern\_typedefs.h, 1121  
CFE\_EVS\_MsgFormat\_Enum\_t  
    default\_cfe\_evs\_extern\_typedefs.h, 1120  
CFE\_EVS\_MsgFormat\_LONG  
    default\_cfe\_evs\_extern\_typedefs.h, 1121  
CFE\_EVS\_MsgFormat\_SHORT  
    default\_cfe\_evs\_extern\_typedefs.h, 1121  
CFE\_EVS\_NO\_FILTER  
    cfe\_evs\_api\_typedefs.h, 999  
CFE\_EVS\_NoArgsCmd, 578  
    CommandHeader, 578  
CFE\_EVS\_NoArgsCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_NOOP\_CC  
    default\_cfe\_evs\_fcncodes.h, 1132  
CFE\_EVS\_NOOP\_EID  
    cfe\_evs\_eventids.h, 1164  
CFE\_EVS\_NoopCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_NOT\_IMPLEMENTED  
    cFE Return Code Defines, 229  
CFE\_EVS\_PacketID, 578  
    AppName, 579  
    EventID, 579  
    EventType, 579  
    ProcessorID, 579  
    SpacecraftID, 579  
CFE\_EVS\_PacketID\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_PORT1\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_PORT2\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_PORT3\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_PORT4\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1146  
CFE\_EVS\_Register  
    cFE Registration APIs, 290  
CFE\_EVS\_RESET\_ALL\_FILTERS\_CC  
    default\_cfe\_evs\_fcncodes.h, 1132  
CFE\_EVS\_RESET\_APP\_COUNTER\_CC  
    default\_cfe\_evs\_fcncodes.h, 1133  
CFE\_EVS\_RESET\_AREA\_POINTER  
    cFE Return Code Defines, 229  
CFE\_EVS\_RESET\_COUNTERS\_CC  
    default\_cfe\_evs\_fcncodes.h, 1134  
CFE\_EVS\_RESET\_FILTER\_CC  
    default\_cfe\_evs\_fcncodes.h, 1135  
CFE\_EVS\_ResetAllFilters  
    cFE Reset Event Filter APIs, 297  
CFE\_EVS\_ResetAllFiltersCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_ResetAppCounterCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_ResetCountersCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1152  
CFE\_EVS\_ResetFilter  
    cFE Reset Event Filter APIs, 297  
CFE\_EVS\_ResetFilterCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1153

CFE\_EVS\_RSTALLFILTER\_EID  
     cfe\_evs\_eventids.h, 1164

CFE\_EVS\_RSTCNT\_EID  
     cfe\_evs\_eventids.h, 1164

CFE\_EVS\_RSTEVCNT\_EID  
     cfe\_evs\_eventids.h, 1165

CFE\_EVS\_RSTFILTER\_EID  
     cfe\_evs\_eventids.h, 1165

CFE\_EVS\_Send  
     cfe\_evs.h, 997

CFE\_EVS\_SEND\_HK\_MID  
     default\_cfe\_evs\_msgids.h, 1147

CFE\_EVS\_SendCrit  
     cfe\_evs.h, 997

CFE\_EVS\_SendDbg  
     cfe\_evs.h, 997

CFE\_EVS\_SendErr  
     cfe\_evs.h, 997

CFE\_EVS\_SendEvent  
     cFE Send Event APIs, 292

CFE\_EVS\_SendEventWithAppID  
     cFE Send Event APIs, 293

CFE\_EVS\_SendHkCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1153

CFE\_EVS\_SendInfo  
     cfe\_evs.h, 997

CFE\_EVS\_SendTimedEvent  
     cFE Send Event APIs, 295

CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC  
     default\_cfe\_evs\_fcncodes.h, 1135

CFE\_EVS\_SET\_FILTER\_CC  
     default\_cfe\_evs\_fcncodes.h, 1136

CFE\_EVS\_SET\_LOG\_MODE\_CC  
     default\_cfe\_evs\_fcncodes.h, 1137

CFE\_EVS\_SetEventFormatCode\_Payload, 580  
     MsgFormat, 580  
     Spare, 580

CFE\_EVS\_SetEventFormatMode\_Payload\_t  
     default\_cfe\_evs\_msgstruct.h, 1153

CFE\_EVS\_SetEventFormatModeCmd, 580  
     CommandHeader, 581  
     Payload, 581

CFE\_EVS\_SetEventFormatModeCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1153

CFE\_EVS\_SETEVTFMTMOD\_EID  
     cfe\_evs\_eventids.h, 1165

CFE\_EVS\_SetFilterCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1153

CFE\_EVS\_SETFILTERMSK\_EID  
     cfe\_evs\_eventids.h, 1165

CFE\_EVS\_SetLogMode\_Payload, 581  
     LogMode, 581  
     Spare, 581

CFE\_EVS\_SetLogMode\_Payload\_t

                default\_cfe\_evs\_msgstruct.h, 1153

                CFE\_EVS\_SetLogModeCmd, 582  
                     CommandHeader, 582  
                     Payload, 582

                CFE\_EVS\_SetLogModeCmd\_t  
                     default\_cfe\_evs\_msgstruct.h, 1153

                CFE\_EVS\_SHORT\_EVENT\_MSG\_MID  
                     default\_cfe\_evs\_msgids.h, 1147

                CFE\_EVS\_ShortEventTlm, 582  
                     Payload, 582  
                     TelemetryHeader, 583

                CFE\_EVS\_ShortEventTlm\_Payload, 583  
                     PacketID, 583

                CFE\_EVS\_ShortEventTlm\_Payload\_t  
                     default\_cfe\_evs\_msgstruct.h, 1153

                CFE\_EVS\_ShortEventTlm\_t  
                     default\_cfe\_evs\_msgstruct.h, 1153

                CFE\_EVS\_SQUELCHED\_ERR\_EID  
                     cfe\_evs\_eventids.h, 1166

                CFE\_EVS\_STARTUP\_EID  
                     cfe\_evs\_eventids.h, 1166

                CFE\_EVS\_UNKNOWN\_FILTER  
                     cFE Return Code Defines, 229

                CFE\_EVS\_WRDAT\_EID  
                     cfe\_evs\_eventids.h, 1166

                CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC  
                     default\_cfe\_evs\_fcncodes.h, 1138

                CFE\_EVS\_WRITE\_HEADER\_ERR\_EID  
                     cfe\_evs\_eventids.h, 1166

                CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC  
                     default\_cfe\_evs\_fcncodes.h, 1139

                CFE\_EVS\_WriteAppDataFileCmd, 583  
                     CommandHeader, 584  
                     Payload, 584

                CFE\_EVS\_WriteAppDataFileCmd\_t  
                     default\_cfe\_evs\_msgstruct.h, 1153

                CFE\_EVS\_WriteLogFileCmd, 584  
                     CommandHeader, 584  
                     Payload, 584

                CFE\_EVS\_WriteLogFileCmd\_t  
                     default\_cfe\_evs\_msgstruct.h, 1153

                CFE\_EVS\_WRLOG\_EID  
                     cfe\_evs\_eventids.h, 1167

                CFE\_EXECUTIVE\_SERVICE  
                     cfe\_error.h, 986

                CFE\_FILE\_SERVICE  
                     cfe\_error.h, 986

                cfe\_fs\_api\_typedefs.h  
                     CFE\_FS\_FileCategory\_BINARY\_DATA\_DUMP,  
                         1003  
                     CFE\_FS\_FileCategory\_DYNAMIC\_MODULE, 1003  
                     CFE\_FS\_FileCategory\_MAX, 1003  
                     CFE\_FS\_FileCategory\_SCRIPT, 1003  
                     CFE\_FS\_FileCategory\_t, 1002

CFE\_FS\_FileCategory\_TEMP, [1003](#)  
CFE\_FS\_FileCategory\_TEXT\_LOG, [1003](#)  
CFE\_FS\_FileCategory\_UNKNOWN, [1003](#)  
CFE\_FS\_FileWriteEvent\_COMPLETE, [1003](#)  
CFE\_FS\_FileWriteEvent\_CREATE\_ERROR, [1003](#)  
CFE\_FS\_FileWriteEvent\_HEADER\_WRITE\_ERROR, [1003](#)  
CFE\_FS\_FileWriteEvent\_MAX, [1003](#)  
CFE\_FS\_FileWriteEvent\_RECORD\_WRITE\_ERROR, [1003](#)  
CFE\_FS\_FileWriteEvent\_t, [1003](#)  
CFE\_FS\_FileWriteEvent\_UNDEFINED, [1003](#)  
CFE\_FS\_FileWriteGetData\_t, [1001](#)  
CFE\_FS\_FileWriteMetaData\_t, [1002](#)  
CFE\_FS\_FileWriteOnEvent\_t, [1002](#)  
CFE\_FS\_BackgroundFileDumpIsPending  
    cFE File Utility APIs, [303](#)  
CFE\_FS\_BackgroundFileDumpRequest  
    cFE File Utility APIs, [304](#)  
CFE\_FS\_BAD\_ARGUMENT  
    cFE Return Code Defines, [229](#)  
CFE\_FS\_ExtractFilenameFromPath  
    cFE File Utility APIs, [304](#)  
CFE\_FS\_FILE\_CONTENT\_ID  
    default\_cfe\_fs\_interface\_cfg.h, [1169](#)  
    example\_mission\_cfg.h, [915](#)  
CFE\_FS\_FileCategory\_BINARY\_DATA\_DUMP  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileCategory\_DYNAMIC\_MODULE  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileCategory\_MAX  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileCategory\_SCRIPT  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileCategory\_t  
    cfe\_fs\_api\_typedefs.h, [1002](#)  
CFE\_FS\_FileCategory\_TEMP  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileCategory\_TEXT\_LOG  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileCategory\_UNKNOWN  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileWriteEvent\_COMPLETE  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileWriteEvent\_CREATE\_ERROR  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileWriteEvent\_HEADER\_WRITE\_ERROR  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileWriteEvent\_MAX  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileWriteEvent\_RECORD\_WRITE\_ERROR  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileWriteEvent\_t  
    cfe\_fs\_api\_typedefs.h, [1003](#)

CFE\_FS\_FileWriteEvent\_UNDEFINED  
    cfe\_fs\_api\_typedefs.h, [1003](#)  
CFE\_FS\_FileWriteGetData\_t  
    cfe\_fs\_api\_typedefs.h, [1001](#)  
CFE\_FS\_FileWriteMetaData, [584](#)  
    Description, [585](#)  
    FileName, [585](#)  
    FileSubType, [585](#)  
    GetData, [585](#)  
    IsPending, [585](#)  
    OnEvent, [585](#)  
CFE\_FS\_FileWriteMetaData\_t  
    cfe\_fs\_api\_typedefs.h, [1002](#)  
CFE\_FS\_FileWriteOnEvent\_t  
    cfe\_fs\_api\_typedefs.h, [1002](#)  
CFE\_FS\_FNAME\_TOO\_LONG  
    cFE Return Code Defines, [229](#)  
CFE\_FS\_GetDefaultExtension  
    cFE File Utility APIs, [305](#)  
CFE\_FS\_GetDefaultMountPoint  
    cFE File Utility APIs, [305](#)  
CFE\_FS\_HDR\_DESC\_MAX\_LEN  
    default\_cfe\_fs\_interface\_cfg.h, [1170](#)  
    example\_mission\_cfg.h, [915](#)  
CFE\_FS\_Header, [586](#)  
    ApplicationID, [586](#)  
    ContentType, [586](#)  
    Description, [586](#)  
    Length, [586](#)  
    ProcessorID, [587](#)  
    SpacecraftID, [587](#)  
    SubType, [587](#)  
    TimeSeconds, [587](#)  
    TimeSubSeconds, [587](#)  
CFE\_FS\_Header\_t  
    default\_cfe\_fs\_filedef.h, [1168](#)  
CFE\_FS\_InitHeader  
    cFE File Header Management APIs, [299](#)  
CFE\_FS\_INVALID\_PATH  
    cFE Return Code Defines, [229](#)  
CFE\_FS\_NOT\_IMPLEMENTED  
    cFE Return Code Defines, [229](#)  
CFE\_FS\_ParseInputFileName  
    cFE File Utility APIs, [305](#)  
CFE\_FS\_ParseInputFileNameEx  
    cFE File Utility APIs, [306](#)  
CFE\_FS\_ReadHeader  
    cFE File Header Management APIs, [299](#)  
CFE\_FS\_SetTimestamp  
    cFE File Header Management APIs, [300](#)  
CFE\_FS\_SubType  
    default\_cfe\_fs\_filedef.h, [1168](#)  
CFE\_FS\_SubType\_Enum\_t  
    default\_cfe\_fs\_filedef.h, [1168](#)

CFE\_FS\_SubType\_ES\_CDS\_REG  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_ES\_ERLOG  
     default\_cfe\_fs\_filedef.h, 1168

CFE\_FS\_SubType\_ES\_PERFDATA  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_ES\_QUERYALL  
     default\_cfe\_fs\_filedef.h, 1168

CFE\_FS\_SubType\_ES\_QUERYALLTASKS  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_ES\_SYSLOG  
     default\_cfe\_fs\_filedef.h, 1168

CFE\_FS\_SubType\_EVS\_APPDATA  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_EVS\_EVENTLOG  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_SB\_MAPDATA  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_SB\_PIPE DATA  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_SB\_ROUTEDATA  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_TBL\_IMG  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_SubType\_TBL\_REG  
     default\_cfe\_fs\_filedef.h, 1169

CFE\_FS\_WriteHeader  
     cFE File Header Management APIs, 301

CFE\_GENERIC\_SERVICE  
     cfe\_error.h, 986

CFE\_MAJOR\_VERSION  
     cfe\_version.h, 1030

CFE\_MAKE\_BIG16  
     cfe\_endian.h, 979

CFE\_MAKE\_BIG32  
     cfe\_endian.h, 979

CFE\_MINOR\_VERSION  
     cfe\_version.h, 1030

CFE\_MISSION\_ES\_APP\_TLM\_MSG  
     default\_cfe\_es\_topicids.h, 1093

CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN  
     default\_cfe\_es\_interface\_cfg.h, 1061

CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH  
     default\_cfe\_es\_interface\_cfg.h, 1061

CFE\_MISSION\_ES\_CMD\_MSG  
     default\_cfe\_es\_topicids.h, 1093

CFE\_MISSION\_ES\_CRC\_16  
     default\_cfe\_es\_interface\_cfg.h, 1061

CFE\_MISSION\_ES\_CRC\_32  
     default\_cfe\_es\_interface\_cfg.h, 1061

CFE\_MISSION\_ES\_CRC\_8  
     default\_cfe\_es\_interface\_cfg.h, 1062

CFE\_MISSION\_ES\_DEFAULT\_CRC  
     default\_cfe\_es\_interface\_cfg.h, 1062

CFE\_MISSION\_ES\_HK\_TLM\_MSG  
     default\_cfe\_es\_topicids.h, 1093

CFE\_MISSION\_ES\_MAIN\_PERF\_ID  
     sample\_perfids.h, 969

CFE\_MISSION\_ES\_MAX\_APPLICATIONS  
     default\_cfe\_es\_interface\_cfg.h, 1062

CFE\_MISSION\_ES\_MEMSTATS\_TLM\_MSG  
     default\_cfe\_es\_topicids.h, 1094

CFE\_MISSION\_ES\_PERF\_EXIT\_BIT  
     sample\_perfids.h, 969

CFE\_MISSION\_ES\_PERF\_MAX\_IDS  
     default\_cfe\_es\_interface\_cfg.h, 1062

CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS  
     default\_cfe\_es\_interface\_cfg.h, 1062

CFE\_MISSION\_ES\_SEND\_HK\_MSG  
     default\_cfe\_es\_topicids.h, 1094

CFE\_MISSION\_EVS\_CMD\_MSG  
     default\_cfe\_evs\_topicids.h, 1154

CFE\_MISSION\_EVS\_HK\_TLM\_MSG  
     default\_cfe\_evs\_topicids.h, 1154

CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_MSG  
     default\_cfe\_evs\_topicids.h, 1155

CFE\_MISSION\_EVS\_MAIN\_PERF\_ID  
     sample\_perfids.h, 969

CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH  
     default\_cfe\_evs\_interface\_cfg.h, 1140

CFE\_MISSION\_EVS\_SEND\_HK\_MSG  
     default\_cfe\_evs\_topicids.h, 1155

CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_MSG  
     default\_cfe\_evs\_topicids.h, 1155

CFE\_MISSION\_MAX\_API\_LEN  
     default\_cfe\_core\_api\_interface\_cfg.h, 972

CFE\_MISSION\_MAX\_FILE\_LEN  
     default\_cfe\_core\_api\_interface\_cfg.h, 972

CFE\_MISSION\_MAX\_NUM\_FILES  
     default\_cfe\_core\_api\_interface\_cfg.h, 973

CFE\_MISSION\_MAX\_PATH\_LEN  
     default\_cfe\_core\_api\_interface\_cfg.h, 973

CFE\_MISSION\_REV  
     cfe\_version.h, 1030

CFE\_MISSION\_SB\_ALLSUBS\_TLM\_MSG  
  default\_cfe\_sb\_topicids.h, 1200

CFE\_MISSION\_SB\_CMD\_MSG  
  default\_cfe\_sb\_topicids.h, 1200

CFE\_MISSION\_SB\_HK\_TLM\_MSG  
  default\_cfe\_sb\_topicids.h, 1201

CFE\_MISSION\_SB\_MAIN\_PERF\_ID  
  sample\_perfids.h, 969

CFE\_MISSION\_SB\_MAX\_PIPES  
  default\_cfe\_sb\_interface\_cfg.h, 1184  
    example\_mission\_cfg.h, 919

CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE  
  default\_cfe\_sb\_interface\_cfg.h, 1184  
    example\_mission\_cfg.h, 919

CFE\_MISSION\_SB\_MSG\_LIM\_PERF\_ID  
  sample\_perfids.h, 969

CFE\_MISSION\_SB\_ONESUB\_TLM\_MSG  
  default\_cfe\_sb\_topicids.h, 1201

CFE\_MISSION\_SB\_PIPE\_OFLOW\_PERF\_ID  
  sample\_perfids.h, 969

CFE\_MISSION\_SB\_SEND\_HK\_MSG  
  default\_cfe\_sb\_topicids.h, 1201

CFE\_MISSION\_SB\_STATS\_TLM\_MSG  
  default\_cfe\_sb\_topicids.h, 1201

CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_MSG  
  default\_cfe\_sb\_topicids.h, 1201

CFE\_MISSION\_TBL\_CMD\_MSG  
  default\_cfe\_tbl\_topicids.h, 1245

CFE\_MISSION\_TBL\_HK\_TLM\_MSG  
  default\_cfe\_tbl\_topicids.h, 1245

CFE\_MISSION\_TBL\_MAIN\_PERF\_ID  
  sample\_perfids.h, 969

CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN  
  default\_cfe\_tbl\_interface\_cfg.h, 1231  
    example\_mission\_cfg.h, 920

CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH  
  default\_cfe\_tbl\_interface\_cfg.h, 1232  
    example\_mission\_cfg.h, 920

CFE\_MISSION\_TBL\_REG\_TLM\_MSG  
  default\_cfe\_tbl\_topicids.h, 1245

CFE\_MISSION\_TBL\_SEND\_HK\_MSG  
  default\_cfe\_tbl\_topicids.h, 1245

CFE\_MISSION\_TIME\_1HZ\_CMD\_MSG  
  default\_cfe\_time\_topicids.h, 1303

CFE\_MISSION\_TIME\_AT\_TONE\_WAS  
  default\_cfe\_time\_interface\_cfg.h, 1286  
    example\_mission\_cfg.h, 920

CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE  
  default\_cfe\_time\_interface\_cfg.h, 1286  
    example\_mission\_cfg.h, 921

CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI  
  default\_cfe\_time\_interface\_cfg.h, 1287  
    example\_mission\_cfg.h, 921

CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC

  default\_cfe\_time\_interface\_cfg.h, 1287  
    example\_mission\_cfg.h, 921

CFE\_MISSION\_TIME\_FAKE\_TONE  
  default\_cfe\_time\_interface\_cfg.h, 1287  
    example\_mission\_cfg.h, 921

CFE\_MISSION\_TIME\_CMD\_MSG  
  default\_cfe\_time\_topicids.h, 1304

CFE\_MISSION\_TIME\_DATA\_CMD\_MSG  
  default\_cfe\_time\_topicids.h, 1304

CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS  
  default\_cfe\_time\_interface\_cfg.h, 1287  
    example\_mission\_cfg.h, 922

CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS  
  default\_cfe\_time\_interface\_cfg.h, 1287  
    example\_mission\_cfg.h, 922

CFE\_MISSION\_TIME\_DEF\_LEAPS  
  default\_cfe\_time\_interface\_cfg.h, 1287  
    example\_mission\_cfg.h, 922

CFE\_MISSION\_TIME\_DEF\_MET\_SECS  
  default\_cfe\_time\_interface\_cfg.h, 1287  
    example\_mission\_cfg.h, 922

CFE\_MISSION\_TIME\_DEF\_MET\_SUBS  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 922

CFE\_MISSION\_TIME\_DEF\_STCF\_SECS  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 922

CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_DIAG\_TLM\_MSG  
  default\_cfe\_time\_topicids.h, 1304

CFE\_MISSION\_TIME\_EPOCH\_DAY  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_EPOCH\_HOUR  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_EPOCH\_MICROS  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_EPOCH\_MINUTE  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_EPOCH\_SECOND  
  default\_cfe\_time\_interface\_cfg.h, 1288  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_EPOCH\_YEAR  
  default\_cfe\_time\_interface\_cfg.h, 1289  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_FS\_FACTOR  
  default\_cfe\_time\_interface\_cfg.h, 1289  
    example\_mission\_cfg.h, 923

CFE\_MISSION\_TIME\_HK\_TLM\_MSG

default\_cfe\_time\_topicids.h, 1304  
**CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID**  
 sample\_perfids.h, 969  
**CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID**  
 sample\_perfids.h, 970  
**CFE\_MISSION\_TIME\_MAIN\_PERF\_ID**  
 sample\_perfids.h, 970  
**CFE\_MISSION\_TIME\_MAX\_ELAPSED**  
 default\_cfe\_time\_interface\_cfg.h, 1289  
 example\_mission\_cfg.h, 924  
**CFE\_MISSION\_TIME\_MIN\_ELAPSED**  
 default\_cfe\_time\_interface\_cfg.h, 1289  
 example\_mission\_cfg.h, 924  
**CFE\_MISSION\_TIME\_SEND\_CMD\_MSG**  
 default\_cfe\_time\_topicids.h, 1304  
**CFE\_MISSION\_TIME\_SEND\_HK\_MSG**  
 default\_cfe\_time\_topicids.h, 1304  
**CFE\_MISSION\_TIME\_SENDMET\_PERF\_ID**  
 sample\_perfids.h, 970  
**CFE\_MISSION\_TIME\_TONE1HZISR\_PERF\_ID**  
 sample\_perfids.h, 970  
**CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID**  
 sample\_perfids.h, 970  
**CFE\_MISSION\_TIME\_TONE\_CMD\_MSG**  
 default\_cfe\_time\_topicids.h, 1305  
**cfe\_msg\_api\_typedefs.h**  
 CFE\_MSG\_Apld\_t, 1007  
 CFE\_MSG\_BAD\_ARGUMENT, 1007  
 CFE\_MSG\_Checksum\_t, 1007  
 CFE\_MSG\_CommandHeader\_t, 1007  
 CFE\_MSG\_EDSVersion\_t, 1007  
 CFE\_MSG\_Endian, 1009  
 CFE\_MSG\_Endian\_Big, 1009  
 CFE\_MSG\_Endian\_Invalid, 1009  
 CFE\_MSG\_Endian\_Little, 1009  
 CFE\_MSG\_Endian\_t, 1007  
 CFE\_MSG\_FcnCode\_t, 1007  
 CFE\_MSG\_HeaderVersion\_t, 1008  
 CFE\_MSG\_Message\_t, 1008  
 CFE\_MSG\_NOT\_IMPLEMENTED, 1007  
 CFE\_MSG\_PlaybackFlag, 1009  
 CFE\_MSG\_PlaybackFlag\_t, 1008  
 CFE\_MSG\_PlayFlag\_Invalid, 1009  
 CFE\_MSG\_PlayFlag\_Original, 1009  
 CFE\_MSG\_PlayFlag\_Playback, 1009  
 CFE\_MSG\_SegFlag\_Continue, 1009  
 CFE\_MSG\_SegFlag\_First, 1009  
 CFE\_MSG\_SegFlag\_Invalid, 1009  
 CFE\_MSG\_SegFlag\_Last, 1009  
 CFE\_MSG\_SegFlag\_Unsegmented, 1009  
 CFE\_MSG\_SegmentationFlag, 1009  
 CFE\_MSG\_SegmentationFlag\_t, 1008  
 CFE\_MSG\_SequenceCount\_t, 1008  
 CFE\_MSG\_Size\_t, 1008  
 CFE\_MSG\_Subsystem\_t, 1008  
 CFE\_MSG\_System\_t, 1008  
 CFE\_MSG\_TelemetryHeader\_t, 1008  
 CFE\_MSG\_Type, 1009  
 CFE\_MSG\_Type\_Cmd, 1009  
 CFE\_MSG\_Type\_Invalid, 1009  
 CFE\_MSG\_Type\_t, 1008  
 CFE\_MSG\_Type\_Tlm, 1009  
 CFE\_MSG\_WRONG\_MSG\_TYPE, 1007  
**CFE\_MSG\_Apld\_t**  
 cfe\_msg\_api\_typedefs.h, 1007  
**CFE\_MSG\_BAD\_ARGUMENT**  
 cfe\_msg\_api\_typedefs.h, 1007  
**CFE\_MSG\_Checksum\_t**  
 cfe\_msg\_api\_typedefs.h, 1007  
**CFE\_MSG\_CommandHeader\_t**  
 cfe\_msg\_api\_typedefs.h, 1007  
**CFE\_MSG\_EDSVersion\_t**  
 cfe\_msg\_api\_typedefs.h, 1007  
**CFE\_MSG\_Endian**  
 cfe\_msg\_api\_typedefs.h, 1009  
**CFE\_MSG\_Endian\_Big**  
 cfe\_msg\_api\_typedefs.h, 1009  
**CFE\_MSG\_Endian\_Invalid**  
 cfe\_msg\_api\_typedefs.h, 1009  
**CFE\_MSG\_Endian\_Little**  
 cfe\_msg\_api\_typedefs.h, 1009  
**CFE\_MSG\_Endian\_t**  
 cfe\_msg\_api\_typedefs.h, 1007  
**CFE\_MSG\_FcnCode\_t**  
 cfe\_msg\_api\_typedefs.h, 1007  
**CFE\_MSG\_GenerateChecksum**  
 cFE Message Secondary Header APIs, 325  
**CFE\_MSG\_GetApld**  
 cFE Message Primary Header APIs, 310  
**CFE\_MSG\_GetEDSVersion**  
 cFE Message Extended Header APIs, 319  
**CFE\_MSG\_GetEndian**  
 cFE Message Extended Header APIs, 320  
**CFE\_MSG\_GetFcnCode**  
 cFE Message Secondary Header APIs, 326  
**CFE\_MSG\_GetHasSecondaryHeader**  
 cFE Message Primary Header APIs, 311  
**CFE\_MSG\_GetHeaderVersion**  
 cFE Message Primary Header APIs, 311  
**CFE\_MSG\_GetMsgId**  
 cFE Message Id APIs, 330  
**CFE\_MSG\_GetMsgTime**  
 cFE Message Secondary Header APIs, 326  
**CFE\_MSG\_GetNextSequenceCount**  
 cFE Message Primary Header APIs, 312  
**CFE\_MSG\_GetPlaybackFlag**  
 cFE Message Extended Header APIs, 320  
**CFE\_MSG\_GetSegmentationFlag**

cFE Message Primary Header APIs, 312  
CFE\_MSG\_GetSequenceCount  
    cFE Message Primary Header APIs, 313  
CFE\_MSG\_GetSize  
    cFE Message Primary Header APIs, 313  
CFE\_MSG\_GetSubsystem  
    cFE Message Extended Header APIs, 321  
CFE\_MSG\_GetSystem  
    cFE Message Extended Header APIs, 321  
CFE\_MSG.GetType  
    cFE Message Primary Header APIs, 314  
CFE\_MSG.GetTypeFromMsgId  
    cFE Message Id APIs, 330  
CFE\_MSG\_HeaderVersion\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_Init  
    cFE Generic Message APIs, 308  
CFE\_MSG\_Message\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_NOT\_IMPLEMENTED  
    `cfe_msg_api_typedefs.h`, 1007  
CFE\_MSG\_PlaybackFlag  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_PlaybackFlag\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_PlayFlag\_Invalid  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_PlayFlag\_Original  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_PlayFlag\_Playback  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_SegFlag\_Continue  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_SegFlag\_First  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_SegFlag\_Invalid  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_SegFlag\_Last  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_SegFlag\_Unsegmented  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_SegmentationFlag  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_SegmentationFlag\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_SequenceCount\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_SetApId  
    cFE Message Primary Header APIs, 314  
CFE\_MSG\_SetEDSVersion  
    cFE Message Extended Header APIs, 322  
CFE\_MSG\_SetEndian  
    cFE Message Extended Header APIs, 322  
CFE\_MSG\_SetFcnCode  
    cFE Message Secondary Header APIs, 327  
CFE\_MSG\_SetHasSecondaryHeader  
    cFE Message Primary Header APIs, 315  
CFE\_MSG\_SetHeaderVersion  
    cFE Message Primary Header APIs, 315  
CFE\_MSG\_SetMsgId  
    cFE Message Id APIs, 331  
CFE\_MSG\_SetMsgTime  
    cFE Message Secondary Header APIs, 327  
CFE\_MSG\_SetPlaybackFlag  
    cFE Message Extended Header APIs, 323  
CFE\_MSG\_SetSegmentationFlag  
    cFE Message Primary Header APIs, 316  
CFE\_MSG\_SetSequenceCount  
    cFE Message Primary Header APIs, 316  
CFE\_MSG\_SetSize  
    cFE Message Primary Header APIs, 317  
CFE\_MSG\_SetSubsystem  
    cFE Message Extended Header APIs, 323  
CFE\_MSG\_SetSystem  
    cFE Message Extended Header APIs, 324  
CFE\_MSG\_SetType  
    cFE Message Primary Header APIs, 317  
CFE\_MSG\_Size\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_Subsystem\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_System\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_TelemetryHeader\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_Type  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_Type\_Cmd  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_Type\_Invalid  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_Type\_t  
    `cfe_msg_api_typedefs.h`, 1008  
CFE\_MSG\_Type\_Tlm  
    `cfe_msg_api_typedefs.h`, 1009  
CFE\_MSG\_UpdateHeader  
    cFE Generic Message APIs, 308  
CFE\_MSG\_ValidateChecksum  
    cFE Message Secondary Header APIs, 328  
CFE\_MSG\_Verify  
    cFE Message Checking APIs, 332  
CFE\_MSG\_WRONG\_MSG\_TYPE  
    `cfe_msg_api_typedefs.h`, 1007  
CFE\_PLATFORM\_CMD\_MID\_BASE  
    `default_cfe_core_api_base_msgids.h`, 971  
CFE\_PLATFORM\_CMD\_MID\_BASE\_GLOB  
    `default_cfe_core_api_base_msgids.h`, 971  
CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC

example\_platform\_cfg.h, 928  
CFE\_PLATFORM\_ENDIAN  
example\_platform\_cfg.h, 929  
CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT  
default\_cfe\_es\_internal\_cfg.h, 1065  
example\_platform\_cfg.h, 929  
CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE  
default\_cfe\_es\_internal\_cfg.h, 1065  
example\_platform\_cfg.h, 929  
CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE  
default\_cfe\_es\_internal\_cfg.h, 1066  
example\_platform\_cfg.h, 930  
CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES  
default\_cfe\_es\_internal\_cfg.h, 1066  
example\_platform\_cfg.h, 930  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01  
default\_cfe\_es\_internal\_cfg.h, 1066  
example\_platform\_cfg.h, 930  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02  
default\_cfe\_es\_internal\_cfg.h, 1066  
example\_platform\_cfg.h, 930  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03  
default\_cfe\_es\_internal\_cfg.h, 1066  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04  
default\_cfe\_es\_internal\_cfg.h, 1066  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 931  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 932  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14  
default\_cfe\_es\_internal\_cfg.h, 1067  
example\_platform\_cfg.h, 932  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15  
default\_cfe\_es\_internal\_cfg.h, 1068  
example\_platform\_cfg.h, 932  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16  
default\_cfe\_es\_internal\_cfg.h, 1068  
example\_platform\_cfg.h, 932  
CFE\_PLATFORM\_ES\_CDS\_SIZE  
default\_cfe\_es\_internal\_cfg.h, 1068  
example\_platform\_cfg.h, 932  
CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE  
default\_cfe\_es\_internal\_cfg.h, 1068  
example\_platform\_cfg.h, 932  
CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE  
default\_cfe\_es\_internal\_cfg.h, 1068  
example\_platform\_cfg.h, 933  
CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE  
default\_cfe\_es\_internal\_cfg.h, 1069  
example\_platform\_cfg.h, 933  
CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME  
default\_cfe\_es\_internal\_cfg.h, 1069  
example\_platform\_cfg.h, 933  
CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE  
default\_cfe\_es\_internal\_cfg.h, 1069  
example\_platform\_cfg.h, 933  
CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE  
default\_cfe\_es\_internal\_cfg.h, 1070  
example\_platform\_cfg.h, 934  
CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE  
default\_cfe\_es\_internal\_cfg.h, 1070  
example\_platform\_cfg.h, 934  
CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE  
default\_cfe\_es\_internal\_cfg.h, 1070  
example\_platform\_cfg.h, 934  
CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE  
default\_cfe\_es\_internal\_cfg.h, 1071  
example\_platform\_cfg.h, 935  
CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES  
default\_cfe\_es\_internal\_cfg.h, 1071  
example\_platform\_cfg.h, 935  
CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE  
default\_cfe\_es\_internal\_cfg.h, 1071  
example\_platform\_cfg.h, 935  
CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS  
default\_cfe\_es\_internal\_cfg.h, 1072  
example\_platform\_cfg.h, 936  
CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE  
default\_cfe\_es\_internal\_cfg.h, 1072  
example\_platform\_cfg.h, 936  
CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS  
default\_cfe\_es\_internal\_cfg.h, 1072  
example\_platform\_cfg.h, 936

CFE\_PLATFORM\_ES\_MAX\_LIBRARIES  
  default\_cfe\_es\_internal\_cfg.h, 1072  
  example\_platform\_cfg.h, 936  
CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS  
  default\_cfe\_es\_internal\_cfg.h, 1073  
  example\_platform\_cfg.h, 937  
CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS  
  default\_cfe\_es\_internal\_cfg.h, 1073  
  example\_platform\_cfg.h, 937  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01  
  default\_cfe\_es\_internal\_cfg.h, 1073  
  example\_platform\_cfg.h, 937  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10  
  default\_cfe\_es\_internal\_cfg.h, 1074  
  example\_platform\_cfg.h, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING  
  default\_cfe\_es\_internal\_cfg.h, 1075  
  example\_platform\_cfg.h, 939  
CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE  
  default\_cfe\_es\_internal\_cfg.h, 1076  
  example\_platform\_cfg.h, 940  
CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE  
  default\_cfe\_es\_internal\_cfg.h, 1076  
  example\_platform\_cfg.h, 940  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY  
  default\_cfe\_es\_internal\_cfg.h, 1076  
  example\_platform\_cfg.h, 940  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY  
  default\_cfe\_es\_internal\_cfg.h, 1076  
  example\_platform\_cfg.h, 940  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE  
  default\_cfe\_es\_internal\_cfg.h, 1077  
  example\_platform\_cfg.h, 941  
CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE  
  default\_cfe\_es\_internal\_cfg.h, 1077  
  example\_platform\_cfg.h, 941  
CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS  
  default\_cfe\_es\_internal\_cfg.h, 1077  
  example\_platform\_cfg.h, 941  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL  
  default\_cfe\_es\_internal\_cfg.h, 1078  
  example\_platform\_cfg.h, 942  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT  
  default\_cfe\_es\_internal\_cfg.h, 1078  
  example\_platform\_cfg.h, 942  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE  
  default\_cfe\_es\_internal\_cfg.h, 1078  
  example\_platform\_cfg.h, 942  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL  
  default\_cfe\_es\_internal\_cfg.h, 1078  
  example\_platform\_cfg.h, 942  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT  
  default\_cfe\_es\_internal\_cfg.h, 1079  
  example\_platform\_cfg.h, 943  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE  
  default\_cfe\_es\_internal\_cfg.h, 1079  
  example\_platform\_cfg.h, 943  
CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS  
  default\_cfe\_es\_internal\_cfg.h, 1079  
  example\_platform\_cfg.h, 943  
CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING  
  default\_cfe\_es\_internal\_cfg.h, 1079  
  example\_platform\_cfg.h, 943

CFE_PLATFORM_ES_RAM_DISK_NUM_SECTORS	CFE_PLATFORM_EVS_MAX_EVENT_FILTERS
default_cfe_es_internal_cfg.h, 1080	default_cfe_evs_internal_cfg.h, 1143
example_platform_cfg.h, 944	example_platform_cfg.h, 950
CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED	CFE_PLATFORM_EVS_PORT_DEFAULT
default_cfe_es_internal_cfg.h, 1080	default_cfe_evs_internal_cfg.h, 1144
example_platform_cfg.h, 944	example_platform_cfg.h, 950
CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE	CFE_PLATFORM_EVS_START_TASK_PRIORITY
default_cfe_es_internal_cfg.h, 1080	default_cfe_evs_internal_cfg.h, 1144
example_platform_cfg.h, 944	example_platform_cfg.h, 950
CFE_PLATFORM_ES_START_TASK_PRIORITY	CFE_PLATFORM_EVS_START_TASK_STACK_SIZE
default_cfe_es_internal_cfg.h, 1081	default_cfe_evs_internal_cfg.h, 1144
example_platform_cfg.h, 945	example_platform_cfg.h, 950
CFE_PLATFORM_ES_START_TASK_STACK_SIZE	CFE_PLATFORM_SB_BUF_MEMORY_BYTES
default_cfe_es_internal_cfg.h, 1081	default_cfe_sb_internal_cfg.h, 1186
example_platform_cfg.h, 945	example_platform_cfg.h, 951
CFE_PLATFORM_ES_STARTUP_SCRIPT_TIMEOUT_MSEC	CFE_PLATFORM_SB_DEFAULT_MAP_FILENAME
default_cfe_es_internal_cfg.h, 1081	default_cfe_sb_internal_cfg.h, 1186
example_platform_cfg.h, 945	example_platform_cfg.h, 951
CFE_PLATFORM_ES_STARTUP_SYNC_POLL_MSEC	CFE_PLATFORM_SB_DEFAULT_MSG_LIMIT
default_cfe_es_internal_cfg.h, 1082	default_cfe_sb_internal_cfg.h, 1187
example_platform_cfg.h, 946	example_platform_cfg.h, 951
CFE_PLATFORM_ES_SYSTEM_LOG_SIZE	CFE_PLATFORM_SB_DEFAULT_PIPE_FILENAME
default_cfe_es_internal_cfg.h, 1082	default_cfe_sb_internal_cfg.h, 1187
example_platform_cfg.h, 946	example_platform_cfg.h, 952
CFE_PLATFORM_ES_USER_RESERVED_SIZE	CFE_PLATFORM_SB_DEFAULT_ROUTING_FILENAME
default_cfe_es_internal_cfg.h, 1082	default_cfe_sb_internal_cfg.h, 1187
example_platform_cfg.h, 946	example_platform_cfg.h, 952
CFE_PLATFORM_ES_VOLATILE_STARTUP_FILE	CFE_PLATFORM_SB_FILTER_MASK1
default_cfe_es_internal_cfg.h, 1083	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 947	example_platform_cfg.h, 952
CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC	CFE_PLATFORM_SB_FILTER_MASK2
default_cfe_evs_internal_cfg.h, 1141	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 947	example_platform_cfg.h, 952
CFE_PLATFORM_EVS_DEFAULT_APP_DATA_FILE	CFE_PLATFORM_SB_FILTER_MASK3
default_cfe_evs_internal_cfg.h, 1141	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 947	example_platform_cfg.h, 953
CFE_PLATFORM_EVS_DEFAULT_LOG_FILE	CFE_PLATFORM_SB_FILTER_MASK4
default_cfe_evs_internal_cfg.h, 1142	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 948	example_platform_cfg.h, 953
CFE_PLATFORM_EVS_DEFAULT_LOG_MODE	CFE_PLATFORM_SB_FILTER_MASK5
default_cfe_evs_internal_cfg.h, 1142	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 948	example_platform_cfg.h, 953
CFE_PLATFORM_EVS_DEFAULT_MSG_FORMAT_MODE	CFE_PLATFORM_SB_FILTER_MASK6
default_cfe_evs_internal_cfg.h, 1142	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 948	example_platform_cfg.h, 953
CFE_PLATFORM_EVS_DEFAULT_TYPE_FLAG	CFE_PLATFORM_SB_FILTER_MASK7
default_cfe_evs_internal_cfg.h, 1142	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 949	example_platform_cfg.h, 953
CFE_PLATFORM_EVS_LOG_MAX	CFE_PLATFORM_SB_FILTER_MASK8
default_cfe_evs_internal_cfg.h, 1143	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 949	example_platform_cfg.h, 953
CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST	CFE_PLATFORM_SB_FILTERED_EVENT1
default_cfe_evs_internal_cfg.h, 1143	default_cfe_sb_internal_cfg.h, 1188
example_platform_cfg.h, 949	example_platform_cfg.h, 953

CFE\_PLATFORM\_SB\_FILTERED\_EVENT2  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 953

CFE\_PLATFORM\_SB\_FILTERED\_EVENT3  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 953

CFE\_PLATFORM\_SB\_FILTERED\_EVENT4  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 953

CFE\_PLATFORM\_SB\_FILTERED\_EVENT5  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 954

CFE\_PLATFORM\_SB\_FILTERED\_EVENT6  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 954

CFE\_PLATFORM\_SB\_FILTERED\_EVENT7  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 954

CFE\_PLATFORM\_SB\_FILTERED\_EVENT8  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 954

CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID  
  default\_cfe\_sb\_internal\_cfg.h, 1189  
  example\_platform\_cfg.h, 954

CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE  
  default\_cfe\_sb\_internal\_cfg.h, 1190  
  example\_platform\_cfg.h, 954

CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT  
  default\_cfe\_sb\_internal\_cfg.h, 1190  
  example\_platform\_cfg.h, 954

CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS  
  default\_cfe\_sb\_internal\_cfg.h, 1190  
  example\_platform\_cfg.h, 955

CFE\_PLATFORM\_SB\_MAX\_PIPES  
  default\_cfe\_sb\_internal\_cfg.h, 1191  
  example\_platform\_cfg.h, 955

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01  
  default\_cfe\_sb\_internal\_cfg.h, 1191  
  example\_platform\_cfg.h, 955

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02  
  default\_cfe\_sb\_internal\_cfg.h, 1191  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03  
  default\_cfe\_sb\_internal\_cfg.h, 1191  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04  
  default\_cfe\_sb\_internal\_cfg.h, 1191  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05  
  default\_cfe\_sb\_internal\_cfg.h, 1191  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 956

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16  
  default\_cfe\_sb\_internal\_cfg.h, 1192  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY  
  default\_cfe\_sb\_internal\_cfg.h, 1193  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE  
  default\_cfe\_sb\_internal\_cfg.h, 1193  
  example\_platform\_cfg.h, 957

CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES  
  default\_cfe\_tbl\_internal\_cfg.h, 1233  
  example\_platform\_cfg.h, 958

CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE  
  default\_cfe\_tbl\_internal\_cfg.h, 1233  
  example\_platform\_cfg.h, 958

CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES  
  default\_cfe\_tbl\_internal\_cfg.h, 1233  
  example\_platform\_cfg.h, 958

CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE  
  default\_cfe\_tbl\_internal\_cfg.h, 1234  
  example\_platform\_cfg.h, 959

CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES  
  default\_cfe\_tbl\_internal\_cfg.h, 1234  
  example\_platform\_cfg.h, 959

CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES  
  default\_cfe\_tbl\_internal\_cfg.h, 1234  
  example\_platform\_cfg.h, 959

CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS  
     default\_cfe\_tbl\_internal\_cfg.h, [1234](#)  
     example\_platform\_cfg.h, [960](#)

CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS  
     default\_cfe\_tbl\_internal\_cfg.h, [1235](#)  
     example\_platform\_cfg.h, [960](#)

CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE  
     default\_cfe\_tbl\_internal\_cfg.h, [1235](#)  
     example\_platform\_cfg.h, [960](#)

CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY  
     default\_cfe\_tbl\_internal\_cfg.h, [1235](#)  
     example\_platform\_cfg.h, [961](#)

CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE  
     default\_cfe\_tbl\_internal\_cfg.h, [1236](#)  
     example\_platform\_cfg.h, [961](#)

CFE\_PLATFORM\_TBL\_U32FROM4CHARS  
     default\_cfe\_tbl\_internal\_cfg.h, [1236](#)  
     example\_platform\_cfg.h, [961](#)

CFE\_PLATFORM\_TBL\_VALID\_PRID\_1  
     default\_cfe\_tbl\_internal\_cfg.h, [1236](#)  
     example\_platform\_cfg.h, [962](#)

CFE\_PLATFORM\_TBL\_VALID\_PRID\_2  
     default\_cfe\_tbl\_internal\_cfg.h, [1236](#)  
     example\_platform\_cfg.h, [962](#)

CFE\_PLATFORM\_TBL\_VALID\_PRID\_3  
     default\_cfe\_tbl\_internal\_cfg.h, [1237](#)  
     example\_platform\_cfg.h, [962](#)

CFE\_PLATFORM\_TBL\_VALID\_PRID\_4  
     default\_cfe\_tbl\_internal\_cfg.h, [1237](#)  
     example\_platform\_cfg.h, [962](#)

CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT  
     default\_cfe\_tbl\_internal\_cfg.h, [1237](#)  
     example\_platform\_cfg.h, [962](#)

CFE\_PLATFORM\_TBL\_VALID\_SCID\_1  
     default\_cfe\_tbl\_internal\_cfg.h, [1237](#)  
     example\_platform\_cfg.h, [962](#)

CFE\_PLATFORM\_TBL\_VALID\_SCID\_2  
     default\_cfe\_tbl\_internal\_cfg.h, [1237](#)  
     example\_platform\_cfg.h, [963](#)

CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT  
     default\_cfe\_tbl\_internal\_cfg.h, [1237](#)  
     example\_platform\_cfg.h, [963](#)

CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY  
     default\_cfe\_time\_internal\_cfg.h, [1291](#)  
     example\_platform\_cfg.h, [963](#)

CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE  
     default\_cfe\_time\_internal\_cfg.h, [1291](#)  
     example\_platform\_cfg.h, [963](#)

CFE\_PLATFORM\_TIME\_CFG\_CLIENT  
     default\_cfe\_time\_internal\_cfg.h, [1291](#)  
     example\_platform\_cfg.h, [963](#)

CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY  
     default\_cfe\_time\_internal\_cfg.h, [1291](#)  
     example\_platform\_cfg.h, [963](#)

CFE\_PLATFORM\_TIME\_CFG\_SERVER  
     default\_cfe\_time\_internal\_cfg.h, [1291](#)  
     example\_platform\_cfg.h, [964](#)

CFE\_PLATFORM\_TIME\_CFG\_SIGNAL  
     default\_cfe\_time\_internal\_cfg.h, [1291](#)  
     example\_platform\_cfg.h, [964](#)

CFE\_PLATFORM\_TIME\_CFG\_SOURCE  
     default\_cfe\_time\_internal\_cfg.h, [1292](#)  
     example\_platform\_cfg.h, [964](#)

CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS  
     default\_cfe\_time\_internal\_cfg.h, [1292](#)  
     example\_platform\_cfg.h, [965](#)

CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET  
     default\_cfe\_time\_internal\_cfg.h, [1292](#)  
     example\_platform\_cfg.h, [965](#)

CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME  
     default\_cfe\_time\_internal\_cfg.h, [1292](#)  
     example\_platform\_cfg.h, [965](#)

CFE\_PLATFORM\_TIME\_CFG\_START\_FLY  
     default\_cfe\_time\_internal\_cfg.h, [1293](#)  
     example\_platform\_cfg.h, [965](#)

CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT  
     default\_cfe\_time\_internal\_cfg.h, [1293](#)  
     example\_platform\_cfg.h, [966](#)

CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL  
     default\_cfe\_time\_internal\_cfg.h, [1293](#)  
     example\_platform\_cfg.h, [966](#)

CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS  
     default\_cfe\_time\_internal\_cfg.h, [1293](#)  
     example\_platform\_cfg.h, [966](#)

CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS  
     default\_cfe\_time\_internal\_cfg.h, [1294](#)  
     example\_platform\_cfg.h, [967](#)

CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS  
     default\_cfe\_time\_internal\_cfg.h, [1294](#)  
     example\_platform\_cfg.h, [967](#)

CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS  
     default\_cfe\_time\_internal\_cfg.h, [1294](#)  
     example\_platform\_cfg.h, [967](#)

CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY  
     default\_cfe\_time\_internal\_cfg.h, [1294](#)  
     example\_platform\_cfg.h, [967](#)

CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE  
     default\_cfe\_time\_internal\_cfg.h, [1294](#)  
     example\_platform\_cfg.h, [967](#)

CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY  
     default\_cfe\_time\_internal\_cfg.h, [1295](#)  
     example\_platform\_cfg.h, [968](#)

CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE  
     default\_cfe\_time\_internal\_cfg.h, [1295](#)  
     example\_platform\_cfg.h, [968](#)

CFE\_PLATFORM\_TLM\_MID\_BASE  
     default\_cfe\_core\_api\_base\_msgids.h, [972](#)  
     cfe\_psp.h

BUFF\_SIZE, 1357  
CFE\_PSP\_AttachExceptions, 1360  
CFE\_PSP\_Decompress, 1361  
CFE\_PSP\_EepromPowerDown, 1361  
CFE\_PSP\_EepromPowerUp, 1361  
CFE\_PSP\_EepromWrite16, 1361  
CFE\_PSP\_EepromWrite32, 1361  
CFE\_PSP\_EepromWrite8, 1361  
CFE\_PSP\_EepromWriteDisable, 1361  
CFE\_PSP\_EepromWriteEnable, 1361  
CFE\_PSP\_Exception\_CopyContext, 1361  
CFE\_PSP\_Exception\_GetCount, 1361  
CFE\_PSP\_Exception\_GetSummary, 1361  
CFE\_PSP\_FlushCaches, 1362  
CFE\_PSP\_Get\_Dec, 1362  
CFE\_PSP\_Get\_Timebase, 1362  
CFE\_PSP\_Get\_Timer\_Tick, 1362  
CFE\_PSP\_GetBuildNumber, 1362  
CFE\_PSP\_GetCDSSize, 1363  
CFE\_PSP\_GetCFETextSegmentInfo, 1363  
CFE\_PSP\_GetKernelTextSegmentInfo, 1363  
CFE\_PSP\_GetProcessorId, 1363  
CFE\_PSP\_GetProcessorName, 1363  
CFE\_PSP\_GetResetArea, 1363  
CFE\_PSP\_GetRestartType, 1363  
CFE\_PSP\_GetSpacecraftId, 1363  
CFE\_PSP\_GetTime, 1363  
CFE\_PSP\_GetTimerLow32Rollover, 1364  
CFE\_PSP\_GetTimerTicksPerSecond, 1364  
CFE\_PSP\_GetUserReservedArea, 1364  
CFE\_PSP\_GetVersionCodeName, 1364  
CFE\_PSP\_GetVersionNumber, 1364  
CFE\_PSP\_GetVersionString, 1365  
CFE\_PSP\_GetVolatileDiskMem, 1365  
CFE\_PSP\_InitSSR, 1365  
CFE\_PSP\_Main, 1365  
CFE\_PSP\_MEM\_ANY, 1357  
CFE\_PSP\_MEM\_ATTR\_READ, 1358  
CFE\_PSP\_MEM\_ATTR\_READWRITE, 1358  
CFE\_PSP\_MEM\_ATTR\_WRITE, 1358  
CFE\_PSP\_MEM\_EEPROM, 1358  
CFE\_PSP\_MEM\_INVALID, 1358  
CFE\_PSP\_MEM\_RAM, 1358  
CFE\_PSP\_MEM\_SIZE\_BYTE, 1358  
CFE\_PSP\_MEM\_SIZE\_DWORD, 1358  
CFE\_PSP\_MEM\_SIZE\_WORD, 1358  
CFE\_PSP\_MemCpy, 1365  
CFE\_PSP\_MemRangeGet, 1365  
CFE\_PSP\_MemRanges, 1365  
CFE\_PSP\_MemRangeSet, 1365  
CFE\_PSP\_MemRead16, 1366  
CFE\_PSP\_MemRead32, 1366  
CFE\_PSP\_MemRead8, 1366  
CFE\_PSP\_MemSet, 1366  
CFE\_PSP\_MemValidateRange, 1366  
CFE\_PSP\_MemWrite16, 1366  
CFE\_PSP\_MemWrite32, 1366  
CFE\_PSP\_MemWrite8, 1366  
CFE\_PSP\_Panic, 1366  
CFE\_PSP\_PANIC\_CORE\_APP, 1358  
CFE\_PSP\_PANIC\_GENERAL\_FAILURE, 1358  
CFE\_PSP\_PANIC\_MEMORY\_ALLOC, 1358  
CFE\_PSP\_PANIC\_NONVOL\_DISK, 1358  
CFE\_PSP\_PANIC\_STARTUP, 1359  
CFE\_PSP\_PANIC\_STARTUP\_SEM, 1359  
CFE\_PSP\_PANIC\_VOLATILE\_DISK, 1359  
CFE\_PSP\_PortRead16, 1367  
CFE\_PSP\_PortRead32, 1367  
CFE\_PSP\_PortRead8, 1367  
CFE\_PSP\_PortWrite16, 1367  
CFE\_PSP\_PortWrite32, 1367  
CFE\_PSP\_PortWrite8, 1367  
CFE\_PSP\_ReadFromCDS, 1367  
CFE\_PSP\_Restart, 1367  
CFE\_PSP\_RST\_SUBTYPE\_BANKSWITCH\_RESET, 1359  
CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION, 1359  
CFE\_PSP\_RST\_SUBTYPE\_HW\_SPECIAL\_COMMAND, 1359  
CFE\_PSP\_RST\_SUBTYPE\_HW\_WATCHDOG, 1359  
CFE\_PSP\_RST\_SUBTYPE\_HWDEBUG\_RESET, 1359  
CFE\_PSP\_RST\_SUBTYPE\_MAX, 1359  
CFE\_PSP\_RST\_SUBTYPE\_POWER\_CYCLE, 1359  
CFE\_PSP\_RST\_SUBTYPE\_PUSH\_BUTTON, 1359  
CFE\_PSP\_RST\_SUBTYPE\_RESET\_COMMAND, 1360  
CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET, 1360  
CFE\_PSP\_RST\_TYPE\_MAX, 1360  
CFE\_PSP\_RST\_TYPE\_POWERON, 1360  
CFE\_PSP\_RST\_TYPE\_PROCESSOR, 1360  
CFE\_PSP\_SetDefaultExceptionEnvironment, 1367  
CFE\_PSP\_SOFT\_TIMEBASE\_NAME, 1360  
CFE\_PSP\_WatchdogDisable, 1367  
CFE\_PSP\_WatchdogEnable, 1367  
CFE\_PSP\_WatchdogGet, 1368  
CFE\_PSP\_WatchdogInit, 1368  
CFE\_PSP\_WatchdogService, 1368  
CFE\_PSP\_WatchdogSet, 1368  
CFE\_PSP\_WriteToCDS, 1368  
SIZE\_BYTE, 1360  
SIZE\_HALF, 1360  
SIZE\_WORD, 1360  
CFE\_PSP\_AttachExceptions  
  cfe\_psp.h, 1360  
CFE\_PSP\_Decompress

cfe\_psp.h, 1361  
CFE\_PSP\_EepromPowerDown  
    cfe\_psp.h, 1361  
CFE\_PSP\_EepromPowerUp  
    cfe\_psp.h, 1361  
CFE\_PSP\_EepromWrite16  
    cfe\_psp.h, 1361  
CFE\_PSP\_EepromWrite32  
    cfe\_psp.h, 1361  
CFE\_PSP\_EepromWrite8  
    cfe\_psp.h, 1361  
CFE\_PSP\_EepromWriteDisable  
    cfe\_psp.h, 1361  
CFE\_PSP\_EepromWriteEnable  
    cfe\_psp.h, 1361  
CFE\_PSP\_ERROR  
    cfe\_psp\_error.h, 1369  
cfe\_psp\_error.h  
    CFE\_PSP\_ERROR, 1369  
    CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED,  
        1369  
    CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED, 1369  
    CFE\_PSP\_ERROR\_TIMEOUT, 1369  
    CFE\_PSP\_INVALID\_INT\_NUM, 1369  
    CFE\_PSP\_INVALID\_MEM\_ADDR, 1369  
    CFE\_PSP\_INVALID\_MEM\_ATTR, 1369  
    CFE\_PSP\_INVALID\_MEM\_RANGE, 1369  
    CFE\_PSP\_INVALID\_MEM\_SIZE, 1369  
    CFE\_PSP\_INVALID\_MEM\_TYPE, 1370  
    CFE\_PSP\_INVALID\_MEM\_WORDSIZE, 1370  
    CFE\_PSP\_INVALID\_MODULE\_ID, 1370  
    CFE\_PSP\_INVALID\_MODULE\_NAME, 1370  
    CFE\_PSP\_INVALID\_POINTER, 1370  
    CFE\_PSP\_NO\_EXCEPTION\_DATA, 1370  
    CFE\_PSP\_STATUS\_C, 1370  
    CFE\_PSP\_STATUS\_STRING\_LENGTH, 1370  
    CFE\_PSP\_Status\_t, 1370  
    CFE\_PSP\_StatusString\_t, 1370  
    CFE\_PSP\_StatusToString, 1371  
    CFE\_PSP\_SUCCESS, 1370  
CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_ERROR\_TIMEOUT  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_Exception\_CopyContext  
    cfe\_psp.h, 1361  
CFE\_PSP\_Exception\_GetCount  
    cfe\_psp.h, 1361  
CFE\_PSP\_Exception\_GetSummary  
    cfe\_psp.h, 1361  
CFE\_PSP\_FlushCaches  
    cfe\_psp.h, 1362  
CFE\_PSP\_GetDec  
    cfe\_psp.h, 1362  
CFE\_PSP\_GetTimebase  
    cfe\_psp.h, 1362  
CFE\_PSP\_GetTimer\_Tick  
    cfe\_psp.h, 1362  
CFE\_PSP\_GetBuildNumber  
    cfe\_psp.h, 1362  
CFE\_PSP\_GetCDSSize  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetCFETextSegmentInfo  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetKernelTextSegmentInfo  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetProcessorId  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetProcessorName  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetResetArea  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetRestartType  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetSpacecraftId  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetTime  
    cfe\_psp.h, 1363  
CFE\_PSP\_GetTimerLow32Rollover  
    cfe\_psp.h, 1364  
CFE\_PSP\_GetTimerTicksPerSecond  
    cfe\_psp.h, 1364  
CFE\_PSP GetUserReservedArea  
    cfe\_psp.h, 1364  
CFE\_PSP\_GetVersionCodeName  
    cfe\_psp.h, 1364  
CFE\_PSP\_GetVersionNumber  
    cfe\_psp.h, 1364  
CFE\_PSP\_GetVersionString  
    cfe\_psp.h, 1365  
CFE\_PSP\_GetVolatileDiskMem  
    cfe\_psp.h, 1365  
CFE\_PSP\_InitSSR  
    cfe\_psp.h, 1365  
CFE\_PSP\_INVALID\_INT\_NUM  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_INVALID\_MEM\_ADDR  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_INVALID\_MEM\_ATTR  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_INVALID\_MEM\_RANGE  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_INVALID\_MEM\_SIZE  
    cfe\_psp\_error.h, 1369  
CFE\_PSP\_INVALID\_MEM\_TYPE  
    cfe\_psp\_error.h, 1370

CFE\_PSP\_INVALID\_MEM\_WORDSIZE  
    cfe\_psp\_error.h, 1370

CFE\_PSP\_INVALID\_MODULE\_ID  
    cfe\_psp\_error.h, 1370

CFE\_PSP\_INVALID\_MODULE\_NAME  
    cfe\_psp\_error.h, 1370

CFE\_PSP\_INVALID\_POINTER  
    cfe\_psp\_error.h, 1370

CFE\_PSP\_Main  
    cfe\_psp.h, 1365

CFE\_PSP\_MEM\_ANY  
    cfe\_psp.h, 1357

CFE\_PSP\_MEM\_ATTR\_READ  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_ATTR\_READWRITE  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_ATTR\_WRITE  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_EEPROM  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_INVALID  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_RAM  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_SIZE\_BYTE  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_SIZE\_DWORD  
    cfe\_psp.h, 1358

CFE\_PSP\_MEM\_SIZE\_WORD  
    cfe\_psp.h, 1358

CFE\_PSP\_MemCopy  
    cfe\_psp.h, 1365

CFE\_PSP\_MemRangeGet  
    cfe\_psp.h, 1365

CFE\_PSP\_MemRanges  
    cfe\_psp.h, 1365

CFE\_PSP\_MemRangeSet  
    cfe\_psp.h, 1365

CFE\_PSP\_MemRead16  
    cfe\_psp.h, 1366

CFE\_PSP\_MemRead32  
    cfe\_psp.h, 1366

CFE\_PSP\_MemRead8  
    cfe\_psp.h, 1366

CFE\_PSP\_MemSet  
    cfe\_psp.h, 1366

CFE\_PSP\_MemValidateRange  
    cfe\_psp.h, 1366

CFE\_PSP\_MemWrite16  
    cfe\_psp.h, 1366

CFE\_PSP\_MemWrite32  
    cfe\_psp.h, 1366

CFE\_PSP\_MemWrite8  
    cfe\_psp.h, 1366

CFE\_PSP\_NO\_EXCEPTION\_DATA  
    cfe\_psp\_error.h, 1370

CFE\_PSP\_Panic  
    cfe\_psp.h, 1366

CFE\_PSP\_PANIC\_CORE\_APP  
    cfe\_psp.h, 1358

CFE\_PSP\_PANIC\_GENERAL\_FAILURE  
    cfe\_psp.h, 1358

CFE\_PSP\_PANIC\_MEMORY\_ALLOC  
    cfe\_psp.h, 1358

CFE\_PSP\_PANIC\_NONVOL\_DISK  
    cfe\_psp.h, 1358

CFE\_PSP\_PANIC\_STARTUP  
    cfe\_psp.h, 1359

CFE\_PSP\_PANIC\_STARTUP\_SEM  
    cfe\_psp.h, 1359

CFE\_PSP\_PANIC\_VOLATILE\_DISK  
    cfe\_psp.h, 1359

CFE\_PSP\_PortRead16  
    cfe\_psp.h, 1367

CFE\_PSP\_PortRead32  
    cfe\_psp.h, 1367

CFE\_PSP\_PortRead8  
    cfe\_psp.h, 1367

CFE\_PSP\_PortWrite16  
    cfe\_psp.h, 1367

CFE\_PSP\_PortWrite32  
    cfe\_psp.h, 1367

CFE\_PSP\_PortWrite8  
    cfe\_psp.h, 1367

CFE\_PSP\_ReadFromCDS  
    cfe\_psp.h, 1367

CFE\_PSP\_Restart  
    cfe\_psp.h, 1367

CFE\_PSP\_RST\_SUBTYPE\_BANKSWITCH\_RESET  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_HW\_SPECIAL\_COMMAND  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_HW\_WATCHDOG  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_HWDEBUG\_RESET  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_MAX  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_POWER\_CYCLE  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_PUSH\_BUTTON  
    cfe\_psp.h, 1359

CFE\_PSP\_RST\_SUBTYPE\_RESET\_COMMAND  
    cfe\_psp.h, 1360

CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET  
    cfe\_psp.h, 1360

CFE\_PSP\_RST\_TYPE\_MAX  
     cfe\_psp.h, 1360

CFE\_PSP\_RST\_TYPE\_POWERON  
     cfe\_psp.h, 1360

CFE\_PSP\_RST\_TYPE\_PROCESSOR  
     cfe\_psp.h, 1360

CFE\_PSP\_SetDefaultExceptionEnvironment  
     cfe\_psp.h, 1367

CFE\_PSP\_SOFT\_TIMEBASE\_NAME  
     cfe\_psp.h, 1360

CFE\_PSP\_STATUS\_C  
     cfe\_psp\_error.h, 1370

CFE\_PSP\_STATUS\_STRING\_LENGTH  
     cfe\_psp\_error.h, 1370

CFE\_PSP\_Status\_t  
     cfe\_psp\_error.h, 1370

CFE\_PSP\_StatusString\_t  
     cfe\_psp\_error.h, 1370

CFE\_PSP\_StatusToString  
     cfe\_psp\_error.h, 1371

CFE\_PSP\_SUCCESS  
     cfe\_psp\_error.h, 1370

CFE\_PSP\_WatchdogDisable  
     cfe\_psp.h, 1367

CFE\_PSP\_WatchdogEnable  
     cfe\_psp.h, 1367

CFE\_PSP\_WatchdogGet  
     cfe\_psp.h, 1368

CFE\_PSP\_WatchdogInit  
     cfe\_psp.h, 1368

CFE\_PSP\_WatchdogService  
     cfe\_psp.h, 1368

CFE\_PSP\_WatchdogSet  
     cfe\_psp.h, 1368

CFE\_PSP\_WriteToCDS  
     cfe\_psp.h, 1368

cfe\_resourceid.h  
     CFE\_Resourceld\_Equal, 1011  
     CFE\_Resourceld\_FindNext, 1012  
     CFE\_Resourceld\_FromInteger, 1012  
     CFE\_Resourceld\_GetBase, 1013  
     CFE\_Resourceld\_GetSerial, 1013  
     CFE\_Resourceld\_IsDefined, 1013  
     CFE\_RESOURCEID\_TEST\_DEFINED, 1011  
     CFE\_RESOURCEID\_TEST\_EQUAL, 1011  
     CFE\_RESOURCEID\_TO ULONG, 1011  
     CFE\_Resourceld\_TolIndex, 1014  
     CFE\_Resourceld\_ToInteger, 1014

cfe\_resourceid\_api\_typedefs.h  
     CFE\_RESOURCEID\_RESERVED, 1015  
     CFE\_RESOURCEID\_UNDEFINED, 1016

cfe\_resourceid\_basevalue.h  
     CFE\_RESOURCEID\_MAKE\_BASE, 1172  
     CFE\_RESOURCEID\_MAX, 1172

CFE\_RESOURCEID\_SHIFT, 1172

CFE\_RESOURCEID\_CONFIGID\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_Resourceld\_Equal  
     cfe\_resourceid.h, 1011

CFE\_RESOURCEID\_ES\_APPID\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_RESOURCEID\_ES\_CDSBLOCKID\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_RESOURCEID\_ES\_COUNTID\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_RESOURCEID\_ES\_LIBID\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_RESOURCEID\_ES\_POOLID\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_RESOURCEID\_ES\_TASKID\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_Resourceld\_FindNext  
     cfe\_resourceid.h, 1012

CFE\_Resourceld\_FromInteger  
     cfe\_resourceid.h, 1012

CFE\_Resourceld\_GetBase  
     cfe\_resourceid.h, 1013

CFE\_Resourceld\_GetSerial  
     cfe\_resourceid.h, 1013

CFE\_Resourceld\_IsDefined  
     cfe\_resourceid.h, 1013

CFE\_RESOURCEID\_MAKE\_BASE  
     cfe\_resourceid\_basevalue.h, 1172

CFE\_RESOURCEID\_MAX  
     cfe\_resourceid\_basevalue.h, 1172

CFE\_RESOURCEID\_RESERVED  
     cfe\_resourceid\_api\_typedefs.h, 1015

CFE\_RESOURCEID\_SB\_PIPEID\_RESOURCE\_BASE\_OFFSET  
     cFE Resource ID base values, 397

CFE\_RESOURCEID\_SHIFT  
     cfe\_resourceid\_basevalue.h, 1172

CFE\_RESOURCEID\_TEST\_DEFINED  
     cfe\_resourceid.h, 1011

CFE\_RESOURCEID\_TEST\_EQUAL  
     cfe\_resourceid.h, 1011

CFE\_RESOURCEID\_TO ULONG  
     cfe\_resourceid.h, 1011

CFE\_Resourceld\_TolIndex  
     cfe\_resourceid.h, 1014

CFE\_Resourceld\_ToInteger  
     cfe\_resourceid.h, 1014

CFE\_RESOURCEID\_UNDEFINED  
     cfe\_resourceid\_api\_typedefs.h, 1016

CFE\_REVISION  
     cfe\_version.h, 1031

cfe\_sb.h  
     CFE\_BIT, 1018  
     CFE\_CLR, 1018

CFE\_SET, 1018  
CFE\_TST, 1018  
CFE\_SB\_AllocateMessageBuffer  
  cFE Zero Copy APIs, 346  
CFE\_SB\_ALLSUBS\_TLM\_MID  
  default\_cfe\_sb\_msgids.h, 1195  
CFE\_SB\_AllSubscriptionsTlm, 587  
  Payload, 587  
  TelemetryHeader, 588  
CFE\_SB\_AllSubscriptionsTlm\_Payload, 588  
  Entries, 588  
  Entry, 588  
  PktSegment, 588  
  TotalSegments, 589  
CFE\_SB\_AllSubscriptionsTlm\_Payload\_t  
  default\_cfe\_sb\_msgstruct.h, 1197  
CFE\_SB\_AllSubscriptionsTlm\_t  
  default\_cfe\_sb\_msgstruct.h, 1197  
cfe\_sb\_api\_typedefs.h  
  CFE\_SB\_Buffer\_t, 1021  
  CFE\_SB\_DEFAULT\_QOS, 1019  
  CFE\_SB\_INVALID\_MSG\_ID, 1019  
  CFE\_SB\_INVALID\_PIPE, 1019  
  CFE\_SB\_MSGID\_C, 1020  
  CFE\_SB\_MSGID\_RESERVED, 1020  
  CFE\_SB\_MSGID\_UNWRAP\_VALUE, 1020  
  CFE\_SB\_MSGID\_WRAP\_VALUE, 1020  
  CFE\_SB\_PEND\_FOREVER, 1020  
  CFE\_SB PIPEID\_C, 1021  
  CFE\_SB\_POLL, 1021  
  CFE\_SB\_SUBSCRIPTION, 1021  
  CFE\_SB\_UNSUBSCRIPTION, 1021  
CFE\_SB\_BAD\_ARGUMENT  
  cFE Return Code Defines, 229  
CFE\_SB\_BAD\_CMD\_CODE\_EID  
  cfe\_sb\_eventids.h, 1204  
CFE\_SB\_BAD\_MSGID\_EID  
  cfe\_sb\_eventids.h, 1204  
CFE\_SB\_BAD\_PIPEID\_EID  
  cfe\_sb\_eventids.h, 1204  
CFE\_SB\_BUF\_ALOC\_ERR  
  cFE Return Code Defines, 229  
CFE\_SB\_BUFFER\_INVALID  
  cFE Return Code Defines, 230  
CFE\_SB\_Buffer\_t  
  cfe\_sb\_api\_typedefs.h, 1021  
CFE\_SB\_CMD0\_RCVD\_EID  
  cfe\_sb\_eventids.h, 1205  
CFE\_SB\_CMD1\_RCVD\_EID  
  cfe\_sb\_eventids.h, 1205  
CFE\_SB\_CMD\_MID  
  default\_cfe\_sb\_msgids.h, 1195  
CFE\_SB\_CR\_PIPE\_BAD\_ARG\_EID  
  cfe\_sb\_eventids.h, 1205  
CFE\_SB\_CR\_PIPE\_ERR\_EID  
  cfe\_sb\_eventids.h, 1205  
CFE\_SB\_CR\_PIPE\_NAME\_TAKEN\_EID  
  cfe\_sb\_eventids.h, 1206  
CFE\_SB\_CR\_PIPE\_NO\_FREE\_EID  
  cfe\_sb\_eventids.h, 1206  
CFE\_SB\_CreatePipe  
  cFE Pipe Management APIs, 333  
CFE\_SB\_DEFAULT\_QOS  
  cfe\_sb\_api\_typedefs.h, 1019  
CFE\_SB\_DEL\_PIPE\_ERR1\_EID  
  cfe\_sb\_eventids.h, 1206  
CFE\_SB\_DEL\_PIPE\_ERR2\_EID  
  cfe\_sb\_eventids.h, 1206  
CFE\_SB\_DeletePipe  
  cFE Pipe Management APIs, 334  
CFE\_SB\_DEST\_BLK\_ERR\_EID  
  cfe\_sb\_eventids.h, 1207  
CFE\_SB\_DISABLE\_ROUTE\_CC  
  default\_cfe\_sb\_fcncodes.h, 1175  
CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC  
  default\_cfe\_sb\_fcncodes.h, 1175  
CFE\_SB\_DisableRouteCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1197  
CFE\_SB\_DisableSubReportingCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1197  
CFE\_SB\_DSBL\_RTE1\_EID  
  cfe\_sb\_eventids.h, 1207  
CFE\_SB\_DSBL\_RTE2\_EID  
  cfe\_sb\_eventids.h, 1207  
CFE\_SB\_DSBL\_RTE3\_EID  
  cfe\_sb\_eventids.h, 1207  
CFE\_SB\_DUP\_SUBSCRIP\_EID  
  cfe\_sb\_eventids.h, 1208  
CFE\_SB\_ENABLE\_ROUTE\_CC  
  default\_cfe\_sb\_fcncodes.h, 1176  
CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC  
  default\_cfe\_sb\_fcncodes.h, 1177  
CFE\_SB\_EnableRouteCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1197  
CFE\_SB\_EnableSubReportingCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1197  
CFE\_SB\_ENBL\_RTE1\_EID  
  cfe\_sb\_eventids.h, 1208  
CFE\_SB\_ENBL\_RTE2\_EID  
  cfe\_sb\_eventids.h, 1208  
CFE\_SB\_ENBL\_RTE3\_EID  
  cfe\_sb\_eventids.h, 1208  
cfe\_sb\_eventids.h  
  CFE\_SB\_BAD\_CMD\_CODE\_EID, 1204  
  CFE\_SB\_BAD\_MSGID\_EID, 1204  
  CFE\_SB\_BAD\_PIPEID\_EID, 1204  
  CFE\_SB\_CMD0\_RCVD\_EID, 1205  
  CFE\_SB\_CMD1\_RCVD\_EID, 1205

CFE\_SB\_CR\_PIPE\_BAD\_ARG\_EID, [1205](#)  
 CFE\_SB\_CR\_PIPE\_ERR\_EID, [1205](#)  
 CFE\_SB\_CR\_PIPE\_NAME\_TAKEN\_EID, [1206](#)  
 CFE\_SB\_CR\_PIPE\_NO\_FREE\_EID, [1206](#)  
 CFE\_SB\_DEL\_PIPE\_ERR1\_EID, [1206](#)  
 CFE\_SB\_DEL\_PIPE\_ERR2\_EID, [1206](#)  
 CFE\_SB\_DEST\_BLK\_ERR\_EID, [1207](#)  
 CFE\_SB\_DSBL RTE1\_EID, [1207](#)  
 CFE\_SB\_DSBL RTE2\_EID, [1207](#)  
 CFE\_SB\_DSBL RTE3\_EID, [1207](#)  
 CFE\_SB\_DUP\_SUBSCRIP\_EID, [1208](#)  
 CFE\_SB\_ENBL RTE1\_EID, [1208](#)  
 CFE\_SB\_ENBL RTE2\_EID, [1208](#)  
 CFE\_SB\_ENBL RTE3\_EID, [1208](#)  
 CFE\_SB\_FILEWRITE\_ERR\_EID, [1209](#)  
 CFE\_SB\_FULL\_SUB\_PKT\_EID, [1209](#)  
 CFE\_SB\_GET\_BUF\_ERR\_EID, [1209](#)  
 CFE\_SB\_GETPIPEIDBYNAME\_EID, [1209](#)  
 CFE\_SB\_GETPIPEIDBYNAME\_NAME\_ERR\_EID, [1210](#)  
 CFE\_SB\_GETPIPEIDBYNAME\_NULL\_ERR\_EID, [1210](#)  
 CFE\_SB\_GETPIPENAME\_EID, [1210](#)  
 CFE\_SB\_GETPIPENAME\_ID\_ERR\_EID, [1210](#)  
 CFE\_SB\_GETPIPENAME\_NULL\_PTR\_EID, [1211](#)  
 CFE\_SB\_GETPIPEOPTS\_EID, [1211](#)  
 CFE\_SB\_GETPIPEOPTS\_ID\_ERR\_EID, [1211](#)  
 CFE\_SB\_GETPIPEOPTS\_PTR\_ERR\_EID, [1211](#)  
 CFE\_SB\_HASHCOLLISION\_EID, [1212](#)  
 CFE\_SB\_INIT\_EID, [1212](#)  
 CFE\_SB\_LEN\_ERR\_EID, [1212](#)  
 CFE\_SB\_MAX\_DESTS\_MET\_EID, [1212](#)  
 CFE\_SB\_MAX\_MSGS\_MET\_EID, [1213](#)  
 CFE\_SB\_MAX\_PIPES\_MET\_EID, [1213](#)  
 CFE\_SB\_MSG\_TOO\_BIG\_EID, [1213](#)  
 CFE\_SB\_MSGID\_LIM\_ERR\_EID, [1213](#)  
 CFE\_SB\_PART\_SUB\_PKT\_EID, [1214](#)  
 CFE\_SB\_PIPE\_ADDED\_EID, [1214](#)  
 CFE\_SB\_PIPE\_DELETED\_EID, [1214](#)  
 CFE\_SB\_Q\_FULL\_ERR\_EID, [1214](#)  
 CFE\_SB\_Q\_RD\_ERR\_EID, [1215](#)  
 CFE\_SB\_Q\_WR\_ERR\_EID, [1215](#)  
 CFE\_SB\_RCV\_BAD\_ARG\_EID, [1215](#)  
 CFE\_SB\_SEND\_BAD\_ARG\_EID, [1215](#)  
 CFE\_SB\_SEND\_INV\_MSGID\_EID, [1216](#)  
 CFE\_SB\_SEND\_NO\_SUBS\_EID, [1216](#)  
 CFE\_SB\_SETPIPEOPTS\_EID, [1216](#)  
 CFE\_SB\_SETPIPEOPTS\_ID\_ERR\_EID, [1216](#)  
 CFE\_SB\_SETPIPEOPTS\_OWNER\_ERR\_EID, [1217](#)  
 CFE\_SB SND RTG\_EID, [1217](#)  
 CFE\_SB SND RTG\_ERR1\_EID, [1217](#)  
 CFE\_SB SND STATS\_EID, [1217](#)  
 CFE\_SB\_SUB\_ARG\_ERR\_EID, [1218](#)  
 CFE\_SB\_SUB\_INV\_CALLER\_EID, [1218](#)  
 CFE\_SB\_SUB\_INV\_PIPE\_EID, [1218](#)  
 CFE\_SB\_SUBSCRIPTION\_RCVD\_EID, [1218](#)  
 CFE\_SB\_SUBSCRIPTION\_REMOVED\_EID, [1219](#)  
 CFE\_SB\_SUBSCRIPTION\_RPT\_EID, [1219](#)  
 CFE\_SB\_UNSUB\_ARG\_ERR\_EID, [1219](#)  
 CFE\_SB\_UNSUB\_INV\_CALLER\_EID, [1219](#)  
 CFE\_SB\_UNSUB\_INV\_PIPE\_EID, [1220](#)  
 CFE\_SB\_UNSUB\_NO\_SUBS\_EID, [1220](#)  
 CFE\_SB\_FILEWRITE\_ERR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1209](#)  
 CFE\_SB\_FULL\_SUB\_PKT\_EID  
     [cfe\\_sb\\_eventids.h](#), [1209](#)  
 CFE\_SB\_GET\_BUF\_ERR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1209](#)  
 CFE\_SB\_GetPipeByName  
     cFE Pipe Management APIs, [334](#)  
 CFE\_SB\_GETPIPEIDBYNAME\_EID  
     [cfe\\_sb\\_eventids.h](#), [1209](#)  
 CFE\_SB\_GETPIPEIDBYNAME\_NAME\_ERR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1210](#)  
 CFE\_SB\_GETPIPEIDBYNAME\_NULL\_ERR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1210](#)  
 CFE\_SB\_GetPipeName  
     cFE Pipe Management APIs, [335](#)  
 CFE\_SB\_GETPIPENAME\_EID  
     [cfe\\_sb\\_eventids.h](#), [1210](#)  
 CFE\_SB\_GETPIPENAME\_ID\_ERR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1210](#)  
 CFE\_SB\_GETPIPENAME\_NULL\_PTR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1211](#)  
 CFE\_SB\_GetPipeOpts  
     cFE Pipe Management APIs, [336](#)  
 CFE\_SB\_GETPIPEOPTS\_EID  
     [cfe\\_sb\\_eventids.h](#), [1211](#)  
 CFE\_SB\_GETPIPEOPTS\_ID\_ERR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1211](#)  
 CFE\_SB\_GETPIPEOPTS\_PTR\_ERR\_EID  
     [cfe\\_sb\\_eventids.h](#), [1211](#)  
 CFE\_SB\_GetUserData  
     cFE Message Characteristics APIs, [349](#)  
 CFE\_SB\_GetUserDataLength  
     cFE Message Characteristics APIs, [349](#)  
 CFE\_SB\_HASHCOLLISION\_EID  
     [cfe\\_sb\\_eventids.h](#), [1212](#)  
 CFE\_SB\_HK\_TLM\_MID  
     [default\\_cfe\\_sb\\_msgids.h](#), [1195](#)  
 CFE\_SB\_HousekeepingTlm, [589](#)  
     Payload, [589](#)  
     TelemetryHeader, [589](#)  
 CFE\_SB\_HousekeepingTlm\_Payload, [589](#)  
     CommandCounter, [590](#)  
     CommandErrorCounter, [590](#)  
     CreatePipeErrorCounter, [591](#)  
     DuplicateSubscriptionsCounter, [591](#)

GetPipeIdByNameErrorCounter, 591  
InternalErrorCounter, 591  
MemInUse, 591  
MemPoolHandle, 591  
MsgLimitErrorCounter, 591  
MsgReceiveErrorCounter, 592  
MsgSendErrorCounter, 592  
NoSubscribersCounter, 592  
PipeOptsErrorCounter, 592  
PipeOverflowErrorCounter, 592  
Spare2Align, 592  
SubscribeErrorCounter, 592  
UnmarkedMem, 593  
CFE\_SB\_HousekeepingTlm\_Payload\_t  
  default\_cfe\_sb\_msgstruct.h, 1197  
CFE\_SB\_HousekeepingTlm\_t  
  default\_cfe\_sb\_msgstruct.h, 1198  
CFE\_SB\_INIT\_EID  
  cfe\_sb\_eventids.h, 1212  
CFE\_SB\_INTERNAL\_ERR  
  cFE Return Code Defines, 230  
CFE\_SB\_INVALID\_MSG\_ID  
  cfe\_sb\_api\_typedefs.h, 1019  
CFE\_SB\_INVALID\_PIPE  
  cfe\_sb\_api\_typedefs.h, 1019  
CFE\_SB\_IsValidMsgId  
  cFE Message ID APIs, 354  
CFE\_SB\_LEN\_ERR\_EID  
  cfe\_sb\_eventids.h, 1212  
CFE\_SB\_MAX\_DESTS\_MET  
  cFE Return Code Defines, 230  
CFE\_SB\_MAX\_DESTS\_MET\_EID  
  cfe\_sb\_eventids.h, 1212  
CFE\_SB\_MAX\_MSGS\_MET  
  cFE Return Code Defines, 230  
CFE\_SB\_MAX\_MSGS\_MET\_EID  
  cfe\_sb\_eventids.h, 1213  
CFE\_SB\_MAX\_PIPES\_MET  
  cFE Return Code Defines, 230  
CFE\_SB\_MAX\_PIPES\_MET\_EID  
  cfe\_sb\_eventids.h, 1213  
CFE\_SB\_MessageStringGet  
  cFE Message Characteristics APIs, 350  
CFE\_SB\_MessageStringSet  
  cFE Message Characteristics APIs, 351  
CFE\_SB\_Msg, 593  
  LongDouble, 593  
  LongInt, 593  
  Msg, 593  
CFE\_SB\_MSG\_TOO\_BIG  
  cFE Return Code Defines, 230  
CFE\_SB\_MSG\_TOO\_BIG\_EID  
  cfe\_sb\_eventids.h, 1213  
CFE\_SB\_MsgId\_Atom\_t  
  default\_cfe\_sb\_extern\_typedefs.h, 1173  
CFE\_SB\_MSGID\_C  
  cfe\_sb\_api\_typedefs.h, 1020  
CFE\_SB\_MsgId\_Equal  
  cFE Message ID APIs, 354  
CFE\_SB\_MSGID\_LIM\_ERR\_EID  
  cfe\_sb\_eventids.h, 1213  
CFE\_SB\_MSGID\_RESERVED  
  cfe\_sb\_api\_typedefs.h, 1020  
CFE\_SB\_MsgId\_t, 594  
  Value, 594  
CFE\_SB\_MSGID\_UNWRAP\_VALUE  
  cfe\_sb\_api\_typedefs.h, 1020  
CFE\_SB\_MSGID\_WRAP\_VALUE  
  cfe\_sb\_api\_typedefs.h, 1020  
CFE\_SB\_MsgIdToValue  
  cFE Message ID APIs, 355  
CFE\_SB\_MsgMapFileEntry, 594  
  Index, 595  
  MsgId, 595  
CFE\_SB\_MsgMapFileEntry\_t  
  default\_cfe\_sb\_msgstruct.h, 1198  
CFE\_SB\_NO\_MESSAGE  
  cFE Return Code Defines, 230  
CFE\_SB\_NOOP\_CC  
  default\_cfe\_sb\_fcncodes.h, 1178  
CFE\_SB\_NoopCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1198  
CFE\_SB\_NOT\_IMPLEMENTED  
  cFE Return Code Defines, 231  
CFE\_SB\_ONESUB\_TLM\_MID  
  default\_cfe\_sb\_msgids.h, 1195  
CFE\_SB\_PART\_SUB\_PKT\_EID  
  cfe\_sb\_eventids.h, 1214  
CFE\_SB\_PEND\_FOREVER  
  cfe\_sb\_api\_typedefs.h, 1020  
CFE\_SB\_PIPE\_ADDED\_EID  
  cfe\_sb\_eventids.h, 1214  
CFE\_SB\_PIPE\_CR\_ERR  
  cFE Return Code Defines, 231  
CFE\_SB\_PIPE\_DELETED\_EID  
  cfe\_sb\_eventids.h, 1214  
CFE\_SB\_PIPE\_RD\_ERR  
  cFE Return Code Defines, 231  
CFE\_SB\_PipeDepthStats, 595  
  CurrentQueueDepth, 595  
  MaxQueueDepth, 595  
  PeakQueueDepth, 596  
  PipeId, 596  
  Spare, 596  
CFE\_SB\_PipeDepthStats\_t  
  default\_cfe\_sb\_msgstruct.h, 1198  
CFE\_SB\_PIPEID\_BASE  
  cFE Resource ID base values, 398

CFE\_SB\_PIPEID\_C  
     cfe\_sb\_api\_typedefs.h, 1021

CFE\_SB\_Pipeld\_t  
     default\_cfe\_sb\_extern\_typedefs.h, 1173

CFE\_SB\_Pipeld\_ToIndex  
     cFE Pipe Management APIs, 336

CFE\_SB\_PipeInfoEntry, 596  
     Appld, 597  
     AppName, 597  
     CurrentQueueDepth, 597  
     MaxQueueDepth, 597  
     Opts, 597  
     PeakQueueDepth, 597  
     Pipeld, 597  
     PipeName, 597  
     SendErrors, 597  
     Spare, 598

CFE\_SB\_PipeInfoEntry\_t  
     default\_cfe\_sb\_msgstruct.h, 1198

CFE\_SB\_PIPEOPTS\_IGNOREMINE  
     cFE SB Pipe options, 356

CFE\_SB\_POLL  
     cfe\_sb\_api\_typedefs.h, 1021

CFE\_SB\_Q\_FULL\_ERR\_EID  
     cfe\_sb\_eventids.h, 1214

CFE\_SB\_Q\_RD\_ERR\_EID  
     cfe\_sb\_eventids.h, 1215

CFE\_SB\_Q\_WR\_ERR\_EID  
     cfe\_sb\_eventids.h, 1215

CFE\_SB\_Qos\_t, 598  
     Priority, 598  
     Reliability, 598

CFE\_SB\_QoSPriority  
     default\_cfe\_sb\_extern\_typedefs.h, 1174

CFE\_SB\_QoSPriority\_Enum\_t  
     default\_cfe\_sb\_extern\_typedefs.h, 1173

CFE\_SB\_QoSPriority\_HIGH  
     default\_cfe\_sb\_extern\_typedefs.h, 1174

CFE\_SB\_QoSPriority\_LOW  
     default\_cfe\_sb\_extern\_typedefs.h, 1174

CFE\_SB\_QoSReliability  
     default\_cfe\_sb\_extern\_typedefs.h, 1174

CFE\_SB\_QoSReliability\_Enum\_t  
     default\_cfe\_sb\_extern\_typedefs.h, 1173

CFE\_SB\_QoSReliability\_HIGH  
     default\_cfe\_sb\_extern\_typedefs.h, 1174

CFE\_SB\_QoSReliability\_LOW  
     default\_cfe\_sb\_extern\_typedefs.h, 1174

CFE\_SB\_RCV\_BAD\_ARG\_EID  
     cfe\_sb\_eventids.h, 1215

CFE\_SB\_ReceiveBuffer  
     cFE Send/Receive Message APIs, 343

CFE\_SB\_ReleaseMessageBuffer  
     cFE Zero Copy APIs, 346

CFE\_SB\_RESET\_COUNTERS\_CC  
     default\_cfe\_sb\_fcncodes.h, 1178

CFE\_SB\_ResetCountersCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1198

CFE\_SB\_RouteCmd, 598  
     CommandHeader, 599  
     Payload, 599

CFE\_SB\_RouteCmd\_Payload, 599  
     MsgId, 599  
     Pipe, 600  
     Spare, 600

CFE\_SB\_RouteCmd\_Payload\_t  
     default\_cfe\_sb\_msgstruct.h, 1198

CFE\_SB\_RouteCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1198

CFE\_SB\_Routeld\_Atom\_t  
     default\_cfe\_sb\_extern\_typedefs.h, 1174

CFE\_SB\_RoutingFileEntry, 600  
     AppName, 600  
     MsgCnt, 600  
     MsgId, 601  
     Pipeld, 601  
     PipeName, 601  
     State, 601

CFE\_SB\_RoutingFileEntry\_t  
     default\_cfe\_sb\_msgstruct.h, 1198

CFE\_SB\_SEND\_BAD\_ARG\_EID  
     cfe\_sb\_eventids.h, 1215

CFE\_SB\_SEND\_HK\_MID  
     default\_cfe\_sb\_msgids.h, 1195

CFE\_SB\_SEND\_INV\_MSGID\_EID  
     cfe\_sb\_eventids.h, 1216

CFE\_SB\_SEND\_NO\_SUBS\_EID  
     cfe\_sb\_eventids.h, 1216

CFE\_SB\_SEND\_PREV\_SUBS\_CC  
     default\_cfe\_sb\_fcncodes.h, 1179

CFE\_SB\_SEND\_SB\_STATS\_CC  
     default\_cfe\_sb\_fcncodes.h, 1180

CFE\_SB\_SendHkCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1198

CFE\_SB\_SendPrevSubsCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1198

CFE\_SB\_SendSbStatsCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1199

CFE\_SB\_SetPipeOpts  
     cFE Pipe Management APIs, 337

CFE\_SB\_SETPIPEOPTS\_EID  
     cfe\_sb\_eventids.h, 1216

CFE\_SB\_SETPIPEOPTS\_ID\_ERR\_EID  
     cfe\_sb\_eventids.h, 1216

CFE\_SB\_SETPIPEOPTS\_OWNER\_ERR\_EID  
     cfe\_sb\_eventids.h, 1217

CFE\_SB\_SetUserDataLength  
     cFE Message Characteristics APIs, 352

CFE\_SB\_SingleSubscriptionTlm, 601  
    Payload, 601  
    TelemetryHeader, 601  
CFE\_SB\_SingleSubscriptionTlm\_Payload, 602  
    MsgId, 602  
    Pipe, 602  
    Qos, 602  
    SubType, 602  
CFE\_SB\_SingleSubscriptionTlm\_Payload\_t  
    default\_cfe\_sb\_msgstruct.h, 1199  
CFE\_SB\_SingleSubscriptionTlm\_t  
    default\_cfe\_sb\_msgstruct.h, 1199  
CFE\_SB SND RTG EID  
    cfe\_sb\_eventids.h, 1217  
CFE\_SB SND RTG\_ERR1 EID  
    cfe\_sb\_eventids.h, 1217  
CFE\_SB SND STATS EID  
    cfe\_sb\_eventids.h, 1217  
CFE\_SB STATS TLM MID  
    default\_cfe\_sb\_msgids.h, 1195  
CFE\_SB\_StatsTlm, 603  
    Payload, 603  
    TelemetryHeader, 603  
CFE\_SB\_StatsTlm\_Payload, 603  
    MaxMemAllowed, 604  
    MaxMsgIdsAllowed, 604  
    MaxPipeDepthAllowed, 604  
    MaxPipesAllowed, 605  
    MaxSubscriptionsAllowed, 605  
    MemInUse, 605  
    MsgIdsInUse, 605  
    PeakMemInUse, 605  
    PeakMsgIdsInUse, 605  
    PeakPipesInUse, 605  
    PeakSBBuffersInUse, 606  
    PeakSubscriptionsInUse, 606  
    PipeDepthStats, 606  
    PipesInUse, 606  
    SBBuffersInUse, 606  
    SubscriptionsInUse, 606  
CFE\_SB\_StatsTlm\_Payload\_t  
    default\_cfe\_sb\_msgstruct.h, 1199  
CFE\_SB\_StatsTlm\_t  
    default\_cfe\_sb\_msgstruct.h, 1199  
CFE\_SB\_SUB\_ARG\_ERR\_EID  
    cfe\_sb\_eventids.h, 1218  
CFE\_SB\_SUB\_ENTRIES\_PER\_PKT  
    default\_cfe\_sb\_extern\_typedefs.h, 1173  
CFE\_SB\_SUB\_INV\_CALLER\_EID  
    cfe\_sb\_eventids.h, 1218  
CFE\_SB\_SUB\_INV\_PIPE\_EID  
    cfe\_sb\_eventids.h, 1218  
CFE\_SB\_SUB\_RPT\_CTRL\_MID  
    default\_cfe\_sb\_msgids.h, 1195  
CFE\_SB\_SubEntries, 607  
    MsgId, 607  
    Pipe, 607  
    Qos, 607  
CFE\_SB\_SubEntries\_t  
    default\_cfe\_sb\_msgstruct.h, 1199  
CFE\_SB\_Subscribe  
    cFE Message Subscription Control APIs, 338  
CFE\_SB\_SubscribeEx  
    cFE Message Subscription Control APIs, 339  
CFE\_SB\_SubscribeLocal  
    cFE Message Subscription Control APIs, 340  
CFE\_SB\_SUBSCRIPTION  
    cfe\_sb\_api\_typedefs.h, 1021  
CFE\_SB\_SUBSCRIPTION\_RCVD\_EID  
    cfe\_sb\_eventids.h, 1218  
CFE\_SB\_SUBSCRIPTION\_REMOVED\_EID  
    cfe\_sb\_eventids.h, 1219  
CFE\_SB\_SUBSCRIPTION\_RPT\_EID  
    cfe\_sb\_eventids.h, 1219  
CFE\_SB\_TIME\_OUT  
    cFE Return Code Defines, 231  
CFE\_SB\_TimeStampMsg  
    cFE Message Characteristics APIs, 352  
CFE\_SB\_TransmitBuffer  
    cFE Zero Copy APIs, 347  
CFE\_SB\_TransmitMsg  
    cFE Send/Receive Message APIs, 344  
CFE\_SB\_UNSUB\_ARG\_ERR\_EID  
    cfe\_sb\_eventids.h, 1219  
CFE\_SB\_UNSUB\_INV\_CALLER\_EID  
    cfe\_sb\_eventids.h, 1219  
CFE\_SB\_UNSUB\_INV\_PIPE\_EID  
    cfe\_sb\_eventids.h, 1220  
CFE\_SB\_UNSUB\_NO\_SUBS\_EID  
    cfe\_sb\_eventids.h, 1220  
CFE\_SB\_Unsubscribe  
    cFE Message Subscription Control APIs, 340  
CFE\_SB\_UnsubscribeLocal  
    cFE Message Subscription Control APIs, 341  
CFE\_SB\_UNSUBSCRIPTION  
    cfe\_sb\_api\_typedefs.h, 1021  
CFE\_SB\_ValueToMsgId  
    cFE Message ID APIs, 355  
CFE\_SB\_WRITE\_MAP\_INFO\_CC  
    default\_cfe\_sb\_fcncodes.h, 1181  
CFE\_SB\_WRITE\_PIPE\_INFO\_CC  
    default\_cfe\_sb\_fcncodes.h, 1182  
CFE\_SB\_WRITE\_ROUTING\_INFO\_CC  
    default\_cfe\_sb\_fcncodes.h, 1183  
CFE\_SB\_WriteFileInfoCmd, 607  
    CommandHeader, 608  
    Payload, 608  
CFE\_SB\_WriteFileInfoCmd\_Payload, 608

Filename, 608  
**CFE\_SB\_WriteFileInfoCmd\_Payload\_t**  
 default\_cfe\_sb\_msgstruct.h, 1199  
**CFE\_SB\_WriteFileInfoCmd\_t**  
 default\_cfe\_sb\_msgstruct.h, 1199  
**CFE\_SB\_WriteMapInfoCmd\_t**  
 default\_cfe\_sb\_msgstruct.h, 1199  
**CFE\_SB\_WritePipeInfoCmd\_t**  
 default\_cfe\_sb\_msgstruct.h, 1199  
**CFE\_SB\_WriteRoutingInfoCmd\_t**  
 default\_cfe\_sb\_msgstruct.h, 1200  
**CFE\_SB\_WRONG\_MSG\_TYPE**  
 cFE Return Code Defines, 231  
**CFE\_SERVICE\_BITMASK**  
 cfe\_error.h, 986  
**CFE\_SET**  
 cfe\_sb.h, 1018  
**CFE\_SEVERITY\_BITMASK**  
 cfe\_error.h, 986  
**CFE\_SEVERITY\_ERROR**  
 cfe\_error.h, 986  
**CFE\_SEVERITY\_INFO**  
 cfe\_error.h, 986  
**CFE\_SEVERITY\_SUCCESS**  
 cfe\_error.h, 986  
**CFE\_SOFTWARE\_BUS\_SERVICE**  
 cfe\_error.h, 987  
**CFE\_SRC\_VERSION**  
 cfe\_version.h, 1031  
**CFE\_STATUS\_BAD\_COMMAND\_CODE**  
 cFE Return Code Defines, 231  
**CFE\_STATUS\_C**  
 cfe\_error.h, 987  
**CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL**  
 cFE Return Code Defines, 231  
**CFE\_STATUS\_INCORRECT\_STATE**  
 cFE Return Code Defines, 232  
**CFE\_STATUS\_NO\_COUNTER\_INCREMENT**  
 cFE Return Code Defines, 232  
**CFE\_STATUS\_NOT\_IMPLEMENTED**  
 cFE Return Code Defines, 232  
**CFE\_STATUS\_RANGE\_ERROR**  
 cFE Return Code Defines, 232  
**CFE\_STATUS\_REQUEST\_ALREADY\_PENDING**  
 cFE Return Code Defines, 232  
**CFE\_STATUS\_STRING\_LENGTH**  
 cfe\_error.h, 987  
**CFE\_Status\_t**  
 cfe\_error.h, 987  
**CFE\_STATUS\_UNKNOWN\_MSG\_ID**  
 cFE Return Code Defines, 232  
**CFE\_STATUS\_VALIDATION\_FAILURE**  
 cFE Return Code Defines, 232  
**CFE\_STATUS\_WRONG\_MSG\_LENGTH**  
 cfe\_error.h, 987  
**cFE Return Code Defines, 233**  
**CFE\_StatusString\_t**  
 cfe\_error.h, 987  
**CFE\_STR**  
 cfe\_version.h, 1031  
**CFE\_STR\_HELPER**  
 cfe\_version.h, 1031  
**CFE\_SUCCESS**  
 cFE Return Code Defines, 233  
**CFE\_TABLE\_SERVICE**  
 cfe\_error.h, 987  
**CFE\_TBL\_ABORT\_LOAD\_CC**  
 default\_cfe\_tbl\_fcncodes.h, 1222  
**CFE\_TBL\_AbortLoadCmd**, 608  
 CommandHeader, 609  
 Payload, 609  
**CFE\_TBL\_AbortLoadCmd\_Payload**, 609  
 TableName, 609  
**CFE\_TBL\_AbortLoadCmd\_Payload\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1242  
**CFE\_TBL\_AbortLoadCmd\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1242  
**CFE\_TBL\_ACTIVATE\_CC**  
 default\_cfe\_tbl\_fcncodes.h, 1223  
**CFE\_TBL\_ACTIVATE\_DUMP\_ONLY\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1248  
**CFE\_TBL\_ACTIVATE\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1248  
**CFE\_TBL\_ActivateCmd**, 610  
 CommandHeader, 610  
 Payload, 610  
**CFE\_TBL\_ActivateCmd\_Payload**, 610  
 TableName, 610  
**CFE\_TBL\_ActivateCmd\_Payload\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1242  
**CFE\_TBL\_ActivateCmd\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1242  
**cfe\_tbl\_api\_typedefs.h**  
 CFE\_TBL\_BAD\_TABLE\_HANDLE, 1024  
 CFE\_TBL\_CallbackFuncPtr\_t, 1024  
 CFE\_TBL\_Handle\_t, 1024  
 CFE\_TBL\_Info\_t, 1024  
 CFE\_TBL\_MAX\_FULL\_NAME\_LEN, 1024  
 CFE\_TBL\_SRC\_ADDRESS, 1025  
 CFE\_TBL\_SRC\_FILE, 1025  
 CFE\_TBL\_SrcEnum, 1024  
 CFE\_TBL\_SrcEnum\_t, 1024  
**CFE\_TBL\_ASSUMED\_VALID\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1248  
**CFE\_TBL\_BAD\_ARGUMENT**  
 cFE Return Code Defines, 233  
**CFE\_TBL\_BAD\_TABLE\_HANDLE**  
 cfe\_tbl\_api\_typedefs.h, 1024  
**CFE\_TBL\_BufferSelect**

default\_cfe\_tbl\_extern\_typedefs.h, [1221](#)  
CFE\_TBL\_BufferSelect\_ACTIVE  
    default\_cfe\_tbl\_extern\_typedefs.h, [1221](#)  
CFE\_TBL\_BufferSelect\_Enum\_t  
    default\_cfe\_tbl\_extern\_typedefs.h, [1221](#)  
CFE\_TBL\_BufferSelect\_INACTIVE  
    default\_cfe\_tbl\_extern\_typedefs.h, [1221](#)  
CFE\_TBL\_CallbackFuncPtr\_t  
    cfe\_tbl\_api\_typedefs.h, [1024](#)  
CFE\_TBL\_CC1\_ERR\_EID  
    cfe\_tbl\_eventids.h, [1249](#)  
CFE\_TBL\_CDS\_DELETE\_ERR\_EID  
    cfe\_tbl\_eventids.h, [1249](#)  
CFE\_TBL\_CDS\_DELETED\_INFO\_EID  
    cfe\_tbl\_eventids.h, [1249](#)  
CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID  
    cfe\_tbl\_eventids.h, [1249](#)  
CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID  
    cfe\_tbl\_eventids.h, [1250](#)  
CFE\_TBL\_CMD\_MID  
    default\_cfe\_tbl\_msgids.h, [1239](#)  
CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID  
    cfe\_tbl\_eventids.h, [1250](#)  
CFE\_TBL\_DeleteCDSCmd\_Payload, [611](#)  
    TableName, [611](#)  
CFE\_TBL\_DeleteCDSCmd\_Payload\_t  
    default\_cfe\_tbl\_msgstruct.h, [1242](#)  
CFE\_TBL\_DELETE\_CDS\_CC  
    default\_cfe\_tbl\_fcncodes.h, [1224](#)  
CFE\_TBL\_DeleteCDSCmd, [611](#)  
    CommandHeader, [612](#)  
    Payload, [612](#)  
CFE\_TBL\_DeleteCDSCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, [1242](#)  
CFE\_TBL\_DUMP\_CC  
    default\_cfe\_tbl\_fcncodes.h, [1224](#)  
CFE\_TBL\_DUMP\_PENDING\_ERR\_EID  
    cfe\_tbl\_eventids.h, [1250](#)  
CFE\_TBL\_DUMP\_REGISTRY\_CC  
    default\_cfe\_tbl\_fcncodes.h, [1225](#)  
CFE\_TBL\_DumpCmd, [612](#)  
    CommandHeader, [612](#)  
    Payload, [612](#)  
CFE\_TBL\_DumpCmd\_Payload, [612](#)  
    ActiveTableFlag, [613](#)  
    DumpFilename, [613](#)  
    TableName, [613](#)  
CFE\_TBL\_DumpCmd\_Payload\_t  
    default\_cfe\_tbl\_msgstruct.h, [1242](#)  
CFE\_TBL\_DumpCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, [1242](#)  
CFE\_TBL\_DumpRegistryCmd, [613](#)  
    CommandHeader, [614](#)  
    Payload, [614](#)

CFE\_TBL\_DumpRegistryCmd\_Payload, [614](#)  
    DumpFilename, [614](#)  
CFE\_TBL\_DumpRegistryCmd\_Payload\_t  
    default\_cfe\_tbl\_msgstruct.h, [1242](#)  
CFE\_TBL\_DumpRegistryCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, [1242](#)  
CFE\_TBL\_DumpToBuffer  
    cFE Manage Table Content APIs, [362](#)  
CFE\_TBL\_ERR\_ACCESS  
    cFE Return Code Defines, [233](#)  
CFE\_TBL\_ERR\_BAD\_CONTENT\_ID  
    cFE Return Code Defines, [233](#)  
CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID  
    cFE Return Code Defines, [233](#)  
CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID  
    cFE Return Code Defines, [233](#)  
CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID  
    cFE Return Code Defines, [233](#)  
CFE\_TBL\_ERR\_DUMP\_ONLY  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_DUPLICATE\_NOT\_OWNED  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_FILE\_TOO\_LARGE  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_HANDLES\_FULL  
    cFE Return Code Defines, [234](#)  
CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE  
    cFE Return Code Defines, [235](#)  
CFE\_TBL\_ERR\_INVALID\_HANDLE  
    cFE Return Code Defines, [235](#)  
CFE\_TBL\_ERR\_INVALID\_NAME  
    cFE Return Code Defines, [235](#)  
CFE\_TBL\_ERR\_INVALID\_OPTIONS  
    cFE Return Code Defines, [235](#)  
CFE\_TBL\_ERR\_INVALID\_SIZE  
    cFE Return Code Defines, [235](#)  
CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS  
    cFE Return Code Defines, [235](#)  
CFE\_TBL\_ERR\_LOAD\_INCOMPLETE  
    cFE Return Code Defines, [236](#)  
CFE\_TBL\_ERR\_NEVER\_LOADED  
    cFE Return Code Defines, [236](#)  
CFE\_TBL\_ERR\_NO\_ACCESS  
    cFE Return Code Defines, [236](#)  
CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL  
    cFE Return Code Defines, [236](#)

CFE\_TBL\_ERR\_NO\_STD\_HEADER  
cFE Return Code Defines, 236

CFE\_TBL\_ERR\_NO\_TBL\_HEADER  
cFE Return Code Defines, 236

CFE\_TBL\_ERR\_PARTIAL\_LOAD  
cFE Return Code Defines, 236

CFE\_TBL\_ERR\_REGISTRY\_FULL  
cFE Return Code Defines, 236

CFE\_TBL\_ERR\_SHORT\_FILE  
cFE Return Code Defines, 237

CFE\_TBL\_ERR\_UNREGISTERED  
cFE Return Code Defines, 237

cfe\_tbl\_eventids.h

- CFE\_TBL\_ACTIVATE\_DUMP\_ONLY\_ERR\_EID, 1248
- CFE\_TBL\_ACTIVATE\_ERR\_EID, 1248
- CFE\_TBL\_ASSUMED\_VALID\_INF\_EID, 1248
- CFE\_TBL\_CC1\_ERR\_EID, 1249
- CFE\_TBL\_CDS\_DELETE\_ERR\_EID, 1249
- CFE\_TBL\_CDS\_DELETED\_INFO\_EID, 1249
- CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID, 1249
- CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID, 1250
- CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID, 1250
- CFE\_TBL\_DUMP\_PENDING\_ERR\_EID, 1250
- CFE\_TBL\_FAIL\_HK\_SEND\_ERR\_EID, 1250
- CFE\_TBL\_FAIL\_NOTIFY\_SEND\_ERR\_EID, 1251
- CFE\_TBL\_FILE\_ACCESS\_ERR\_EID, 1251
- CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID, 1251
- CFE\_TBL\_FILE\_LOADED\_INF\_EID, 1251
- CFE\_TBL\_FILE\_STD\_HDR\_ERR\_EID, 1252
- CFE\_TBL\_FILE\_SUBTYPE\_ERR\_EID, 1252
- CFE\_TBL\_FILE\_TBL\_HDR\_ERR\_EID, 1252
- CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID, 1252
- CFE\_TBL\_FILE\_TYPE\_ERR\_EID, 1253
- CFE\_TBL\_HANDLE\_ACCESS\_ERR\_EID, 1253
- CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID, 1253
- CFE\_TBL\_IN\_REGISTRY\_ERR\_EID, 1253
- CFE\_TBL\_INIT\_INF\_EID, 1254
- CFE\_TBL\_LEN\_ERR\_EID, 1254
- CFE\_TBL\_LOAD\_ABORT\_ERR\_EID, 1254
- CFE\_TBL\_LOAD\_ABORT\_INF\_EID, 1254
- CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID, 1255
- CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID, 1255
- CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID, 1255
- CFE\_TBL\_LOAD\_PEND\_REQ\_INF\_EID, 1255
- CFE\_TBL\_LOAD\_SUCCESS\_INF\_EID, 1256
- CFE\_TBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID, 1256
- CFE\_TBL\_LOAD\_TYPE\_ERR\_EID, 1256
- CFE\_TBL\_LOAD\_VAL\_ERR\_EID, 1256

CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID, 1257

CFE\_TBL\_LOADING\_PENDING\_ERR\_EID, 1257

CFE\_TBL\_MID\_ERR\_EID, 1257

CFE\_TBL\_NO\_INACTIVE\_BUFFER\_ERR\_EID, 1257

CFE\_TBL\_NO\_SUCH\_TABLE\_ERR\_EID, 1258

CFE\_TBL\_NO\_WORK\_BUFFERS\_ERR\_EID, 1258

CFE\_TBL\_NOOP\_INF\_EID, 1258

CFE\_TBL\_NOT\_CRITICAL\_TBL\_ERR\_EID, 1258

CFE\_TBL\_NOT\_IN\_CRIT\_REG\_ERR\_EID, 1259

CFE\_TBL\_OVERWRITE\_DUMP\_INF\_EID, 1259

CFE\_TBL\_OVERWRITE\_REG\_DUMP\_INF\_EID, 1259

CFE\_TBL\_PARTIAL\_LOAD\_ERR\_EID, 1259

CFE\_TBL\_PROCESSOR\_ID\_ERR\_EID, 1260

CFE\_TBL\_REGISTER\_ERR\_EID, 1260

CFE\_TBL\_RESET\_INF\_EID, 1260

CFE\_TBL\_SHARE\_ERR\_EID, 1260

CFE\_TBL\_SPACECRAFT\_ID\_ERR\_EID, 1261

CFE\_TBL\_TLM\_REG\_CMD\_INF\_EID, 1261

CFE\_TBL\_TOO\_MANY\_DUMPS\_ERR\_EID, 1261

CFE\_TBL\_TOO\_MANY\_VALIDATIONS\_ERR\_EID, 1261

CFE\_TBL\_UNREGISTER\_ERR\_EID, 1262

CFE\_TBL\_UNVALIDATED\_ERR\_EID, 1262

CFE\_TBL\_UPDATE\_ERR\_EID, 1262

CFE\_TBL\_UPDATE\_SUCCESS\_INF\_EID, 1262

CFE\_TBL\_VAL\_REQ\_MADE\_INF\_EID, 1263

CFE\_TBL\_VALIDATION\_ERR\_EID, 1263

CFE\_TBL\_VALIDATION\_INF\_EID, 1263

CFE\_TBL\_WRITE\_CFE\_HDR\_ERR\_EID, 1263

CFE\_TBL\_WRITE\_DUMP\_INF\_EID, 1264

CFE\_TBL\_WRITE\_REG\_DUMP\_INF\_EID, 1264

CFE\_TBL\_WRITE\_TBL\_HDR\_ERR\_EID, 1264

CFE\_TBL\_WRITE\_TBL\_IMG\_ERR\_EID, 1264

CFE\_TBL\_WRITE\_TBL\_REG\_ERR\_EID, 1265

CFE\_TBL\_ZERO\_LENGTH\_LOAD\_ERR\_EID, 1265

CFE\_TBL\_FAIL\_HK\_SEND\_ERR\_EID

- cfe\_tbl\_eventids.h, 1250

CFE\_TBL\_FAIL\_NOTIFY\_SEND\_ERR\_EID

- cfe\_tbl\_eventids.h, 1251

CFE\_TBL\_FILE\_ACCESS\_ERR\_EID

- cfe\_tbl\_eventids.h, 1251

CFE\_TBL\_File\_Hdr, 614

- NumBytes, 615
- Offset, 615
- Reserved, 615
- TableName, 615

CFE\_TBL\_File\_Hdr\_t

- default\_cfe\_tbl\_extern\_typedefs.h, 1221

CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID

- cfe\_tbl\_eventids.h, 1251

CFE\_TBL\_FILE\_LOADED\_INF\_EID

cfe\_tbl\_eventids.h, 1251  
CFE\_TBL\_FILE\_STD\_HDR\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1252  
CFE\_TBL\_FILE\_SUBTYPE\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1252  
CFE\_TBL\_FILE\_TBL\_HDR\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1252  
CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1252  
CFE\_TBL\_TYPE\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1253  
CFE\_TBL\_FILEDEF  
    cfe\_tbl\_filedef.h, 1025  
CFE\_TBL\_FileDef, 615  
    Description, 616  
    ObjectName, 616  
    ObjectSize, 616  
    TableName, 616  
    TgtFilename, 617  
cfe\_tbl\_filedef.h  
    CFE\_TBL\_FILEDEF, 1025  
    CFE\_TBL\_FileDef\_t, 1026  
CFE\_TBL\_FileDef\_t  
    cfe\_tbl\_filedef.h, 1026  
CFE\_TBL\_GetAddress  
    cFE Access Table Content APIs, 368  
CFE\_TBL\_GetAddresses  
    cFE Access Table Content APIs, 369  
CFE\_TBL\_GetInfo  
    cFE Get Table Information APIs, 373  
CFE\_TBL\_GetStatus  
    cFE Get Table Information APIs, 374  
CFE\_TBL\_HANDLE\_ACCESS\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1253  
CFE\_TBL\_Handle\_t  
    cfe\_tbl\_api\_typedefs.h, 1024  
CFE\_TBL\_HK\_TLM\_MID  
    default\_cfe\_tbl\_msgids.h, 1239  
CFE\_TBL\_HousekeepingTlm, 617  
    Payload, 617  
    TelemetryHeader, 617  
CFE\_TBL\_HousekeepingTlm\_Payload, 617  
    ActiveBuffer, 619  
    ByteAlignPad1, 619  
    CommandCounter, 619  
    CommandErrorCounter, 619  
    FailedValCounter, 619  
    LastFileDumped, 619  
    LastFileLoaded, 619  
    LastTableLoaded, 619  
    LastUpdatedTable, 620  
    LastUpdateTime, 620  
    LastValCrc, 620  
    LastValStatus, 620  
LastValTableName, 620  
MemPoolHandle, 620  
NumFreeSharedBufs, 620  
NumLoadPending, 621  
NumTables, 621  
NumValRequests, 621  
SuccessValCounter, 621  
ValidationCounter, 621  
CFE\_TBL\_HousekeepingTlm\_Payload\_t  
    default\_cfe\_tbl\_msgstruct.h, 1242  
CFE\_TBL\_HousekeepingTlm\_t  
    default\_cfe\_tbl\_msgstruct.h, 1243  
CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1253  
CFE\_TBL\_IN\_REGISTRY\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1253  
CFE\_TBL\_Info, 621  
    Crc, 622  
    Critical, 622  
    DoubleBuffered, 622  
    DumpOnly, 622  
    FileCreateTimeSecs, 623  
    FileCreateTimeSubSecs, 623  
    LastFileLoaded, 623  
    NumUsers, 623  
    Size, 623  
    TableLoadedOnce, 623  
    TimeOfLastUpdate, 623  
    UserDefAddr, 623  
CFE\_TBL\_INFO\_DUMP\_PENDING  
    cFE Return Code Defines, 237  
CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING  
    cFE Return Code Defines, 237  
CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING  
    cFE Return Code Defines, 237  
CFE\_TBL\_INFO\_RECOVERED\_TBL  
    cFE Return Code Defines, 237  
CFE\_TBL\_Info\_t  
    cfe\_tbl\_api\_typedefs.h, 1024  
CFE\_TBL\_INFO\_TABLE\_LOCKED  
    cFE Return Code Defines, 237  
CFE\_TBL\_INFO\_UPDATE\_PENDING  
    cFE Return Code Defines, 237  
CFE\_TBL\_INFO\_UPDATED  
    cFE Return Code Defines, 238  
CFE\_TBL\_INFO\_VALIDATION\_PENDING  
    cFE Return Code Defines, 238  
CFE\_TBL\_INIT\_INF\_EID  
    cfe\_tbl\_eventids.h, 1254  
CFE\_TBL\_LEN\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1254  
CFE\_TBL\_Load  
    cFE Manage Table Content APIs, 363  
CFE\_TBL\_LOAD\_ABORT\_ERR\_EID

cfe\_tbl\_eventids.h, 1254  
**CFE\_TBL\_LOAD\_ABORT\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1254  
**CFE\_TBL\_LOAD\_CC**  
 default\_cfe\_tbl\_fcncodes.h, 1226  
**CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1255  
**CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1255  
**CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1255  
**CFE\_TBL\_LOAD\_PEND\_REQ\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1255  
**CFE\_TBL\_LOAD\_SUCCESS\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1256  
**CFE\_TBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1256  
**CFE\_TBL\_LOAD\_TYPE\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1256  
**CFE\_TBL\_LOAD\_VAL\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1256  
**CFE\_TBL\_LoadCmd**, 623  
 CommandHeader, 624  
 Payload, 624  
**CFE\_TBL\_LoadCmd\_Payload**, 624  
 LoadFilename, 624  
**CFE\_TBL\_LoadCmd\_Payload\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1243  
**CFE\_TBL\_LoadCmd\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1243  
**CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1257  
**CFE\_TBL\_LOADING\_PENDING\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1257  
**CFE\_TBL\_Manage**  
 cFE Manage Table Content APIs, 364  
**CFE\_TBL\_MAX\_FULL\_NAME\_LEN**  
 cfe\_tbl\_api\_typedefs.h, 1024  
**CFE\_TBL\_MESSAGE\_ERROR**  
 cFE Return Code Defines, 238  
**CFE\_TBL\_MID\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1257  
**CFE\_TBL\_Modified**  
 cFE Manage Table Content APIs, 365  
**CFE\_TBL\_NO\_INACTIVE\_BUFFER\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1257  
**CFE\_TBL\_NO SUCH\_TABLE\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1258  
**CFE\_TBL\_NO\_WORK\_BUFFERS\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1258  
**CFE\_TBL\_NoArgsCmd**, 625  
 CommandHeader, 625  
**CFE\_TBL\_NoArgsCmd\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1243  
**CFE\_TBL\_NOOP\_CC**  
 default\_cfe\_tbl\_fcncodes.h, 1227  
**CFE\_TBL\_NOOP\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1258  
**CFE\_TBL\_NoopCmd\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1243  
**CFE\_TBL\_NOT\_CRITICAL\_TBL\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1258  
**CFE\_TBL\_NOT\_IMPLEMENTED**  
 cFE Return Code Defines, 238  
**CFE\_TBL\_NOT\_IN\_CRIT\_REG\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1259  
**CFE\_TBL\_NotifyByMessage**  
 cFE Get Table Information APIs, 374  
**CFE\_TBL\_NotifyCmd**, 625  
 CommandHeader, 625  
 Payload, 626  
**CFE\_TBL\_NotifyCmd\_Payload**, 626  
 Parameter, 626  
**CFE\_TBL\_NotifyCmd\_Payload\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1243  
**CFE\_TBL\_NotifyCmd\_t**  
 default\_cfe\_tbl\_msgstruct.h, 1243  
**CFE\_TBL\_OPT\_BUFFER\_MSK**  
 cFE Table Type Defines, 376  
**CFE\_TBL\_OPT\_CRITICAL**  
 cFE Table Type Defines, 376  
**CFE\_TBL\_OPT\_CRITICAL\_MSK**  
 cFE Table Type Defines, 376  
**CFE\_TBL\_OPT\_DBL\_BUFFER**  
 cFE Table Type Defines, 376  
**CFE\_TBL\_OPT\_DEFAULT**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_DUMP\_ONLY**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_LD\_DMP\_MSK**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_LOAD\_DUMP**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_NOT\_CRITICAL**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_NOT\_USR\_DEF**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_SNGL\_BUFFER**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_USR\_DEF\_ADDR**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OPT\_USR\_DEF\_MSK**  
 cFE Table Type Defines, 377  
**CFE\_TBL\_OVERWRITE\_DUMP\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1259  
**CFE\_TBL\_OVERWRITE\_REG\_DUMP\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1259  
**CFE\_TBL\_PARTIAL\_LOAD\_ERR\_EID**

cfe\_tbl\_eventids.h, 1259  
CFE\_TBL\_PROCESSOR\_ID\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1260  
CFE\_TBL\_REG\_TLM\_MID  
    default\_cfe\_tbl\_msgids.h, 1239  
CFE\_TBL\_Register  
    cFE Registration APIs, 357  
CFE\_TBL\_REGISTER\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1260  
CFE\_TBL\_ReleaseAddress  
    cFE Access Table Content APIs, 370  
CFE\_TBL\_ReleaseAddresses  
    cFE Access Table Content APIs, 371  
CFE\_TBL\_RESET\_COUNTERS\_CC  
    default\_cfe\_tbl\_fcncodes.h, 1228  
CFE\_TBL\_RESET\_INF\_EID  
    cfe\_tbl\_eventids.h, 1260  
CFE\_TBL\_ResetCountersCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1243  
CFE\_TBL\_SEND\_HK\_MID  
    default\_cfe\_tbl\_msgids.h, 1239  
CFE\_TBL\_SEND\_REGISTRY\_CC  
    default\_cfe\_tbl\_fcncodes.h, 1229  
CFE\_TBL\_SendHkCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1243  
CFE\_TBL\_SendRegistryCmd, 626  
    CommandHeader, 627  
    Payload, 627  
CFE\_TBL\_SendRegistryCmd\_Payload, 627  
    TableName, 627  
CFE\_TBL\_SendRegistryCmd\_Payload\_t  
    default\_cfe\_tbl\_msgstruct.h, 1243  
CFE\_TBL\_SendRegistryCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1244  
CFE\_TBL\_Share  
    cFE Registration APIs, 359  
CFE\_TBL\_SHARE\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1260  
CFE\_TBL\_SPACECRAFT\_ID\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1261  
CFE\_TBL\_SRC\_ADDRESS  
    cfe\_tbl\_api\_typedefs.h, 1025  
CFE\_TBL\_SRC\_FILE  
    cfe\_tbl\_api\_typedefs.h, 1025  
CFE\_TBL\_SrcEnum  
    cfe\_tbl\_api\_typedefs.h, 1024  
CFE\_TBL\_SrcEnum\_t  
    cfe\_tbl\_api\_typedefs.h, 1024  
CFE\_TBL\_TableRegistryTlm, 627  
    Payload, 628  
    TelemetryHeader, 628  
CFE\_TBL\_TableRegistryTlm\_t  
    default\_cfe\_tbl\_msgstruct.h, 1244  
CFE\_TBL\_TblRegPacket\_Payload, 628  
    ActiveBufferAddr, 629  
    ByteAlign4, 629  
    Crc, 629  
    Critical, 629  
    DoubleBuffered, 630  
    DumpOnly, 630  
    FileCreateTimeSecs, 630  
    FileCreateTimeSubSecs, 630  
    InactiveBufferAddr, 630  
    LastFileLoaded, 630  
    LoadPending, 630  
    Name, 631  
    OwnerAppName, 631  
    Size, 631  
    TableLoadedOnce, 631  
    TimeOfLastUpdate, 631  
    ValidationFuncPtr, 631  
CFE\_TBL\_TblRegPacket\_Payload\_t  
    default\_cfe\_tbl\_msgstruct.h, 1244  
CFE\_TBL\_TLM\_REG\_CMD\_INF\_EID  
    cfe\_tbl\_eventids.h, 1261  
CFE\_TBL\_TOO\_MANY\_DUMPS\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1261  
CFE\_TBL\_TOO\_MANY\_VALIDATIONS\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1261  
CFE\_TBL\_Unregister  
    cFE Registration APIs, 360  
CFE\_TBL\_UNREGISTER\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1262  
CFE\_TBL\_UNVALIDATED\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1262  
CFE\_TBL\_Update  
    cFE Manage Table Content APIs, 366  
CFE\_TBL\_UPDATE\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1262  
CFE\_TBL\_UPDATE\_SUCCESS\_INF\_EID  
    cfe\_tbl\_eventids.h, 1262  
CFE\_TBL\_VAL\_REQ\_MADE\_INF\_EID  
    cfe\_tbl\_eventids.h, 1263  
CFE\_TBL\_Validate  
    cFE Manage Table Content APIs, 366  
CFE\_TBL\_VALIDATE\_CC  
    default\_cfe\_tbl\_fcncodes.h, 1230  
CFE\_TBL\_ValidateCmd, 632  
    CommandHeader, 632  
    Payload, 632  
CFE\_TBL\_ValidateCmd\_Payload, 632  
    ActiveTableFlag, 632  
    TableName, 633  
CFE\_TBL\_ValidateCmd\_Payload\_t  
    default\_cfe\_tbl\_msgstruct.h, 1244  
CFE\_TBL\_ValidateCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1244  
CFE\_TBL\_VALIDATION\_ERR\_EID

cfe\_tbl\_eventids.h, 1263  
**CFE\_TBL\_VALIDATION\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1263  
**CFE\_TBL\_WARN\_DUPLICATE**  
 cFE Return Code Defines, 238  
**CFE\_TBL\_WARN\_NOT\_CRITICAL**  
 cFE Return Code Defines, 238  
**CFE\_TBL\_WARN\_PARTIAL\_LOAD**  
 cFE Return Code Defines, 238  
**CFE\_TBL\_WARN\_SHORT\_FILE**  
 cFE Return Code Defines, 239  
**CFE\_TBL\_WRITE\_CFE\_HDR\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1263  
**CFE\_TBL\_WRITE\_DUMP\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1264  
**CFE\_TBL\_WRITE\_REG\_DUMP\_INF\_EID**  
 cfe\_tbl\_eventids.h, 1264  
**CFE\_TBL\_WRITE\_TBL\_HDR\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1264  
**CFE\_TBL\_WRITE\_TBL\_IMG\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1264  
**CFE\_TBL\_WRITE\_TBL\_REG\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1265  
**CFE\_TBL\_ZERO\_LENGTH\_LOAD\_ERR\_EID**  
 cfe\_tbl\_eventids.h, 1265  
**cfe\_time.h**  
 CFE\_TIME\_Copy, 1028  
**CFE\_TIME\_1HZ\_CFG\_EID**  
 cfe\_time\_eventids.h, 1306  
**CFE\_TIME\_1HZ\_CMD\_MID**  
 default\_cfe\_time\_msgids.h, 1297  
**CFE\_TIME\_1HZ\_EID**  
 cfe\_time\_eventids.h, 1306  
**CFE\_TIME\_1HzCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1300  
**CFE\_TIME\_A\_GT\_B**  
 cfe\_time\_api\_typedefs.h, 1029  
**CFE\_TIME\_A\_LT\_B**  
 cfe\_time\_api\_typedefs.h, 1029  
**CFE\_TIME\_Add**  
 cFE Time Arithmetic APIs, 384  
**CFE\_TIME\_Add1HZAdjustmentCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1300  
**CFE\_TIME\_ADD\_1HZ\_ADJUSTMENT\_CC**  
 default\_cfe\_time\_fnccodes.h, 1271  
**CFE\_TIME\_ADD\_ADJUST\_CC**  
 default\_cfe\_time\_fnccodes.h, 1272  
**CFE\_TIME\_ADD\_DELAY\_CC**  
 default\_cfe\_time\_fnccodes.h, 1272  
**CFE\_TIME\_AddAdjustCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1300  
**CFE\_TIME\_AddDelayCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1300  
**CFE\_TIME\_AdjustDirection**  
 default\_cfe\_time\_extern\_typedefs.h, 1268  
**CFE\_TIME\_AdjustDirection\_ADD**  
 default\_cfe\_time\_extern\_typedefs.h, 1268  
**CFE\_TIME\_AdjustDirection\_Enum\_t**  
 default\_cfe\_time\_extern\_typedefs.h, 1267  
**CFE\_TIME\_AdjustDirection\_SUBTRACT**  
 default\_cfe\_time\_extern\_typedefs.h, 1268  
**cfe\_time\_api\_typedefs.h**  
 CFE\_TIME\_A\_GT\_B, 1029  
 CFE\_TIME\_A\_LT\_B, 1029  
 CFE\_TIME\_Compare, 1029  
 CFE\_TIME\_Compare\_t, 1029  
 CFE\_TIME\_EQUAL, 1029  
 CFE\_TIME\_PRINTED\_STRING\_SIZE, 1028  
 CFE\_TIME\_SynchCallbackPtr\_t, 1029  
**CFE\_TIME\_BAD\_ARGUMENT**  
 cFE Return Code Defines, 239  
**CFE\_TIME\_CALLBACK\_NOT\_REGISTERED**  
 cFE Return Code Defines, 239  
**CFE\_TIME\_CC\_ERR\_EID**  
 cfe\_time\_eventids.h, 1307  
**CFE\_TIME\_ClockState**  
 default\_cfe\_time\_extern\_typedefs.h, 1268  
**CFE\_TIME\_ClockState\_Enum\_t**  
 default\_cfe\_time\_extern\_typedefs.h, 1267  
**CFE\_TIME\_ClockState\_FLYWHEEL**  
 default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_ClockState\_INVALID**  
 default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_ClockState\_VALID**  
 default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_CMD\_MID**  
 default\_cfe\_time\_msgids.h, 1297  
**CFE\_TIME\_Compare**  
 cFE Time Arithmetic APIs, 384  
 cfe\_time\_api\_typedefs.h, 1029  
**CFE\_TIME\_Compare\_t**  
 cfe\_time\_api\_typedefs.h, 1029  
**CFE\_TIME\_Copy**  
 cfe\_time.h, 1028  
**CFE\_TIME\_DATA\_CMD\_MID**  
 default\_cfe\_time\_msgids.h, 1297  
**CFE\_TIME\_DELAY\_CFG\_EID**  
 cfe\_time\_eventids.h, 1307  
**CFE\_TIME\_DELAY\_EID**  
 cfe\_time\_eventids.h, 1307  
**CFE\_TIME\_DELAY\_ERR\_EID**  
 cfe\_time\_eventids.h, 1307  
**CFE\_TIME\_DELTA\_CFG\_EID**  
 cfe\_time\_eventids.h, 1308  
**CFE\_TIME\_DELTA\_EID**  
 cfe\_time\_eventids.h, 1308  
**CFE\_TIME\_DELTA\_ERR\_EID**  
 cfe\_time\_eventids.h, 1308

CFE\_TIME\_DIAG\_EID  
  cfe\_time\_eventids.h, 1308

CFE\_TIME\_DIAG\_TLM\_MID  
  default\_cfe\_time\_msgids.h, 1297

CFE\_TIME\_DiagnosticTlm, 633  
  Payload, 633  
  TelemetryHeader, 633

CFE\_TIME\_DiagnosticTlm\_Payload, 633  
  AtToneDelay, 635  
  AtToneLatch, 636  
  AtToneLeapSeconds, 636  
  AtToneMET, 636  
  AtToneSTCF, 636  
  ClockFlyState, 636  
  ClockSetState, 636  
  ClockSignal, 636  
  ClockSource, 637  
  ClockStateAPI, 637  
  ClockStateFlags, 637  
  CurrentLatch, 637  
  CurrentMET, 637  
  CurrentTAI, 637  
  CurrentUTC, 637  
  DataStoreStatus, 638  
  DelayDirection, 638  
  Forced2Fly, 638  
  LocalIntCounter, 638  
  LocalTaskCounter, 638  
  MaxElapsed, 638  
  MaxLocalClock, 638  
  MinElapsed, 639  
  OneHzAdjust, 639  
  OneHzDirection, 639  
  OneTimeAdjust, 639  
  OneTimeDirection, 639  
  ServerFlyState, 639  
  TimeSinceTone, 639  
  ToneDataCounter, 640  
  ToneDataLatch, 640  
  ToneIntCounter, 640  
  ToneIntErrorCounter, 640  
  ToneMatchCounter, 640  
  ToneMatchErrorCounter, 640  
  ToneOverLimit, 640  
  ToneSignalCounter, 641  
  ToneSignalLatch, 641  
  ToneTaskCounter, 641  
  ToneUnderLimit, 641  
  VersionCounter, 641  
  VirtualMET, 641

CFE\_TIME\_DiagnosticTlm\_Payload\_t  
  default\_cfe\_time\_msgstruct.h, 1300

CFE\_TIME\_DiagnosticTlm\_t  
  default\_cfe\_time\_msgstruct.h, 1300

CFE\_TIME\_EQUAL  
  cfe\_time\_api\_typedefs.h, 1029

cfe\_time\_eventids.h  
  CFE\_TIME\_1HZ\_CFG\_EID, 1306  
  CFE\_TIME\_1HZ\_EID, 1306  
  CFE\_TIME\_CC\_ERR\_EID, 1307  
  CFE\_TIME\_DELAY\_CFG\_EID, 1307  
  CFE\_TIME\_DELAY\_EID, 1307  
  CFE\_TIME\_DELAY\_ERR\_EID, 1307  
  CFE\_TIME\_DELTA\_CFG\_EID, 1308  
  CFE\_TIME\_DELTA\_EID, 1308  
  CFE\_TIME\_DELTA\_ERR\_EID, 1308  
  CFE\_TIME\_DIAG\_EID, 1308  
  CFE\_TIME\_FLY\_OFF\_EID, 1309  
  CFE\_TIME\_FLY\_ON\_EID, 1309  
  CFE\_TIME\_ID\_ERR\_EID, 1309  
  CFE\_TIME\_INIT\_EID, 1309  
  CFE\_TIME\_LEAPS\_CFG\_EID, 1310  
  CFE\_TIME\_LEAPS\_EID, 1310  
  CFE\_TIME\_LEN\_ERR\_EID, 1310  
  CFE\_TIME\_MET\_CFG\_EID, 1310  
  CFE\_TIME\_MET\_EID, 1311  
  CFE\_TIME\_MET\_ERR\_EID, 1311  
  CFE\_TIME\_NOOP\_EID, 1311  
  CFE\_TIME\_RESET\_EID, 1311  
  CFE\_TIME\_SIGNAL\_CFG\_EID, 1312  
  CFE\_TIME\_SIGNAL\_EID, 1312  
  CFE\_TIME\_SIGNAL\_ERR\_EID, 1312  
  CFE\_TIME\_SOURCE\_CFG\_EID, 1312  
  CFE\_TIME\_SOURCE\_EID, 1313  
  CFE\_TIME\_SOURCE\_ERR\_EID, 1313  
  CFE\_TIME\_STATE\_EID, 1313  
  CFE\_TIME\_STATE\_ERR\_EID, 1313  
  CFE\_TIME\_STCF\_CFG\_EID, 1314  
  CFE\_TIME\_STCF\_EID, 1314  
  CFE\_TIME\_STCF\_ERR\_EID, 1314  
  CFE\_TIME\_TIME\_CFG\_EID, 1314  
  CFE\_TIME\_TIME\_EID, 1315  
  CFE\_TIME\_TIME\_ERR\_EID, 1315

CFE\_TIME\_ExternalGPS  
  cFE External Time Source APIs, 389

CFE\_TIME\_ExternalMET  
  cFE External Time Source APIs, 390

CFE\_TIME\_ExternalTime  
  cFE External Time Source APIs, 390

CFE\_TIME\_ExternalTone  
  cFE External Time Source APIs, 391

CFE\_TIME\_FakeToneCmd\_t  
  default\_cfe\_time\_msgstruct.h, 1300

CFE\_TIME\_FLAG\_ADD1HZ  
  cFE Clock State Flag Defines, 399

CFE\_TIME\_FLAG\_ADDADJ  
  cFE Clock State Flag Defines, 399

CFE\_TIME\_FLAG\_ADDTCL

cFE Clock State Flag Defines, 399  
**CFE\_TIME\_FLAG\_CLKSET**  
  cFE Clock State Flag Defines, 399  
**CFE\_TIME\_FLAG\_CMDFLY**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_FLYING**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_GDTONE**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_REFERR**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_SERVER**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_SIGPRI**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_SRCINT**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_SRVFLY**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FLAG\_UNUSED**  
  cFE Clock State Flag Defines, 400  
**CFE\_TIME\_FlagBit**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_ADD1HZ**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_ADDADJ**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_ADDTCL**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_CLKSET**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_CMDFLY**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_Enum\_t**  
  default\_cfe\_time\_extern\_typedefs.h, 1267  
**CFE\_TIME\_FlagBit\_FLYING**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_GDTONE**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_SERVER**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_SIGPRI**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_SRCINT**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlagBit\_SRVFLY**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FLY\_OFF\_EID**  
  cfe\_time\_eventids.h, 1309  
**CFE\_TIME\_FLY\_ON\_EID**  
  cfe\_time\_eventids.h, 1309  
**CFE\_TIME\_FlywheelState**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlywheelState\_Enum\_t**  
  default\_cfe\_time\_extern\_typedefs.h, 1267  
**CFE\_TIME\_FlywheelState\_IS\_FLY**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_FlywheelState\_NO\_FLY**  
  default\_cfe\_time\_extern\_typedefs.h, 1269  
**CFE\_TIME\_GetClockInfo**  
  cFE Get Time Information APIs, 381  
**CFE\_TIME\_GetClockState**  
  cFE Get Time Information APIs, 381  
**CFE\_TIME\_GetLeapSeconds**  
  cFE Get Time Information APIs, 382  
**CFE\_TIME\_GetMET**  
  cFE Get Current Time APIs, 378  
**CFE\_TIME\_GetMETseconds**  
  cFE Get Current Time APIs, 378  
**CFE\_TIME\_GetMETsubsecs**  
  cFE Get Current Time APIs, 379  
**CFE\_TIME\_GetSTCF**  
  cFE Get Time Information APIs, 382  
**CFE\_TIME\_GetTAI**  
  cFE Get Current Time APIs, 379  
**CFE\_TIME\_GetTime**  
  cFE Get Current Time APIs, 380  
**CFE\_TIME\_GetUTC**  
  cFE Get Current Time APIs, 380  
**CFE\_TIME\_HK\_TLM\_MID**  
  default\_cfe\_time\_msgids.h, 1297  
**CFE\_TIME\_HousekeepingTlm**, 642  
  Payload, 642  
  TelemetryHeader, 642  
**CFE\_TIME\_HousekeepingTlm\_Payload**, 642  
  ClockStateAPI, 643  
  ClockStateFlags, 643  
  CommandCounter, 643  
  CommandErrorCounter, 643  
  LeapSeconds, 643  
  Seconds1HzAdj, 644  
  SecondsDelay, 644  
  SecondsMET, 644  
  SecondsSTCF, 644  
  Subsecs1HzAdj, 644  
  SubsecsDelay, 644  
  SubsecsMET, 644  
  SubsecsSTCF, 645  
**CFE\_TIME\_HousekeepingTlm\_Payload\_t**  
  default\_cfe\_time\_msgstruct.h, 1301  
**CFE\_TIME\_HousekeepingTlm\_t**  
  default\_cfe\_time\_msgstruct.h, 1301  
**CFE\_TIME\_ID\_ERR\_EID**  
  cfe\_time\_eventids.h, 1309  
**CFE\_TIME\_INIT\_EID**  
  cfe\_time\_eventids.h, 1309  
**CFE\_TIME\_INTERNAL\_ONLY**  
  cFE Return Code Defines, 239

CFE\_TIME\_LEAPS\_CFG\_EID  
    cfe\_time\_eventids.h, 1310

CFE\_TIME\_LEAPS\_EID  
    cfe\_time\_eventids.h, 1310

CFE\_TIME\_LeapsCmd\_Payload, 645  
    LeapSeconds, 645

CFE\_TIME\_LeapsCmd\_Payload\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_LEN\_ERR\_EID  
    cfe\_time\_eventids.h, 1310

CFE\_TIME\_Local1HzISR  
    cFE Miscellaneous Time APIs, 394

CFE\_TIME\_MET2SCTime  
    cFE Time Conversion APIs, 387

CFE\_TIME\_MET\_CFG\_EID  
    cfe\_time\_eventids.h, 1310

CFE\_TIME\_MET\_EID  
    cfe\_time\_eventids.h, 1311

CFE\_TIME\_MET\_ERR\_EID  
    cfe\_time\_eventids.h, 1311

CFE\_TIME\_Micro2SubSecs  
    cFE Time Conversion APIs, 387

CFE\_TIME\_NoArgsCmd, 645  
    CommandHeader, 646

CFE\_TIME\_NoArgsCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_NOOP\_CC  
    default\_cfe\_time\_fcncodes.h, 1273

CFE\_TIME\_NOOP\_EID  
    cfe\_time\_eventids.h, 1311

CFE\_TIME\_NoopCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_NOT\_IMPLEMENTED  
    cFE Return Code Defines, 239

CFE\_TIME\_OneHzAdjustmentCmd, 646  
    CommandHeader, 646  
    Payload, 646

CFE\_TIME\_OneHzAdjustmentCmd\_Payload, 646  
    Seconds, 647  
    Subseconds, 647

CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_OneHzAdjustmentCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_OUT\_OF\_RANGE  
    cFE Return Code Defines, 239

CFE\_TIME\_Print  
    cFE Miscellaneous Time APIs, 394

CFE\_TIME\_PRINTED\_STRING\_SIZE  
    cfe\_time\_api\_typedefs.h, 1028

CFE\_TIME\_RegisterSyncCallback  
    cFE External Time Source APIs, 391

CFE\_TIME\_RESET\_COUNTERS\_CC  
    default\_cfe\_time\_fcncodes.h, 1274

CFE\_TIME\_RESET\_EID  
    cfe\_time\_eventids.h, 1311

CFE\_TIME\_ResetCountersCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_SEND\_CMD\_MID  
    default\_cfe\_time\_msgids.h, 1298

CFE\_TIME\_SEND\_DIAGNOSTIC\_TLM\_CC  
    default\_cfe\_time\_fcncodes.h, 1275

CFE\_TIME\_SEND\_HK\_MID  
    default\_cfe\_time\_msgids.h, 1298

CFE\_TIME\_SendDiagnosticCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_SendHkCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_SERVICE  
    cfe\_error.h, 987

CFE\_TIME\_SET\_LEAP\_SECONDS\_CC  
    default\_cfe\_time\_fcncodes.h, 1276

CFE\_TIME\_SET\_MET\_CC  
    default\_cfe\_time\_fcncodes.h, 1277

CFE\_TIME\_SET\_SIGNAL\_CC  
    default\_cfe\_time\_fcncodes.h, 1278

CFE\_TIME\_SET\_SOURCE\_CC  
    default\_cfe\_time\_fcncodes.h, 1278

CFE\_TIME\_SET\_STATE\_CC  
    default\_cfe\_time\_fcncodes.h, 1279

CFE\_TIME\_SET\_STCF\_CC  
    default\_cfe\_time\_fcncodes.h, 1281

CFE\_TIME\_SET\_TIME\_CC  
    default\_cfe\_time\_fcncodes.h, 1281

CFE\_TIME\_SetLeapSecondsCmd, 647  
    CommandHeader, 647  
    Payload, 647

CFE\_TIME\_SetLeapSecondsCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_SetMETCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1301

CFE\_TIME\_SetSignalCmd, 648  
    CommandHeader, 648  
    Payload, 648

CFE\_TIME\_SetSignalCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1302

CFE\_TIME\_SetSourceCmd, 648  
    CommandHeader, 648  
    Payload, 649

CFE\_TIME\_SetSourceCmd\_t  
    default\_cfe\_time\_msgstruct.h, 1302

CFE\_TIME\_SetState  
    default\_cfe\_time\_extern\_typedefs.h, 1269

CFE\_TIME\_SetState\_Enum\_t  
    default\_cfe\_time\_extern\_typedefs.h, 1267

CFE\_TIME\_SetState\_NOT\_SET  
    default\_cfe\_time\_extern\_typedefs.h, 1270

CFE\_TIME\_SetState\_WAS\_SET

**default\_cfe\_time\_extern\_typedefs.h**, [1270](#)  
**CFE\_TIME\_SetStateCmd**, [649](#)

- CommandHeader, [649](#)
- Payload, [649](#)

**CFE\_TIME\_SetStateCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_SetSTCFCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_SetTimeCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_SIGNAL\_CFG\_EID**

- cfe\_time\_eventids.h, [1312](#)

**CFE\_TIME\_SIGNAL\_EID**

- cfe\_time\_eventids.h, [1312](#)

**CFE\_TIME\_SIGNAL\_ERR\_EID**

- cfe\_time\_eventids.h, [1312](#)

**CFE\_TIME\_SignalCmd\_Payload**, [649](#)

- ToneSource, [650](#)

**CFE\_TIME\_SignalCmd\_Payload\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_SOURCE\_CFG\_EID**

- cfe\_time\_eventids.h, [1312](#)

**CFE\_TIME\_SOURCE\_EID**

- cfe\_time\_eventids.h, [1313](#)

**CFE\_TIME\_SOURCE\_ERR\_EID**

- cfe\_time\_eventids.h, [1313](#)

**CFE\_TIME\_SourceCmd\_Payload**, [650](#)

- TimeSource, [650](#)

**CFE\_TIME\_SourceCmd\_Payload\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_SourceSelect**

- default\_cfe\_time\_extern\_typedefs.h, [1270](#)

**CFE\_TIME\_SourceSelect\_Enum\_t**

- default\_cfe\_time\_extern\_typedefs.h, [1268](#)

**CFE\_TIME\_SourceSelect\_EXTERNAL**

- default\_cfe\_time\_extern\_typedefs.h, [1270](#)

**CFE\_TIME\_SourceSelect\_INTERNAL**

- default\_cfe\_time\_extern\_typedefs.h, [1270](#)

**CFE\_TIME\_STATE\_EID**

- cfe\_time\_eventids.h, [1313](#)

**CFE\_TIME\_STATE\_ERR\_EID**

- cfe\_time\_eventids.h, [1313](#)

**CFE\_TIME\_StateCmd\_Payload**, [651](#)

- ClockState, [651](#)

**CFE\_TIME\_StateCmd\_Payload\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_STCF\_CFG\_EID**

- cfe\_time\_eventids.h, [1314](#)

**CFE\_TIME\_STCF\_EID**

- cfe\_time\_eventids.h, [1314](#)

**CFE\_TIME\_STCF\_ERR\_EID**

- cfe\_time\_eventids.h, [1314](#)

**CFE\_TIME\_Sub1HZAdjustmentCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_Sub2MicroSecs**

- cFE Time Conversion APIs, [388](#)

**CFE\_TIME\_SUB\_1HZ\_ADJUSTMENT\_CC**

- default\_cfe\_time\_fnccodes.h, [1282](#)

**CFE\_TIME\_SUB\_ADJUST\_CC**

- default\_cfe\_time\_fnccodes.h, [1283](#)

**CFE\_TIME\_SUB\_DELAY\_CC**

- default\_cfe\_time\_fnccodes.h, [1284](#)

**CFE\_TIME\_SubAdjustCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_SubDelayCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_Subtract**

- cFE Time Arithmetic APIs, [385](#)

**CFE\_TIME\_SynchCallbackPtr\_t**

- cfe\_time\_api\_typedefs.h, [1029](#)

**CFE\_TIME\_SysTime**, [651](#)

- Seconds, [652](#)
- Subseconds, [652](#)

**CFE\_TIME\_SysTime\_t**

- default\_cfe\_time\_extern\_typedefs.h, [1268](#)

**CFE\_TIME\_TIME\_CFG\_EID**

- cfe\_time\_eventids.h, [1314](#)

**CFE\_TIME\_TIME\_EID**

- cfe\_time\_eventids.h, [1315](#)

**CFE\_TIME\_TIME\_ERR\_EID**

- cfe\_time\_eventids.h, [1315](#)

**CFE\_TIME\_TimeCmd**, [652](#)

- CommandHeader, [652](#)
- Payload, [652](#)

**CFE\_TIME\_TimeCmd\_Payload**, [652](#)

- MicroSeconds, [653](#)
- Seconds, [653](#)

**CFE\_TIME\_TimeCmd\_Payload\_t**

- default\_cfe\_time\_msgstruct.h, [1302](#)

**CFE\_TIME\_TONE\_CMD\_MID**

- default\_cfe\_time\_msgids.h, [1298](#)

**CFE\_TIME\_ToneDataCmd**, [653](#)

- CommandHeader, [653](#)
- Payload, [653](#)

**CFE\_TIME\_ToneDataCmd\_Payload**, [654](#)

- AtToneLeapSeconds, [654](#)
- AtToneMET, [654](#)
- AtToneState, [654](#)
- AtToneSTCF, [654](#)

**CFE\_TIME\_ToneDataCmd\_Payload\_t**

- default\_cfe\_time\_msgstruct.h, [1303](#)

**CFE\_TIME\_ToneDataCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1303](#)

**CFE\_TIME\_ToneSignalCmd\_t**

- default\_cfe\_time\_msgstruct.h, [1303](#)

**CFE\_TIME\_ToneSignalSelect**

default\_cfe\_time\_extern\_typedefs.h, 1270  
CFE\_TIME\_ToneSignalSelect\_Enum\_t  
    default\_cfe\_time\_extern\_typedefs.h, 1268  
CFE\_TIME\_ToneSignalSelect\_PRIMARY  
    default\_cfe\_time\_extern\_typedefs.h, 1270  
CFE\_TIME\_ToneSignalSelect\_REDUNDANT  
    default\_cfe\_time\_extern\_typedefs.h, 1270  
CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS  
    cFE Return Code Defines, 239  
CFE\_TIME\_UnregisterSynchCallback  
    cFE External Time Source APIs, 392  
CFE\_TST  
    cfe\_sb.h, 1018  
cfe\_version.h  
    CFE\_BUILD\_BASELINE, 1030  
    CFE\_BUILD\_NUMBER, 1030  
    CFE\_MAJOR\_VERSION, 1030  
    CFE\_MINOR\_VERSION, 1030  
    CFE\_MISSION\_REV, 1030  
    CFE\_REVISION, 1031  
    CFE\_SRC\_VERSION, 1031  
    CFE\_STR, 1031  
    CFE\_STR\_HELPER, 1031  
    CFE\_VERSION\_STRING, 1031  
CFE\_VERSION\_STRING  
    cfe\_version.h, 1031  
CfeCoreBaseline  
    CS\_HkPacket\_Payload\_t, 669  
CFECoreChecksum  
    CFE\_ES\_HousekeepingTlm\_Payload, 540  
CfeCoreCodeSeg  
    CS\_AppData\_t, 656  
CfeCoreCSErrCounter  
    CS\_HkPacket\_Payload\_t, 669  
CfeCoreCSState  
    CS\_HkPacket\_Payload\_t, 670  
CFEMajorVersion  
    CFE\_ES\_HousekeepingTlm\_Payload, 540  
CFEMinorVersion  
    CFE\_ES\_HousekeepingTlm\_Payload, 540  
CFEMissionRevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 540  
CFERevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 540  
CFS Checksum Command Codes, 175  
    CS\_CANCEL\_ONE\_SHOT\_CC, 176  
    CS\_DISABLE\_ALL\_CS\_CC, 177  
    CS\_DISABLE\_APPS\_CC, 178  
    CS\_DISABLE\_CFE\_CORE\_CC, 179  
    CS\_DISABLE\_EEPROM\_CC, 179  
    CS\_DISABLE\_ENTRY\_EEPROM\_CC, 180  
    CS\_DISABLE\_ENTRY\_MEMORY\_CC, 181  
    CS\_DISABLE\_MEMORY\_CC, 182  
    CS\_DISABLE\_NAME\_APP\_CC, 182  
    CS\_DISABLE\_NAME\_TABLE\_CC, 183  
    CS\_DISABLE\_OS\_CC, 184  
    CS\_DISABLE\_TABLES\_CC, 185  
    CS\_ENABLE\_ALL\_CS\_CC, 185  
    CS\_ENABLE\_APPS\_CC, 186  
    CS\_ENABLE\_CFE\_CORE\_CC, 187  
    CS\_ENABLE\_EEPROM\_CC, 188  
    CS\_ENABLE\_ENTRY\_EEPROM\_CC, 188  
    CS\_ENABLE\_ENTRY\_MEMORY\_CC, 189  
    CS\_ENABLE\_MEMORY\_CC, 190  
    CS\_ENABLE\_NAME\_APP\_CC, 191  
    CS\_ENABLE\_NAME\_TABLE\_CC, 191  
    CS\_ENABLE\_OS\_CC, 192  
    CS\_ENABLE\_TABLES\_CC, 193  
    CS\_GET\_ENTRY\_ID\_EEPROM\_CC, 194  
    CS\_GET\_ENTRY\_ID\_MEMORY\_CC, 194  
    CS\_NOOP\_CC, 195  
    CS\_ONE\_SHOT\_CC, 196  
    CS\_RECOMPUTE\_BASELINE\_APP\_CC, 197  
    CS\_RECOMPUTE\_BASELINE\_CFE\_CORE\_CC,  
        198  
    CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC, 199  
    CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC, 200  
    CS\_RECOMPUTE\_BASELINE\_OS\_CC, 201  
    CS\_RECOMPUTE\_BASELINE\_TABLE\_CC, 201  
    CS\_REPORT\_BASELINE\_APP\_CC, 202  
    CS\_REPORT\_BASELINE\_CFE\_CORE\_CC, 203  
    CS\_REPORT\_BASELINE\_EEPROM\_CC, 204  
    CS\_REPORT\_BASELINE\_MEMORY\_CC, 205  
    CS\_REPORT\_BASELINE\_OS\_CC, 205  
    CS\_REPORT\_BASELINE\_TABLE\_CC, 206  
    CS\_RESET\_CC, 207  
CFS Checksum Command Message IDs, 208  
    CS\_BACKGROUND\_CYCLE\_MID, 208  
    CS\_CMD\_MID, 208  
    CS\_SEND\_HK\_MID, 208  
CFS Checksum Command Structures, 174  
CFS Checksum Event IDs, 125  
    CS\_APP\_MISCOMPARE\_ERR\_EID, 132  
    CS\_BASELINE\_APP\_INF\_EID, 132  
    CS\_BASELINE\_CFECORE\_INF\_EID, 132  
    CS\_BASELINE\_EEPROM\_INF\_EID, 133  
    CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID,  
        133  
    CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID,  
        133  
    CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID,  
        133  
    CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID,  
        134  
    CS\_BASELINE\_MEMORY\_INF\_EID, 134  
    CS\_BASELINE\_OS\_INF\_EID, 134  
    CS\_BASELINE\_TABLES\_INF\_EID, 134  
    CS\_BKGND\_COMPUTE\_PROG\_INF\_EID, 135

CS\_CC1\_ERR\_EID, 135  
 CS\_CFE\_TEXT\_SEG\_INF\_EID, 135  
 CS\_CFECORE\_MISCOMPARE\_ERR\_EID, 135  
 CS\_CMD\_COMPUTE\_PROG\_ERR\_EID, 136  
 CS\_COMPUTE\_APP\_ERR\_EID, 136  
 CS\_COMPUTE\_APP\_NOT\_FOUND\_ERR\_EID, 136  
 CS\_COMPUTE\_APP\_PLATFORM\_DBG\_EID, 136  
 CS\_COMPUTE\_TABLES\_ERR\_EID, 137  
 CS\_COMPUTE\_TABLES\_NOT\_FOUND\_ERR\_EID, 137  
 CS\_COMPUTE\_TABLES\_RELEASE\_ERR\_EID, 137  
 CS\_CR\_CDS\_CPY\_ERR\_EID, 137  
 CS\_CR\_CDS\_REG\_ERR\_EID, 138  
 CS\_CR\_CDS\_RES\_ERR\_EID, 138  
 CS\_DISABLE\_ALL\_INF\_EID, 138  
 CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID, 138  
 CS\_DISABLE\_APP\_INF\_EID, 139  
 CS\_DISABLE\_APP\_NAME\_INF\_EID, 139  
 CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID, 139  
 CS\_DISABLE\_CFECORE\_INF\_EID, 139  
 CS\_DISABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID, 140  
 CS\_DISABLE\_EEPROM\_ENTRY\_INF\_EID, 140  
 CS\_DISABLE\_EEPROM\_INF\_EID, 140  
 CS\_DISABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID, 140  
 CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID, 141  
 CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID, 141  
 CS\_DISABLE\_MEMORY\_INF\_EID, 141  
 CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID, 141  
 CS\_DISABLE\_OS\_INF\_EID, 142  
 CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID, 142  
 CS\_DISABLE\_TABLES\_INF\_EID, 142  
 CS\_DISABLE\_TABLES\_NAME\_INF\_EID, 142  
 CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID, 143  
 CS\_EEPROM\_MISCOMPARE\_ERR\_EID, 143  
 CS\_ENABLE\_ALL\_INF\_EID, 143  
 CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID, 143  
 CS\_ENABLE\_APP\_INF\_EID, 144  
 CS\_ENABLE\_APP\_NAME\_INF\_EID, 144  
 CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID, 144  
 CS\_ENABLE\_CFECORE\_INF\_EID, 144  
 CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID, 145  
 CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID, 145  
 CS\_ENABLE\_EEPROM\_INF\_EID, 145  
 CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID, 145  
 CS\_ENABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID, 146  
 CS\_ENABLE\_MEMORY\_ENTRY\_INF\_EID, 146  
 CS\_ENABLE\_MEMORY\_INF\_EID, 146  
 CS\_ENABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID, 146  
 CS\_ENABLE\_OS\_INF\_EID, 147  
 CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID, 147  
 CS\_ENABLE\_TABLES\_INF\_EID, 147  
 CS\_ENABLE\_TABLES\_NAME\_INF\_EID, 147  
 CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID, 148  
 CS\_EXIT\_ERR\_EID, 148  
 CS\_EXIT\_INF\_EID, 148  
 CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID, 148  
 CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID, 149  
 CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID, 149  
 CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID, 149  
 CS\_INIT\_APP\_ERR\_EID, 150  
 CS\_INIT\_EEPROM\_ERR\_EID, 150  
 CS\_INIT\_INF\_EID, 150  
 CS\_INIT\_MEMORY\_ERR\_EID, 150  
 CS\_INIT\_SB\_CREATE\_ERR\_EID, 151  
 CS\_INIT\_SB\_SUBSCRIBE\_BACK\_ERR\_EID, 151  
 CS\_INIT\_SB\_SUBSCRIBE\_CMD\_ERR\_EID, 151  
 CS\_INIT\_SB\_SUBSCRIBE\_HK\_ERR\_EID, 151  
 CS\_INIT\_TABLES\_ERR\_EID, 152  
 CS\_LEN\_ERR\_EID, 152  
 CS\_MEMORY\_MISCOMPARE\_ERR\_EID, 152  
 CS\_MID\_ERR\_EID, 152  
 CS\_NO\_BASELINE\_APP\_INF\_EID, 153  
 CS\_NO\_BASELINE\_CFECORE\_INF\_EID, 153  
 CS\_NO\_BASELINE\_EEPROM\_INF\_EID, 153  
 CS\_NO\_BASELINE\_MEMORY\_INF\_EID, 153  
 CS\_NO\_BASELINE\_OS\_INF\_EID, 154  
 CS\_NO\_BASELINE\_TABLES\_INF\_EID, 154  
 CS\_NOOP\_INF\_EID, 154  
 CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID, 154  
 CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID, 155  
 CS\_ONESHOT\_CANCELLED\_INF\_EID, 155  
 CS\_ONESHOT\_CHDTASK\_ERR\_EID, 155  
 CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID, 155  
 CS\_ONESHOT\_FINISHED\_INF\_EID, 156  
 CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID, 156  
 CS\_ONESHOT\_STARTED\_DBG\_EID, 156  
 CS\_OS\_MISCOMPARE\_ERR\_EID, 156

CS\_OS\_TEXT\_SEG\_INF\_EID, 157  
CS\_PROCESS\_APP\_NO\_ENTRIES\_INF\_EID, 157  
CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID, 157  
CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID, 157  
CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID, 158  
CS\_RECOMPUTE\_APP\_CREATE\_CHDTASK\_ERR\_EID, 158  
CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID, 158  
CS\_RECOMPUTE\_CFECORE\_CHDTASK\_ERR\_EID, 158  
CS\_RECOMPUTE\_CFECORE\_CREATE\_CHDTASK\_ERR\_EID, 159  
CS\_RECOMPUTE\_CFECORE\_STARTED\_DBG\_EID, 159  
CS\_RECOMPUTE\_EEPROM\_CHDTASK\_ERR\_EID, 159  
CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID, 160  
CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID, 160  
CS\_RECOMPUTE\_ERROR\_APP\_ERR\_EID, 160  
CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID, 160  
CS\_RECOMPUTE\_FINISH\_APP\_INF\_EID, 161  
CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID, 161  
CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID, 161  
CS\_RECOMPUTE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID, 161  
CS\_RECOMPUTE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID, 162  
CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR\_EID, 162  
CS\_RECOMPUTE\_MEMORY\_CREATE\_CHDTASK\_ERR\_EID, 162  
CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID, 163  
CS\_RECOMPUTE\_OS\_CHDTASK\_ERR\_EID, 163  
CS\_RECOMPUTE\_OS\_CREATE\_CHDTASK\_ERR\_EID, 163  
CS\_RECOMPUTE\_OS\_STARTED\_DBG\_EID, 163  
CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID, 164  
CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID, 164  
CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID, 164  
CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID, 165  
CS\_RESET\_DBG\_EID, 165  
CS\_TABLES\_MISCOMPARE\_ERR\_EID, 165  
CS\_TBL\_INIT\_ERR\_EID, 166  
~~CS\_TBL\_UPDATE\_ERR\_EID, 166~~  
CS\_UPDATE\_APP\_ERR\_EID, 166  
CS\_UPDATE\_CDS\_ERR\_EID, 166  
CS\_UPDATE EEPROM\_ERR\_EID, 167  
CS\_UPDATE\_MEMORY\_ERR\_EID, 167  
CS\_UPDATE\_TABLES\_ERR\_EID, 167  
CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID, 167  
CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID, 168  
CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID, 168  
CS\_VAL\_APP\_INF\_EID, 168  
CS\_VAL\_APP\_STATE\_ERR\_EID, 169  
CS\_VAL\_EEPROM\_INF\_EID, 169  
CS\_VAL\_EEPROM\_RANGE\_ERR\_EID, 169  
CS\_VAL\_EEPROM\_STATE\_ERR\_EID, 169  
~~CS\_VAL\_MEMORY\_INF\_EID, 170~~  
CS\_VAL\_MEMORY\_RANGE\_ERR\_EID, 170  
CS\_VAL\_MEMORY\_STATE\_ERR\_EID, 170  
CS\_VAL\_TABLES\_DEF\_TBL\_DUPL\_ERR\_EID, 170  
CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID, 171  
CS\_VAL\_TABLES\_INF\_EID, 171  
CS\_VAL\_TABLES\_STATE\_ERR\_EID, 171  
Checksum Mission Configuration, 172  
CS\_APPMAIN\_PERF\_ID, 172  
CS\_DEFAULT\_ALGORITHM, 172  
Checksum Platform Configuration, 210  
CS\_APPS\_TBL\_POWERON\_STATE, 211  
CS\_CDS\_NAME, 211  
CS\_CFECORE\_CHECKSUM\_STATE, 211  
CS\_CHILD\_TASK\_DELAY, 211  
CS\_CHILD\_TASK\_PRIORITY, 211  
CS\_DEF\_APP\_TABLE\_FILENAME, 212  
CS\_DEF\_EEPROM\_TABLE\_FILENAME, 212  
CS\_DEF\_MEMORY\_TABLE\_FILENAME, 212  
CS\_DEF\_TABLES\_TABLE\_FILENAME, 212  
CS\_DEFAULT\_BYTES\_PER\_CYCLE, 213  
CS\_EEPROM\_TBL\_POWERON\_STATE, 213  
CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES, 213  
CS\_MAX\_NUM\_EEPROM\_TABLE\_ENTRIES, 213  
CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES, 214  
CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES, 214  
CS\_MEMORY\_TBL\_POWERON\_STATE, 214  
CS\_MISSION\_REV, 214  
CS\_OSCS\_CHECKSUM\_STATE, 215  
CS\_PIPE\_DEPTH, 215  
CS\_PRESERVE\_STATES\_ON\_PROCESSOR\_RESET, 215  
CS\_STARTUP\_TIMEOUT, 215  
CS\_TABLES\_TBL\_POWERON\_STATE, 216  
CFS Checksum Telemetry, 173

CFS Checksum Telemetry Message IDs, [209](#)  
   CS\_HK\_TLM\_MID, [209](#)

CFS Checksum Version, [217](#)  
   CS\_MAJOR\_VERSION, [217](#)  
   CS\_MINOR\_VERSION, [217](#)  
   CS\_REVISION, [217](#)

CheckErrCtr  
   CFE\_ES\_MemPoolStats, [546](#)

ChecksumState  
   CS\_HkPacket\_Payload\_t, [670](#)

ChildTaskEntryID  
   CS\_AppData\_t, [657](#)

ChildTaskID  
   CS\_AppData\_t, [657](#)

ChildTaskTable  
   CS\_AppData\_t, [657](#)

ClockFlyState  
   CFE\_TIME\_DiagnosticTlm\_Payload, [636](#)

ClockSetState  
   CFE\_TIME\_DiagnosticTlm\_Payload, [636](#)

ClockSignal  
   CFE\_TIME\_DiagnosticTlm\_Payload, [636](#)

ClockSource  
   CFE\_TIME\_DiagnosticTlm\_Payload, [637](#)

ClockState  
   CFE\_TIME\_StateCmd\_Payload, [651](#)

ClockStateAPI  
   CFE\_TIME\_DiagnosticTlm\_Payload, [637](#)  
   CFE\_TIME\_HousekeepingTlm\_Payload, [643](#)

ClockStateFlags  
   CFE\_TIME\_DiagnosticTlm\_Payload, [637](#)  
   CFE\_TIME\_HousekeepingTlm\_Payload, [643](#)

CmdCounter  
   CS\_HkPacket\_Payload\_t, [670](#)

CmdErrCounter  
   CS\_HkPacket\_Payload\_t, [670](#)

CmdHeader  
   CS\_AppNameCmd\_t, [663](#)  
   CS\_EntryCmd\_t, [666](#)  
   CS\_GetEntryIDCmd\_t, [667](#)  
   CS\_NoArgsCmd\_t, [674](#)  
   CS\_OneShotCmd\_t, [676](#)  
   CS\_TableNameCmd\_t, [683](#)

CmdPipe  
   CS\_AppData\_t, [657](#)

code\_address  
   OS\_module\_address\_t, [689](#)

code\_size  
   OS\_module\_address\_t, [689](#)

CodeAddress  
   CFE\_ES\_AppInfo, [527](#)

CodeSize  
   CFE\_ES\_AppInfo, [528](#)

CommandCounter

CFE\_ES\_HousekeepingTlm\_Payload, [540](#)  
   CFE\_EVS\_HousekeepingTlm\_Payload, [574](#)  
   CFE\_SB\_HousekeepingTlm\_Payload, [590](#)  
   CFE\_TBL\_HousekeepingTlm\_Payload, [619](#)  
   CFE\_TIME\_HousekeepingTlm\_Payload, [643](#)

CommandErrorCounter  
   CFE\_ES\_HousekeepingTlm\_Payload, [540](#)  
   CFE\_EVS\_HousekeepingTlm\_Payload, [574](#)  
   CFE\_SB\_HousekeepingTlm\_Payload, [590](#)  
   CFE\_TBL\_HousekeepingTlm\_Payload, [619](#)  
   CFE\_TIME\_HousekeepingTlm\_Payload, [643](#)

CommandHeader  
   CFE\_ES\_AppNameCmd, [530](#)  
   CFE\_ES\_DeleteCDSCmd, [534](#)  
   CFE\_ES\_DumpCDSRegistryCmd, [535](#)  
   CFE\_ES\_FileNameCmd, [536](#)  
   CFE\_ES\_NoArgsCmd, [548](#)  
   CFE\_ES\_OverWriteSysLogCmd, [549](#)  
   CFE\_ES\_ReloadAppCmd, [552](#)  
   CFE\_ES\_RestartCmd, [553](#)  
   CFE\_ES\_SendMemPoolStatsCmd, [554](#)  
   CFE\_ES\_SetMaxPRCountCmd, [555](#)  
   CFE\_ES\_SetPerfFilterMaskCmd, [556](#)  
   CFE\_ES\_SetPerfTriggerMaskCmd, [557](#)  
   CFE\_ES\_StartApp, [558](#)  
   CFE\_ES\_StartPerfDataCmd, [561](#)  
   CFE\_ES\_StopPerfDataCmd, [562](#)  
   CFE\_EVS\_AppNameBitMaskCmd, [564](#)  
   CFE\_EVS\_AppNameCmd, [566](#)  
   CFE\_EVS\_AppNameEventIDCmd, [567](#)  
   CFE\_EVS\_AppNameEventIDMaskCmd, [568](#)  
   CFE\_EVS\_BitMaskCmd, [571](#)  
   CFE\_EVS\_NoArgsCmd, [578](#)  
   CFE\_EVS\_SetEventFormatModeCmd, [581](#)  
   CFE\_EVS\_SetLogModeCmd, [582](#)  
   CFE\_EVS\_WriteAppDataFileCmd, [584](#)  
   CFE\_EVS\_WriteLogFileCmd, [584](#)  
   CFE\_SB\_RouteCmd, [599](#)  
   CFE\_SB\_WriteFileInfoCmd, [608](#)  
   CFE\_TBL\_AbortLoadCmd, [609](#)  
   CFE\_TBL\_ActivateCmd, [610](#)  
   CFE\_TBL\_DeleteCDSCmd, [612](#)  
   CFE\_TBL\_DumpCmd, [612](#)  
   CFE\_TBL\_DumpRegistryCmd, [614](#)  
   CFE\_TBL\_LoadCmd, [624](#)  
   CFE\_TBL\_NoArgsCmd, [625](#)  
   CFE\_TBL\_NotifyCmd, [625](#)  
   CFE\_TBL\_SendRegistryCmd, [627](#)  
   CFE\_TBL\_ValidateCmd, [632](#)  
   CFE\_TIME\_NoArgsCmd, [646](#)  
   CFE\_TIME\_OneHzAdjustmentCmd, [646](#)  
   CFE\_TIME\_SetLeapSecondsCmd, [647](#)  
   CFE\_TIME\_SetSignalCmd, [648](#)  
   CFE\_TIME\_SetSourceCmd, [648](#)

CFE\_TIME\_SetStateCmd, 649  
CFE\_TIME\_TimeCmd, 652  
CFE\_TIME\_ToneDataCmd, 653  
common\_types.h  
  \_EXTENSION\_, 1317  
CompileTimeAssert, 1317, 1319, 1320  
cpuaddr, 1317  
cpudiff, 1318  
cpusize, 1318  
int16, 1318  
int32, 1318  
int64, 1318  
int8, 1318  
intptr, 1318  
OS\_ArgCallback\_t, 1318  
OS\_PRINTF, 1317  
OS\_USED, 1317  
OSAL\_BLOCKCOUNT\_C, 1317  
osal\_blockcount\_t, 1318  
osal\_id\_t, 1318  
OSAL\_INDEX\_C, 1317  
osal\_index\_t, 1318  
OSAL\_OBJTYPE\_C, 1317  
osal\_objtype\_t, 1319  
OSAL\_SIZE\_C, 1317  
OSAL\_STATUS\_C, 1317  
osal\_status\_t, 1319  
uint16, 1319  
uint32, 1319  
uint64, 1319  
uint8, 1319  
ComparisonValue  
  CS\_Res\_App\_Table\_Entry\_t, 677  
  CS\_Res\_EepromMemory\_Table\_Entry\_t, 678  
  CS\_Res\_Tables\_Table\_Entry\_t, 680  
CompileTimeAssert  
  common\_types.h, 1317, 1319, 1320  
ComputedYet  
  CS\_Res\_App\_Table\_Entry\_t, 677  
  CS\_Res\_EepromMemory\_Table\_Entry\_t, 678  
  CS\_Res\_Tables\_Table\_Entry\_t, 680  
ContentType  
  CFE\_FS\_Header, 586  
cpuaddr  
  common\_types.h, 1317  
cpudiff  
  common\_types.h, 1318  
cpusize  
  common\_types.h, 1318  
Crc  
  CFE\_TBL\_Info, 622  
  CFE\_TBL\_TblRegPacket\_Payload, 629  
CreatePipeErrorCounter  
  CFE\_SB\_HousekeepingTlm\_Payload, 591  
creator  
  OS\_bin\_sem\_prop\_t, 683  
  OS\_condvar\_prop\_t, 684  
  OS\_count\_sem\_prop\_t, 684  
  OS\_mut\_sem\_prop\_t, 691  
  OS\_queue\_prop\_t, 691  
  OS\_socket\_prop\_t, 693  
  OS\_task\_prop\_t, 695  
  OS\_timebase\_prop\_t, 697  
  OS\_timer\_prop\_t, 698  
Critical  
  CFE\_TBL\_Info, 622  
  CFE\_TBL\_TblRegPacket\_Payload, 629  
cs\_app.c  
  CS\_AppData, 735  
  CS\_AppInit, 725  
  CS\_AppMain, 726  
  CS\_AppPipe, 728  
  CS\_CreateRestoreStatesFromCDS, 730  
  CS\_HousekeepingCmd, 731  
  CS\_NUM\_DATA\_STORE\_STATES, 725  
  CS\_ProcessCmd, 732  
  CS\_UpdateCDS, 734  
cs\_app.h  
  CS\_AppData, 748  
  CS\_AppInit, 739  
  CS\_AppMain, 740  
  CS\_AppPipe, 741  
  CS\_CMD\_PIPE\_NAME, 737  
  CS\_CMD\_PIPE\_NAME\_LEN, 737  
  CS\_CreateRestoreStatesFromCDS, 743  
  CS\_ERR\_NOT\_FOUND, 737  
  CS\_ERROR, 737  
  CS\_HousekeepingCmd, 744  
  CS\_ONESHOT\_TASK\_NAME, 738  
  CS\_ProcessCmd, 745  
  CS\_RECOMP\_APP\_TASK\_NAME, 738  
  CS\_RECOMP\_CFECORE\_TASK\_NAME, 738  
  CS\_RECOMP\_EEPROM\_TASK\_NAME, 738  
  CS\_RECOMP\_MEMORY\_TASK\_NAME, 738  
  CS\_RECOMP\_OS\_TASK\_NAME, 738  
  CS\_RECOMP\_TABLES\_TASK\_NAME, 738  
  CS\_TABLE\_ERROR, 738  
  CS\_TABLETYPE\_NAME\_SIZE, 738  
  CS\_UpdateCDS, 747  
  CS\_WAKEUP\_TIMEOUT, 738  
cs\_app\_cmds.c  
  CS\_DisableAppCmd, 749  
  CS\_DisableNameAppCmd, 750  
  CS\_EnableAppCmd, 751  
  CS\_EnableNameAppCmd, 752  
  CS\_RecomputeBaselineAppCmd, 753  
  CS\_ReportBaselineAppCmd, 754  
cs\_app\_cmds.h

CS\_DisableAppCmd, 755  
 CS\_DisableNameAppCmd, 756  
 CS\_EnableAppCmd, 757  
 CS\_EnableNameAppCmd, 758  
 CS\_RecomputeBaselineAppCmd, 759  
 CS\_ReportBaselineAppCmd, 760  
**CS\_APP\_MISCOMPARE\_ERR\_EID**  
 CFS Checksum Event IDs, 132  
**CS\_APP\_TABLE**  
 cs\_msgdefs.h, 708  
**CS\_AppData**  
 cs\_app.c, 735  
 cs\_app.h, 748  
**CS\_AppData\_t**, 654  
 AppResTablesTblPtr, 656  
 CfeCoreCodeSeg, 656  
 ChildTaskEntryID, 657  
 ChildTaskID, 657  
 ChildTaskTable, 657  
 CmdPipe, 657  
 DataStoreHandle, 657  
 DefAppTableHandle, 657  
 DefAppTblPtr, 657  
 DefaultAppDefTable, 657  
 DefaultEepromDefTable, 658  
 DefaultMemoryDefTable, 658  
 DefaultTablesDefTable, 658  
 DefEepromTableHandle, 658  
 DefEepromTblPtr, 658  
 DefMemoryTableHandle, 658  
 DefMemoryTblPtr, 658  
 DefTablesTableHandle, 659  
 DefTablesTblPtr, 659  
 EepResTablesTblPtr, 659  
 HkPacket, 659  
 MaxBytesPerCycle, 659  
 MemResTablesTblPtr, 660  
 OSCodeSeg, 660  
 PipeDepth, 660  
 PipeName, 660  
 RecomputeAppEntryPtr, 660  
 RecomputeEepromMemoryEntryPtr, 660  
 RecomputeTablesEntryPtr, 660  
 ResAppTableHandle, 660  
 ResAppTblPtr, 661  
 ResEepromTableHandle, 661  
 ResEepromTblPtr, 661  
 ResMemoryTableHandle, 661  
 ResMemoryTblPtr, 661  
 ResTablesTableHandle, 661  
 ResTablesTblPtr, 661  
 RunStatus, 662  
 TblResTablesTblPtr, 662  
**CS\_AppInit**  
 cs\_app.c, 725  
 cs\_app.h, 739  
**CS\_AppMain**  
 cs\_app.c, 726  
 cs\_app.h, 740  
**CS\_APPMAIN\_PERF\_ID**  
 CFS Checksum Mission Configuration, 172  
**CS\_AppNameCmd\_Payload\_t**, 662  
 Name, 662  
**CS\_AppNameCmd\_t**, 662  
 CmdHeader, 663  
 Payload, 663  
**CS\_AppPipe**  
 cs\_app.c, 728  
 cs\_app.h, 741  
**CS\_APPS\_TBL\_POWERON\_STATE**  
 CFS Checksum Platform Configuration, 211  
**CS\_AppTable**  
 cs\_apptbl.c, 905  
**cs\_apptbl.c**  
 CS\_AppTable, 905  
**CS\_AttemptTableReshare**  
 cs\_utils.c, 866  
 cs\_utils.h, 886  
**CS\_BACKGROUND\_CYCLE\_MID**  
 CFS Checksum Command Message IDs, 208  
**CS\_BackgroundApp**  
 cs\_utils.c, 867  
 cs\_utils.h, 887  
**CS\_BackgroundCfeCore**  
 cs\_utils.c, 868  
 cs\_utils.h, 888  
**CS\_BackgroundCheckCycle**  
 cs\_cmds.c, 762  
 cs\_cmds.h, 777  
**CS\_BackgroundEeprom**  
 cs\_utils.c, 869  
 cs\_utils.h, 889  
**CS\_BackgroundMemory**  
 cs\_utils.c, 870  
 cs\_utils.h, 890  
**CS\_BackgroundOS**  
 cs\_utils.c, 871  
 cs\_utils.h, 891  
**CS\_BackgroundTables**  
 cs\_utils.c, 872  
 cs\_utils.h, 892  
**CS\_BASELINE\_APP\_INF\_EID**  
 CFS Checksum Event IDs, 132  
**CS\_BASELINE\_CFECORE\_INF\_EID**  
 CFS Checksum Event IDs, 132  
**CS\_BASELINE\_EEPROM\_INF\_EID**  
 CFS Checksum Event IDs, 133  
**CS\_BASELINE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID**

CFS Checksum Event IDs, 133  
CS\_BASELINE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID  
    CFS Checksum Event IDs, 133  
CS\_BASELINE\_INVALID\_NAME\_APP\_ERR\_EID  
    CFS Checksum Event IDs, 133  
CS\_BASELINE\_INVALID\_NAME\_TABLES\_ERR\_EID  
    CFS Checksum Event IDs, 134  
CS\_BASELINE\_MEMORY\_INF\_EID  
    CFS Checksum Event IDs, 134  
CS\_BASELINE\_OS\_INF\_EID  
    CFS Checksum Event IDs, 134  
CS\_BASELINE\_TABLES\_INF\_EID  
    CFS Checksum Event IDs, 134  
CS\_BKGND\_COMPUTE\_PROG\_INF\_EID  
    CFS Checksum Event IDs, 135  
CS\_CANCEL\_ONE\_SHOT\_CC  
    CFS Checksum Command Codes, 176  
CS\_CANCEL\_ONESHOT\_CC  
    cs\_msgdefs.h, 708  
CS\_CancelOneShotCmd  
    cs\_cmdu.c, 763  
    cs\_cmdu.h, 778  
CS\_CC1\_ERR\_EID  
    CFS Checksum Event IDs, 135  
CS\_CDS\_NAME  
    CFS Checksum Platform Configuration, 211  
CS\_CFE\_TEXT\_SEG\_INF\_EID  
    CFS Checksum Event IDs, 135  
CS\_CFECORE  
    cs\_msgdefs.h, 708  
CS\_CFECORE\_CHECKSUM\_STATE  
    CFS Checksum Platform Configuration, 211  
CS\_CFECORE\_MISCOMPARE\_ERR\_EID  
    CFS Checksum Event IDs, 135  
CS\_CheckRecomputeOneshot  
    cs\_utils.c, 873  
    cs\_utils.h, 893  
CS\_CHILD\_TASK\_DELAY  
    CFS Checksum Platform Configuration, 211  
CS\_CHILD\_TASK\_PRIORITY  
    CFS Checksum Platform Configuration, 211  
CS\_CMD\_COMPUTE\_PROG\_ERR\_EID  
    CFS Checksum Event IDs, 136  
CS\_CMD\_MID  
    CFS Checksum Command Message IDs, 208  
CS\_CMD\_PIPE\_NAME  
    cs\_app.h, 737  
CS\_CMD\_PIPE\_NAME\_LEN  
    cs\_app.h, 737  
cs\_cmdu.c  
    CS\_BackgroundCheckCycle, 762  
    CS\_CancelOneShotCmd, 763  
    CS\_DisableAllCSCmd, 764  
    CS\_DisableCfeCoreCmd, 765  
    CS\_DisableOSCmd, 766  
    CS\_EnableAllCSCmd, 767  
    CS\_EnableCfeCoreCmd, 768  
    CS\_EnableOSCmd, 769  
    CS\_NoopCmd, 770  
    CS\_OneShotCmd, 771  
    CS\_RecomputeBaselineCfeCoreCmd, 772  
    CS\_RecomputeBaselineOSCmd, 773  
    CS\_ReportBaselineCfeCoreCmd, 774  
    CS\_ReportBaselineOSCmd, 775  
    CS\_ResetCmd, 775  
cs\_cmdu.h  
    CS\_BackgroundCheckCycle, 777  
    CS\_CancelOneShotCmd, 778  
    CS\_DisableAllCSCmd, 779  
    CS\_DisableCfeCoreCmd, 780  
    CS\_DisableOSCmd, 781  
    CS\_EnableAllCSCmd, 782  
    CS\_EnableCfeCoreCmd, 783  
    CS\_EnableOSCmd, 784  
    CS\_NoopCmd, 785  
    CS\_OneShotCmd, 786  
    CS\_RecomputeBaselineCfeCoreCmd, 787  
    CS\_RecomputeBaselineOSCmd, 788  
    CS\_ReportBaselineCfeCoreCmd, 789  
    CS\_ReportBaselineOSCmd, 790  
    CS\_ResetCmd, 790  
cs\_compute.c  
    CS\_ComputeApp, 792  
    CS\_ComputeEepromMemory, 793  
    CS\_ComputeTables, 794  
    CS\_OneShotChildTask, 795  
    CS\_RecomputeAppChildTask, 796  
    CS\_RecomputeEepromMemoryChildTask, 797  
    CS\_RecomputeTablesChildTask, 798  
cs\_compute.h  
    CS\_ComputeApp, 799  
    CS\_ComputeEepromMemory, 801  
    CS\_ComputeTables, 802  
    CS\_OneShotChildTask, 803  
    CS\_RecomputeAppChildTask, 804  
    CS\_RecomputeEepromMemoryChildTask, 805  
    CS\_RecomputeTablesChildTask, 806  
CS COMPUTE\_APP\_ERR\_EID  
    CFS Checksum Event IDs, 136  
CS COMPUTE\_APP\_NOT\_FOUND\_ERR\_EID  
    CFS Checksum Event IDs, 136  
CS COMPUTE\_APP\_PLATFORM\_DBG\_EID  
    CFS Checksum Event IDs, 136  
CS COMPUTE\_TABLES\_ERR\_EID  
    CFS Checksum Event IDs, 137  
CS COMPUTE\_TABLES\_NOT\_FOUND\_ERR\_EID  
    CFS Checksum Event IDs, 137  
CS COMPUTE\_TABLES\_RELEASE\_ERR\_EID

CFS Checksum Event IDs, [137](#)

**CS\_ComputeApp**  
    cs\_compute.c, [792](#)  
    cs\_compute.h, [799](#)

**CS\_ComputeEepromMemory**  
    cs\_compute.c, [793](#)  
    cs\_compute.h, [801](#)

**CS\_ComputeTables**  
    cs\_compute.c, [794](#)  
    cs\_compute.h, [802](#)

**CS\_CR\_CDS\_CPY\_ERR\_EID**  
    CFS Checksum Event IDs, [137](#)

**CS\_CR\_CDS\_REG\_ERR\_EID**  
    CFS Checksum Event IDs, [138](#)

**CS\_CR\_CDS\_RES\_ERR\_EID**  
    CFS Checksum Event IDs, [138](#)

**CS\_CreateRestoreStatesFromCDS**  
    cs\_app.c, [730](#)  
    cs\_app.h, [743](#)

**CS\_Def\_App\_Table\_Entry\_t**, [663](#)  
    Name, [663](#)  
    State, [663](#)

**CS\_DEF\_APP\_TABLE\_FILENAME**  
    CFS Checksum Platform Configuration, [212](#)

**CS\_DEF\_APP\_TABLE\_NAME**  
    cs\_tbldefs.h, [713](#)

**CS\_DEF\_EEPROM\_TABLE\_FILENAME**  
    CFS Checksum Platform Configuration, [212](#)

**CS\_DEF\_EEPROM\_TABLE\_NAME**  
    cs\_tbldefs.h, [713](#)

**CS\_Def\_EepromMemory\_Table\_Entry\_t**, [664](#)  
    Filler16, [664](#)  
    NumBytesToChecksum, [664](#)  
    StartAddress, [664](#)  
    State, [664](#)

**CS\_DEF\_MEMORY\_TABLE\_FILENAME**  
    CFS Checksum Platform Configuration, [212](#)

**CS\_DEF\_MEMORY\_TABLE\_NAME**  
    cs\_tbldefs.h, [713](#)

**CS\_Def\_Tables\_Table\_Entry\_t**, [665](#)  
    Name, [665](#)  
    State, [665](#)

**CS\_DEF\_TABLES\_TABLE\_FILENAME**  
    CFS Checksum Platform Configuration, [212](#)

**CS\_DEF\_TABLES\_TABLE\_NAME**  
    cs\_tbldefs.h, [713](#)

**CS\_DEFAULT\_ALGORITHM**  
    CFS Checksum Mission Configuration, [172](#)

**CS\_DEFAULT\_BYTES\_PER\_CYCLE**  
    CFS Checksum Platform Configuration, [213](#)

**CS\_DISABLE\_ALL\_CS\_CC**  
    CFS Checksum Command Codes, [177](#)

**CS\_DISABLE\_ALL\_INF\_EID**  
    CFS Checksum Event IDs, [138](#)

**CS\_DISABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID**  
    CFS Checksum Event IDs, [138](#)

**CS\_DISABLE\_APP\_INF\_EID**  
    CFS Checksum Event IDs, [139](#)

**CS\_DISABLE\_APP\_NAME\_INF\_EID**  
    CFS Checksum Event IDs, [139](#)

**CS\_DISABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID**  
    CFS Checksum Event IDs, [139](#)

**CS\_DISABLE\_APPS\_CC**  
    CFS Checksum Command Codes, [178](#)

**CS\_DISABLE\_CFE\_CORE\_CC**  
    CFS Checksum Command Codes, [179](#)

**CS\_DISABLE\_CFECORE\_CC**  
    cs\_msgdefs.h, [709](#)

**CS\_DISABLE\_CFECORE\_INF\_EID**  
    CFS Checksum Event IDs, [139](#)

**CS\_DISABLE\_EEPROM\_CC**  
    CFS Checksum Command Codes, [179](#)

**CS\_DISABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID**  
    CFS Checksum Event IDs, [140](#)

**CS\_DISABLE\_EEPROM\_ENTRY\_INF\_EID**  
    CFS Checksum Event IDs, [140](#)

**CS\_DISABLE\_EEPROM\_INF\_EID**  
    CFS Checksum Event IDs, [140](#)

**CS\_DISABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID**  
    CFS Checksum Event IDs, [140](#)

**CS\_DISABLE\_ENTRY\_EEPROM\_CC**  
    CFS Checksum Command Codes, [180](#)

**CS\_DISABLE\_ENTRY\_MEMORY\_CC**  
    CFS Checksum Command Codes, [181](#)

**CS\_DISABLE\_MEMORY\_CC**  
    CFS Checksum Command Codes, [182](#)

**CS\_DISABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID**  
    CFS Checksum Event IDs, [141](#)

**CS\_DISABLE\_MEMORY\_ENTRY\_INF\_EID**  
    CFS Checksum Event IDs, [141](#)

**CS\_DISABLE\_MEMORY\_INF\_EID**  
    CFS Checksum Event IDs, [141](#)

**CS\_DISABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID**  
    CFS Checksum Event IDs, [141](#)

**CS\_DISABLE\_NAME\_APP\_CC**  
    CFS Checksum Command Codes, [182](#)

**CS\_DISABLE\_NAME\_TABLE\_CC**  
    CFS Checksum Command Codes, [183](#)

**CS\_DISABLE\_OS\_CC**  
    CFS Checksum Command Codes, [184](#)

**CS\_DISABLE\_OS\_INF\_EID**  
    CFS Checksum Event IDs, [142](#)

**CS\_DISABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID**  
    CFS Checksum Event IDs, [142](#)

**CS\_DISABLE\_TABLES\_CC**  
    CFS Checksum Command Codes, [185](#)

**CS\_DISABLE\_TABLES\_INF\_EID**  
    CFS Checksum Event IDs, [142](#)

CS\_DISABLE\_TABLES\_NAME\_INF\_EID  
    CFS Checksum Event IDs, 142

CS\_DISABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID  
    CFS Checksum Event IDs, 143

CS\_DisableAllCSCmd  
    cs\_cmds.c, 764  
    cs\_cmds.h, 779

CS\_DisableAppCmd  
    cs\_app\_cmds.c, 749  
    cs\_app\_cmds.h, 755

CS\_DisableCfeCoreCmd  
    cs\_cmds.c, 765  
    cs\_cmds.h, 780

CS\_DisableEepromCmd  
    cs\_eeprom\_cmds.c, 808  
    cs\_eeprom\_cmds.h, 814

CS\_DisableEntryIDEepromCmd  
    cs\_eeprom\_cmds.c, 808  
    cs\_eeprom\_cmds.h, 815

CS\_DisableEntryIDMemoryCmd  
    cs\_memory\_cmds.c, 829  
    cs\_memory\_cmds.h, 835

CS\_DisableMemoryCmd  
    cs\_memory\_cmds.c, 830  
    cs\_memory\_cmds.h, 836

CS\_DisableNameAppCmd  
    cs\_app\_cmds.c, 750  
    cs\_app\_cmds.h, 756

CS\_DisableNameTablesCmd  
    cs\_table\_cmds.c, 843  
    cs\_table\_cmds.h, 848

CS\_DisableOSCmd  
    cs\_cmds.c, 766  
    cs\_cmds.h, 781

CS\_DisableTablesCmd  
    cs\_table\_cmds.c, 844  
    cs\_table\_cmds.h, 849

cs\_eeprom\_cmds.c  
    CS\_DisableEepromCmd, 808  
    CS\_DisableEntryIDEepromCmd, 808  
    CS\_EnableEepromCmd, 809  
    CS\_EnableEntryIDEepromCmd, 810  
    CS\_GetEntryIDEepromCmd, 811  
    CS\_RecomputeBaselineEepromCmd, 812  
    CS\_ReportBaselineEntryIDEepromCmd, 813

cs\_eeprom\_cmds.h  
    CS\_DisableEepromCmd, 814  
    CS\_DisableEntryIDEepromCmd, 815  
    CS\_EnableEepromCmd, 816  
    CS\_EnableEntryIDEepromCmd, 817  
    CS\_GetEntryIDEepromCmd, 818  
    CS\_RecomputeBaselineEepromCmd, 819  
    CS\_ReportBaselineEntryIDEepromCmd, 819

CS EEPROM MISCOMPARE\_ERR\_EID  
    CFS Checksum Event IDs, 143

CS EEPROM\_TABLE  
    cs\_msgdefs.h, 709

CS EEPROM\_TBL\_POWERON\_STATE  
    CFS Checksum Platform Configuration, 213

CS\_EepromTable  
    cs\_eepromtbl.c, 906

cs\_eepromtbl.c  
    CS\_EepromTable, 906

CS\_ENABLE\_ALL\_CS\_CC  
    CFS Checksum Command Codes, 185

CS\_ENABLE\_ALL\_INF\_EID  
    CFS Checksum Event IDs, 143

CS\_ENABLE\_APP\_DEF\_NOT\_FOUND\_DBG\_EID  
    CFS Checksum Event IDs, 143

CS\_ENABLE\_APP\_INF\_EID  
    CFS Checksum Event IDs, 144

CS\_ENABLE\_APP\_NAME\_INF\_EID  
    CFS Checksum Event IDs, 144

CS\_ENABLE\_APP\_UNKNOWN\_NAME\_ERR\_EID  
    CFS Checksum Event IDs, 144

CS\_ENABLE\_APPS\_CC  
    CFS Checksum Command Codes, 186

CS\_ENABLE\_CFE\_CORE\_CC  
    CFS Checksum Command Codes, 187

CS\_ENABLE\_CFECORE\_CC  
    cs\_msgdefs.h, 709

CS\_ENABLE\_CFECORE\_INF\_EID  
    CFS Checksum Event IDs, 144

CS\_ENABLE\_EEPROM\_CC  
    CFS Checksum Command Codes, 188

CS\_ENABLE\_EEPROM\_DEF\_EMPTY\_DBG\_EID  
    CFS Checksum Event IDs, 145

CS\_ENABLE\_EEPROM\_ENTRY\_INF\_EID  
    CFS Checksum Event IDs, 145

CS\_ENABLE\_EEPROM\_INF\_EID  
    CFS Checksum Event IDs, 145

CS\_ENABLE\_EEPROM\_INVALID\_ENTRY\_ERR\_EID  
    CFS Checksum Event IDs, 145

CS\_ENABLE\_ENTRY\_EEPROM\_CC  
    CFS Checksum Command Codes, 188

CS\_ENABLE\_ENTRY\_MEMORY\_CC  
    CFS Checksum Command Codes, 189

CS\_ENABLE\_MEMORY\_CC  
    CFS Checksum Command Codes, 190

CS\_ENABLE\_MEMORY\_DEF\_EMPTY\_DBG\_EID  
    CFS Checksum Event IDs, 146

CS\_ENABLE\_MEMORY\_ENTRY\_INF\_EID  
    CFS Checksum Event IDs, 146

CS\_ENABLE\_MEMORY\_INF\_EID  
    CFS Checksum Event IDs, 146

CS\_ENABLE\_MEMORY\_INVALID\_ENTRY\_ERR\_EID  
    CFS Checksum Event IDs, 146

CS\_ENABLE\_NAME\_APP\_CC

CFS Checksum Command Codes, 191  
**CS\_ENABLE\_NAME\_TABLE\_CC**  
     CFS Checksum Command Codes, 191  
**CS\_ENABLE\_OS\_CC**  
     CFS Checksum Command Codes, 192  
**CS\_ENABLE\_OS\_INF\_EID**  
     CFS Checksum Event IDs, 147  
**CS\_ENABLE\_TABLE\_DEF\_NOT\_FOUND\_DBG\_EID**  
     CFS Checksum Event IDs, 147  
**CS\_ENABLE\_TABLES\_CC**  
     CFS Checksum Command Codes, 193  
**CS\_ENABLE\_TABLES\_INF\_EID**  
     CFS Checksum Event IDs, 147  
**CS\_ENABLE\_TABLES\_NAME\_INF\_EID**  
     CFS Checksum Event IDs, 147  
**CS\_ENABLE\_TABLES\_UNKNOWN\_NAME\_ERR\_EID**  
     CFS Checksum Event IDs, 148  
**CS\_EnableAllCSCmd**  
     cs\_cmds.c, 767  
     cs\_cmds.h, 782  
**CS\_EnableAppCmd**  
     cs\_app\_cmds.c, 751  
     cs\_app\_cmds.h, 757  
**CS\_EnableCfeCoreCmd**  
     cs\_cmds.c, 768  
     cs\_cmds.h, 783  
**CS\_EnableEepromCmd**  
     cs\_eeprom\_cmds.c, 809  
     cs\_eeprom\_cmds.h, 816  
**CS\_EnableEntryIDEepromCmd**  
     cs\_eeprom\_cmds.c, 810  
     cs\_eeprom\_cmds.h, 817  
**CS\_EnableEntryIDMemoryCmd**  
     cs\_memory\_cmds.c, 831  
     cs\_memory\_cmds.h, 837  
**CS\_EnableMemoryCmd**  
     cs\_memory\_cmds.c, 832  
     cs\_memory\_cmds.h, 838  
**CS\_EnableNameAppCmd**  
     cs\_app\_cmds.c, 752  
     cs\_app\_cmds.h, 758  
**CS\_EnableNameTablesCmd**  
     cs\_table\_cmds.c, 844  
     cs\_table\_cmds.h, 850  
**CS\_EnableOSCmd**  
     cs\_cmds.c, 769  
     cs\_cmds.h, 784  
**CS\_EnableTablesCmd**  
     cs\_table\_cmds.c, 845  
     cs\_table\_cmds.h, 851  
**CS\_EntryCmd\_Payload\_t**, 665  
     EntryID, 666  
**CS\_EntryCmd\_t**, 666  
     CmdHeader, 666  
  
     Payload, 666  
**CS\_ERR\_NOT\_FOUND**  
     cs\_app.h, 737  
**CS\_ERROR**  
     cs\_app.h, 737  
**CS\_EXIT\_ERR\_EID**  
     CFS Checksum Event IDs, 148  
**CS\_EXIT\_INF\_EID**  
     CFS Checksum Event IDs, 148  
**CS\_FindEnabledAppEntry**  
     cs\_utils.c, 873  
     cs\_utils.h, 893  
**CS\_FindEnabledEepromEntry**  
     cs\_utils.c, 874  
     cs\_utils.h, 894  
**CS\_FindEnabledMemoryEntry**  
     cs\_utils.c, 875  
     cs\_utils.h, 895  
**CS\_FindEnabledTablesEntry**  
     cs\_utils.c, 875  
     cs\_utils.h, 895  
**CS\_GET\_ENTRY\_ID\_EEPROM\_CC**  
     CFS Checksum Command Codes, 194  
**CS\_GET\_ENTRY\_ID\_EEPROM\_INF\_EID**  
     CFS Checksum Event IDs, 148  
**CS\_GET\_ENTRY\_ID\_EEPROM\_NOT\_FOUND\_INF\_EID**  
     CFS Checksum Event IDs, 149  
**CS\_GET\_ENTRY\_ID\_MEMORY\_CC**  
     CFS Checksum Command Codes, 194  
**CS\_GET\_ENTRY\_ID\_MEMORY\_INF\_EID**  
     CFS Checksum Event IDs, 149  
**CS\_GET\_ENTRY\_ID\_MEMORY\_NOT\_FOUND\_INF\_EID**  
     CFS Checksum Event IDs, 149  
**CS\_GetAppDefTblEntryByName**  
     cs\_utils.c, 876  
     cs\_utils.h, 896  
**CS\_GetAppResTblEntryByName**  
     cs\_utils.c, 877  
     cs\_utils.h, 897  
**CS\_GetEntryIDCmd\_Payload\_t**, 667  
     Address, 667  
**CS\_GetEntryIDCmd\_t**, 667  
     CmdHeader, 667  
     Payload, 667  
**CS\_GetEntryIDEepromCmd**  
     cs\_eeprom\_cmds.c, 811  
     cs\_eeprom\_cmds.h, 818  
**CS\_GetEntryIDMemoryCmd**  
     cs\_memory\_cmds.c, 832  
     cs\_memory\_cmds.h, 839  
**CS\_GetTableDefTblEntryByName**  
     cs\_utils.c, 877  
     cs\_utils.h, 897  
**CS\_GetTableResTblEntryByName**

cs\_utils.c, 878  
cs\_utils.h, 898  
CS\_GoToNextTable  
    cs\_utils.c, 879  
    cs\_utils.h, 899  
CS\_HandleRoutineTableUpdates  
    cs\_utils.c, 879  
    cs\_utils.h, 899  
CS\_HandleTableUpdate  
    cs\_table\_processing.c, 855  
    cs\_tbldefs.h, 714  
CS\_HK\_TLM\_MID  
    CFS Checksum Telemetry Message IDs, 209  
CS\_HkPacket\_Payload\_t, 668  
    AppCSErrCounter, 669  
    AppCSState, 669  
    CfeCoreBaseline, 669  
    CfeCoreCSErrCounter, 669  
    CfeCoreCSState, 670  
    ChecksumState, 670  
    CmdCounter, 670  
    CmdErrCounter, 670  
    CurrentCSTable, 670  
    CurrentEntryInTable, 670  
    EepromBaseline, 671  
    EepromCSErrCounter, 671  
    EepromCSState, 671  
    Filler8, 671  
    LastOneShotAddress, 671  
    LastOneShotChecksum, 671  
    LastOneShotMaxBytesPerCycle, 671  
    LastOneShotSize, 671  
    MemoryCSErrCounter, 672  
    MemoryCSState, 672  
    OneShotInProgress, 672  
    OSBaseline, 672  
    OSCSErrCounter, 672  
    OSCSState, 672  
    PassCounter, 672  
    RecomputeInProgress, 672  
    TablesCSErrCounter, 673  
    TablesCSState, 673  
CS\_HkPacket\_t, 673  
    Payload, 673  
    TlmHeader, 674  
CS\_HousekeepingCmd  
    cs\_app.c, 731  
    cs\_app.h, 744  
cs\_init.c  
    CS\_InitAllTables, 821  
    CS\_InitSegments, 822  
    CS\_SbInit, 823  
cs\_init.h  
    CS\_InitAllTables, 824  
    CS\_InitSegments, 826  
    CS\_SbInit, 827  
CS\_INIT\_APP\_ERR\_EID  
    CFS Checksum Event IDs, 150  
CS\_INIT EEPROM\_ERR\_EID  
    CFS Checksum Event IDs, 150  
CS\_INIT\_INF\_EID  
    CFS Checksum Event IDs, 150  
CS\_INIT\_MEMORY\_ERR\_EID  
    CFS Checksum Event IDs, 150  
CS\_INIT\_SB\_CREATE\_ERR\_EID  
    CFS Checksum Event IDs, 151  
CS\_INIT\_SB\_SUBSCRIBE\_BACK\_ERR\_EID  
    CFS Checksum Event IDs, 151  
CS\_INIT\_SB\_SUBSCRIBE\_CMD\_ERR\_EID  
    CFS Checksum Event IDs, 151  
CS\_INIT\_SB\_SUBSCRIBE\_HK\_ERR\_EID  
    CFS Checksum Event IDs, 151  
CS\_INIT\_TABLES\_ERR\_EID  
    CFS Checksum Event IDs, 152  
CS\_InitAllTables  
    cs\_init.c, 821  
    cs\_init.h, 824  
CS\_InitializeDefaultTables  
    cs\_utils.c, 880  
    cs\_utils.h, 900  
CS\_InitSegments  
    cs\_init.c, 822  
    cs\_init.h, 826  
CS\_LEN\_ERR\_EID  
    CFS Checksum Event IDs, 152  
CS\_MAJOR\_VERSION  
    CFS Checksum Version, 217  
CS\_MAX\_NUM\_APP\_TABLE\_ENTRIES  
    CFS Checksum Platform Configuration, 213  
CS\_MAX\_NUM EEPROM\_TABLE\_ENTRIES  
    CFS Checksum Platform Configuration, 213  
CS\_MAX\_NUM\_MEMORY\_TABLE\_ENTRIES  
    CFS Checksum Platform Configuration, 214  
CS\_MAX\_NUM\_TABLES\_TABLE\_ENTRIES  
    CFS Checksum Platform Configuration, 214  
cs\_memory\_cmds.c  
    CS\_DisableEntryIDMemoryCmd, 829  
    CS\_DisableMemoryCmd, 830  
    CS\_EnableEntryIDMemoryCmd, 831  
    CS\_EnableMemoryCmd, 832  
    CS\_GetEntryIDMemoryCmd, 832  
    CS\_RecomputeBaselineMemoryCmd, 833  
    CS\_ReportBaselineEntryIDMemoryCmd, 834  
cs\_memory\_cmds.h  
    CS\_DisableEntryIDMemoryCmd, 835  
    CS\_DisableMemoryCmd, 836  
    CS\_EnableEntryIDMemoryCmd, 837  
    CS\_EnableMemoryCmd, 838

CS\_GetEntryIDMemoryCmd, 839  
 CS\_RecomputeBaselineMemoryCmd, 840  
 CS\_ReportBaselineEntryIDMemoryCmd, 841  
**CS\_MEMORY\_MISCOMPARE\_ERR\_EID**  
     CFS Checksum Event IDs, 152  
**CS\_MEMORY\_TABLE**  
     cs\_msgdefs.h, 709  
**CS\_MEMORY\_TBL\_POWERON\_STATE**  
     CFS Checksum Platform Configuration, 214  
**CS\_MemoryTable**  
     cs\_memorytbl.c, 906  
**cs\_memorytbl.c**  
     CS\_MemoryTable, 906  
**CS\_MID\_ERR\_EID**  
     CFS Checksum Event IDs, 152  
**CS\_MINOR\_VERSION**  
     CFS Checksum Version, 217  
**CS\_MISSION\_REV**  
     CFS Checksum Platform Configuration, 214  
**cs\_msgdefs.h**  
     CS\_APP\_TABLE, 708  
     CS\_CANCEL\_ONESHOT\_CC, 708  
     CS\_CFECORE, 708  
     CS\_DISABLE\_CFECORE\_CC, 709  
     CS\_EEPROM\_TABLE, 709  
     CS\_ENABLE\_CFECORE\_CC, 709  
     CS\_MEMORY\_TABLE, 709  
     CS\_NUM\_TABLES, 709  
     CS\_ONESHOT\_CC, 709  
     CS\_OSCORE, 709  
     CS\_RECOMPUTE\_BASELINE\_CFECORE\_CC, 709  
     CS\_REPORT\_BASELINE\_CFECORE\_CC, 709  
     CS\_STATE\_DISABLED, 709  
     CS\_STATE\_EMPTY, 709  
     CS\_STATE\_ENABLED, 710  
     CS\_STATE\_UNDEFINED, 710  
     CS\_TABLES\_TABLE, 710  
**CS\_NO\_BASELINE\_APP\_INF\_EID**  
     CFS Checksum Event IDs, 153  
**CS\_NO\_BASELINE\_CFECORE\_INF\_EID**  
     CFS Checksum Event IDs, 153  
**CS\_NO\_BASELINE\_EEPROM\_INF\_EID**  
     CFS Checksum Event IDs, 153  
**CS\_NO\_BASELINE\_MEMORY\_INF\_EID**  
     CFS Checksum Event IDs, 153  
**CS\_NO\_BASELINE\_OS\_INF\_EID**  
     CFS Checksum Event IDs, 154  
**CS\_NO\_BASELINE\_TABLES\_INF\_EID**  
     CFS Checksum Event IDs, 154  
**CS\_NoArgsCmd\_t**, 674  
     CmdHeader, 674  
**CS\_NOOP\_CC**  
     CFS Checksum Command Codes, 195  
**CS\_NOOP\_INF\_EID**

CFS Checksum Event IDs, 154  
**CS\_NoopCmd**  
     cs\_cmds.c, 770  
     cs\_cmds.h, 785  
**CS\_NUM\_DATA\_STORE\_STATES**  
     cs\_app.c, 725  
**CS\_NUM\_TABLES**  
     cs\_msgdefs.h, 709  
**CS\_ONE\_SHOT\_CC**  
     CFS Checksum Command Codes, 196  
**CS\_ONESHOT\_CANCEL\_DELETE\_CHDTASK\_ERR\_EID**  
     CFS Checksum Event IDs, 154  
**CS\_ONESHOT\_CANCEL\_NO\_CHDTASK\_ERR\_EID**  
     CFS Checksum Event IDs, 155  
**CS\_ONESHOT\_CANCELLED\_INF\_EID**  
     CFS Checksum Event IDs, 155  
**CS\_ONESHOT\_CC**  
     cs\_msgdefs.h, 709  
**CS\_ONESHOT\_CHDTASK\_ERR\_EID**  
     CFS Checksum Event IDs, 155  
**CS\_ONESHOT\_CREATE\_CHDTASK\_ERR\_EID**  
     CFS Checksum Event IDs, 155  
**CS\_ONESHOT\_FINISHED\_INF\_EID**  
     CFS Checksum Event IDs, 156  
**CS\_ONESHOT\_MEMVALIDATE\_ERR\_EID**  
     CFS Checksum Event IDs, 156  
**CS\_ONESHOT\_STARTED\_DBG\_EID**  
     CFS Checksum Event IDs, 156  
**CS\_ONESHOT\_TASK\_NAME**  
     cs\_app.h, 738  
**CS\_OneShotChildTask**  
     cs\_compute.c, 795  
     cs\_compute.h, 803  
**CS\_OneShotCmd**  
     cs\_cmds.c, 771  
     cs\_cmds.h, 786  
**CS\_OneShotCmd\_Payload\_t**, 675  
     Address, 675  
     MaxBytesPerCycle, 675  
     Size, 675  
**CS\_OneShotCmd\_t**, 675  
     CmdHeader, 676  
     Payload, 676  
**CS\_OS\_MISCOMPARE\_ERR\_EID**  
     CFS Checksum Event IDs, 156  
**CS\_OS\_TEXT\_SEG\_INF\_EID**  
     CFS Checksum Event IDs, 157  
**CS\_OSCORE**  
     cs\_msgdefs.h, 709  
**CS\_OSCS\_CHECKSUM\_STATE**  
     CFS Checksum Platform Configuration, 215  
**CS\_PIPE\_DEPTH**  
     CFS Checksum Platform Configuration, 215  
**CS\_PRESERVE\_STATES\_ON\_PROCESSOR\_RESET**

CFS Checksum Platform Configuration, 215  
CS\_PROCESS\_APP\_NO\_ENTRIES\_INF\_EID  
    CFS Checksum Event IDs, 157  
CS\_PROCESS\_EEPROM\_MEMORY\_NO\_ENTRIES\_INF\_EID  
    CFS Checksum Event IDs, 157  
CS\_PROCESS\_TABLES\_NO\_ENTRIES\_INF\_EID  
    CFS Checksum Event IDs, 157  
CS\_ProcessCmd  
    cs\_app.c, 732  
    cs\_app.h, 745  
CS\_ProcessNewAppDefinitionTable  
    cs\_table\_processing.c, 856  
    cs\_tbldefs.h, 715  
CS\_ProcessNewEepromMemoryDefinitionTable  
    cs\_table\_processing.c, 857  
    cs\_tbldefs.h, 716  
CS\_ProcessNewTablesDefinitionTable  
    cs\_table\_processing.c, 858  
    cs\_tbldefs.h, 717  
CS\_RECOMP\_APP\_TASK\_NAME  
    cs\_app.h, 738  
CS\_RECOMP\_CFECORE\_TASK\_NAME  
    cs\_app.h, 738  
CS\_RECOMP\_EEPROM\_TASK\_NAME  
    cs\_app.h, 738  
CS\_RECOMP\_MEMORY\_TASK\_NAME  
    cs\_app.h, 738  
CS\_RECOMP\_OS\_TASK\_NAME  
    cs\_app.h, 738  
CS\_RECOMP\_TABLES\_TASK\_NAME  
    cs\_app.h, 738  
CS\_RECOMPUTE\_APP\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 158  
CS\_RECOMPUTE\_APP\_CREATE\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 158  
CS\_RECOMPUTE\_APP\_STARTED\_DBG\_EID  
    CFS Checksum Event IDs, 158  
CS\_RECOMPUTE\_BASELINE\_APP\_CC  
    CFS Checksum Command Codes, 197  
CS\_RECOMPUTE\_BASELINE\_CFE\_CORE\_CC  
    CFS Checksum Command Codes, 198  
CS\_RECOMPUTE\_BASELINE\_CFECORE\_CC  
    cs\_msgdefs.h, 709  
CS\_RECOMPUTE\_BASELINE\_EEPROM\_CC  
    CFS Checksum Command Codes, 199  
CS\_RECOMPUTE\_BASELINE\_MEMORY\_CC  
    CFS Checksum Command Codes, 200  
CS\_RECOMPUTE\_BASELINE\_OS\_CC  
    CFS Checksum Command Codes, 201  
CS\_RECOMPUTE\_BASELINE\_TABLE\_CC  
    CFS Checksum Command Codes, 201  
CS\_RECOMPUTE\_CFECORE\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 158  
CS\_RECOMPUTE\_CFECORE\_CREATE\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 159  
CS\_RECOMPUTE\_CFECORE\_STARTED\_DBG\_EID  
    CFS Checksum Event IDs, 159  
CS\_RECOMPUTE\_EEPROM\_CREATE\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 160  
CS\_RECOMPUTE\_EEPROM\_STARTED\_DBG\_EID  
    CFS Checksum Event IDs, 160  
CS\_RECOMPUTE\_ERROR\_APP\_ERR\_EID  
    CFS Checksum Event IDs, 160  
CS\_RECOMPUTE\_ERROR\_TABLES\_ERR\_EID  
    CFS Checksum Event IDs, 160  
CS\_RECOMPUTE\_FINISH\_APP\_INF\_EID  
    CFS Checksum Event IDs, 161  
CS\_RECOMPUTE\_FINISH\_EEPROM\_MEMORY\_INF\_EID  
    CFS Checksum Event IDs, 161  
CS\_RECOMPUTE\_FINISH\_TABLES\_INF\_EID  
    CFS Checksum Event IDs, 161  
CS\_RECOMPUTE\_INVALID\_ENTRY\_EEPROM\_ERR\_EID  
    CFS Checksum Event IDs, 161  
CS\_RECOMPUTE\_INVALID\_ENTRY\_MEMORY\_ERR\_EID  
    CFS Checksum Event IDs, 162  
CS\_RECOMPUTE\_MEMORY\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 162  
CS\_RECOMPUTE\_MEMORY\_CREATE\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 162  
CS\_RECOMPUTE\_MEMORY\_STARTED\_DBG\_EID  
    CFS Checksum Event IDs, 163  
CS\_RECOMPUTE\_OS\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 163  
CS\_RECOMPUTE\_OS\_CREATE\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 163  
CS\_RECOMPUTE\_OS\_STARTED\_DBG\_EID  
    CFS Checksum Event IDs, 163  
CS\_RECOMPUTE\_TABLES\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 164  
CS\_RECOMPUTE\_TABLES\_CREATE\_CHDTASK\_ERR\_EID  
    CFS Checksum Event IDs, 164  
CS\_RECOMPUTE\_TABLES\_STARTED\_DBG\_EID  
    CFS Checksum Event IDs, 164  
CS\_RECOMPUTE\_UNKNOWN\_NAME\_APP\_ERR\_EID  
    CFS Checksum Event IDs, 165  
CS\_RECOMPUTE\_UNKNOWN\_NAME\_TABLES\_ERR\_EID  
    CFS Checksum Event IDs, 165  
CS\_RecomputeAppChildTask  
    cs\_compute.c, 796  
    cs\_compute.h, 804  
CS\_RecomputeBaselineAppCmd  
    cs\_app\_cmds.c, 753  
    cs\_app\_cmds.h, 759  
CS\_RecomputeBaselineCfeCoreCmd  
    cs\_cmds.c, 772  
    cs\_cmds.h, 787

CS\_RecomputeBaselineEepromCmd  
     cs\_eeprom\_cmds.c, 812  
     cs\_eeprom\_cmds.h, 819

CS\_RecomputeBaselineMemoryCmd  
     cs\_memory\_cmds.c, 833  
     cs\_memory\_cmds.h, 840

CS\_RecomputeBaselineOSCmd  
     cs\_cmds.c, 773  
     cs\_cmds.h, 788

CS\_RecomputeBaselineTablesCmd  
     cs\_table\_cmds.c, 846  
     cs\_table\_cmds.h, 852

CS\_RecomputeEepromMemoryChildTask  
     cs\_compute.c, 797  
     cs\_compute.h, 805

CS\_RecomputeTablesChildTask  
     cs\_compute.c, 798  
     cs\_compute.h, 806

CS\_REPORT\_BASELINE\_APP\_CC  
     CFS Checksum Command Codes, 202

CS\_REPORT\_BASELINE\_CFE\_CORE\_CC  
     CFS Checksum Command Codes, 203

CS\_REPORT\_BASELINE\_CFECORE\_CC  
     cs\_msgdefs.h, 709

CS\_REPORT\_BASELINE\_EEPROM\_CC  
     CFS Checksum Command Codes, 204

CS\_REPORT\_BASELINE\_MEMORY\_CC  
     CFS Checksum Command Codes, 205

CS\_REPORT\_BASELINE\_OS\_CC  
     CFS Checksum Command Codes, 205

CS\_REPORT\_BASELINE\_TABLE\_CC  
     CFS Checksum Command Codes, 206

CS\_ReportBaselineAppCmd  
     cs\_app\_cmds.c, 754  
     cs\_app\_cmds.h, 760

CS\_ReportBaselineCfeCoreCmd  
     cs\_cmds.c, 774  
     cs\_cmds.h, 789

CS\_ReportBaselineEntryIDEepromCmd  
     cs\_eeprom\_cmds.c, 813  
     cs\_eeprom\_cmds.h, 819

CS\_ReportBaselineEntryIDMemoryCmd  
     cs\_memory\_cmds.c, 834  
     cs\_memory\_cmds.h, 841

CS\_ReportBaselineOSCmd  
     cs\_cmds.c, 775  
     cs\_cmds.h, 790

CS\_ReportBaselineTablesCmd  
     cs\_table\_cmds.c, 847  
     cs\_table\_cmds.h, 853

CS\_Res\_App\_Table\_Entry\_t, 676  
     ByteOffset, 676  
     ComparisonValue, 677  
     ComputedYet, 677

Name, 677  
     NumBytesToChecksum, 677  
     StartAddress, 677  
     State, 677  
     TempChecksumValue, 677

CS\_Res\_EepromMemory\_Table\_Entry\_t, 678  
     ByteOffset, 678  
     ComparisonValue, 678  
     ComputedYet, 678  
     Filler32, 679  
     NumBytesToChecksum, 679  
     StartAddress, 679  
     State, 679  
     TempChecksumValue, 679

CS\_Res\_Tables\_Table\_Entry\_t, 679  
     ByteOffset, 680  
     ComparisonValue, 680  
     ComputedYet, 680  
     Filler8, 681  
     IsCSOwner, 681  
     Name, 681  
     NumBytesToChecksum, 681  
     StartAddress, 681  
     State, 681  
     TblHandle, 681  
     TempChecksumValue, 681

CS\_RESET\_CC  
     CFS Checksum Command Codes, 207

CS\_RESET\_DBG\_EID  
     CFS Checksum Event IDs, 165

CS\_ResetCmd  
     cs\_cmds.c, 775  
     cs\_cmds.h, 790

CS\_ResetTablesTblResultEntry  
     cs\_utils.c, 881  
     cs\_utils.h, 901

CS\_RESULTS\_APP\_TABLE\_NAME  
     cs\_tbldefs.h, 713

CS\_RESULTS\_EEPROM\_TABLE\_NAME  
     cs\_tbldefs.h, 713

CS\_RESULTS\_MEMORY\_TABLE\_NAME  
     cs\_tbldefs.h, 714

CS\_RESULTS\_TABLES\_TABLE\_NAME  
     cs\_tbldefs.h, 714

CS\_REVISION  
     CFS Checksum Version, 217

CS\_SblInit  
     cs\_init.c, 823  
     cs\_init.h, 827

CS\_SEND\_HK\_MID  
     CFS Checksum Command Message IDs, 208

CS\_STARTUP\_TIMEOUT  
     CFS Checksum Platform Configuration, 215

CS\_STATE\_DISABLED

cs\_msgdefs.h, 709  
CS\_STATE\_EMPTY  
    cs\_msgdefs.h, 709  
CS\_STATE\_ENABLED  
    cs\_msgdefs.h, 710  
CS\_STATE\_UNDEFINED  
    cs\_msgdefs.h, 710  
cs\_table\_cmds.c  
    CS\_DisableNameTablesCmd, 843  
    CS\_DisableTablesCmd, 844  
    CS\_EnableNameTablesCmd, 844  
    CS\_EnableTablesCmd, 845  
    CS\_RecomputeBaselineTablesCmd, 846  
    CS\_ReportBaselineTablesCmd, 847  
cs\_table\_cmds.h  
    CS\_DisableNameTablesCmd, 848  
    CS\_DisableTablesCmd, 849  
    CS\_EnableNameTablesCmd, 850  
    CS\_EnableTablesCmd, 851  
    CS\_RecomputeBaselineTablesCmd, 852  
    CS\_ReportBaselineTablesCmd, 853  
CS\_TABLE\_ERROR  
    cs\_app.h, 738  
cs\_table\_processing.c  
    CS\_HandleTableUpdate, 855  
    CS\_ProcessNewAppDefinitionTable, 856  
    CS\_ProcessNewEepromMemoryDefinitionTable, 857  
    CS\_ProcessNewTablesDefinitionTable, 858  
    CS\_TableInit, 859  
    CS\_ValidateAppChecksumDefinitionTable, 861  
    CS\_ValidateEepromChecksumDefinitionTable, 862  
    CS\_ValidateMemoryChecksumDefinitionTable, 863  
    CS\_ValidateTablesChecksumDefinitionTable, 864  
CS\_TableInit  
    cs\_table\_processing.c, 859  
    cs\_tbldefs.h, 718  
CS\_TableNameCmd\_Payload\_t, 682  
    Name, 682  
CS\_TableNameCmd\_t, 682  
    CmdHeader, 683  
    Payload, 683  
CS\_TABLES\_MISCOMPARE\_ERR\_EID  
    CFS Checksum Event IDs, 165  
CS\_TABLES\_TABLE  
    cs\_msgdefs.h, 710  
CS\_TABLES\_TBL\_POWERON\_STATE  
    CFS Checksum Platform Configuration, 216  
CS\_TablesTable  
    cs\_tablestbl.c, 907  
cs\_tablestbl.c  
    CS\_TablesTable, 907  
CS\_TABLETYPE\_NAME\_SIZE  
    cs\_app.h, 738  
CS\_TBL\_INIT\_ERR\_EID  
    CFS Checksum Event IDs, 166  
CS\_TBL\_UPDATE\_ERR\_EID  
    CFS Checksum Event IDs, 166  
cs\_tbldefs.h  
    CS\_DEF\_APP\_TABLE\_NAME, 713  
    CS\_DEF\_EEPROM\_TABLE\_NAME, 713  
    CS\_DEF\_MEMORY\_TABLE\_NAME, 713  
    CS\_DEF\_TABLES\_TABLE\_NAME, 713  
    CS\_HandleTableUpdate, 714  
    CS\_ProcessNewAppDefinitionTable, 715  
    CS\_ProcessNewEepromMemoryDefinitionTable, 716  
    CS\_ProcessNewTablesDefinitionTable, 717  
    CS\_RESULTS\_APP\_TABLE\_NAME, 713  
    CS\_RESULTS\_EEPROM\_TABLE\_NAME, 713  
    CS\_RESULTS\_MEMORY\_TABLE\_NAME, 714  
    CS\_RESULTS\_TABLES\_TABLE\_NAME, 714  
    CS\_TableInit, 718  
    CS\_ValidateAppChecksumDefinitionTable, 720  
    CS\_ValidateEepromChecksumDefinitionTable, 721  
    CS\_ValidateMemoryChecksumDefinitionTable, 722  
    CS\_ValidateTablesChecksumDefinitionTable, 723  
CS\_UPDATE\_APP\_ERR\_EID  
    CFS Checksum Event IDs, 166  
CS\_UPDATE\_CDS\_ERR\_EID  
    CFS Checksum Event IDs, 166  
CS\_UPDATE\_EEPROM\_ERR\_EID  
    CFS Checksum Event IDs, 167  
CS\_UPDATE\_MEMORY\_ERR\_EID  
    CFS Checksum Event IDs, 167  
CS\_UPDATE\_TABLES\_ERR\_EID  
    CFS Checksum Event IDs, 167  
CS\_UpdateCDS  
    cs\_app.c, 734  
    cs\_app.h, 747  
cs\_utils.c  
    CS\_AttemptTableReshare, 866  
    CS\_BackgroundApp, 867  
    CS\_BackgroundCfeCore, 868  
    CS\_BackgroundEeprom, 869  
    CS\_BackgroundMemory, 870  
    CS\_BackgroundOS, 871  
    CS\_BackgroundTables, 872  
    CS\_CheckRecomputeOneshot, 873  
    CS\_FindEnabledAppEntry, 873  
    CS\_FindEnabledEepromEntry, 874  
    CS\_FindEnabledMemoryEntry, 875  
    CS\_FindEnabledTablesEntry, 875  
    CS\_GetAppDefTblEntryByName, 876  
    CS\_GetAppResTblEntryByName, 877  
    CS\_GetTableDefTblEntryByName, 877  
    CS\_GetTableResTblEntryByName, 878  
    CS\_GoToNextTable, 879  
    CS\_HandleRoutineTableUpdates, 879  
    CS\_InitializeDefaultTables, 880

CS\_ResetTablesTblResultEntry, 881  
 CS\_VerifyCmdLength, 881  
 CS\_ZeroAppTempValues, 882  
 CS\_ZeroCfeCoreTempValues, 883  
 CS\_ZeroEepromTempValues, 883  
 CS\_ZeroMemoryTempValues, 883  
 CS\_ZeroOSTempValues, 884  
 CS\_ZeroTablesTempValues, 884  
**cs\_utils.h**  
   CS\_AttemptTableReshare, 886  
   CS\_BackgroundApp, 887  
   CS\_BackgroundCfeCore, 888  
   CS\_BackgroundEeprom, 889  
   CS\_BackgroundMemory, 890  
   CS\_BackgroundOS, 891  
   CS\_BackgroundTables, 892  
   CS\_CheckRecomputeOneshot, 893  
   CS\_FindEnabledAppEntry, 893  
   CS\_FindEnabledEepromEntry, 894  
   CS\_FindEnabledMemoryEntry, 895  
   CS\_FindEnabledTablesEntry, 895  
   CS\_GetAppDefTblEntryByName, 896  
   CS\_GetAppResTblEntryByName, 897  
   CS\_GetTableDefTblEntryByName, 897  
   CS\_GetTableResTblEntryByName, 898  
   CS\_GoToNextTable, 899  
   CS\_HandleRoutineTableUpdates, 899  
   CS\_InitializeDefaultTables, 900  
   CS\_ResetTablesTblResultEntry, 901  
   CS\_VerifyCmdLength, 901  
   CS\_ZeroAppTempValues, 902  
   CS\_ZeroCfeCoreTempValues, 903  
   CS\_ZeroEepromTempValues, 903  
   CS\_ZeroMemoryTempValues, 903  
   CS\_ZeroOSTempValues, 904  
   CS\_ZeroTablesTempValues, 904  
**CS\_VAL\_APP\_DEF\_TBL\_DUPL\_ERR\_EID**  
   CFS Checksum Event IDs, 167  
**CS\_VAL\_APP\_DEF\_TBL\_LONG\_NAME\_ERR\_EID**  
   CFS Checksum Event IDs, 168  
**CS\_VAL\_APP\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID**  
   CFS Checksum Event IDs, 168  
**CS\_VAL\_APP\_INF\_EID**  
   CFS Checksum Event IDs, 168  
**CS\_VAL\_APP\_STATE\_ERR\_EID**  
   CFS Checksum Event IDs, 169  
**CS\_VAL\_EEPROM\_INF\_EID**  
   CFS Checksum Event IDs, 169  
**CS\_VAL\_EEPROM\_RANGE\_ERR\_EID**  
   CFS Checksum Event IDs, 169  
**CS\_VAL\_EEPROM\_STATE\_ERR\_EID**  
   CFS Checksum Event IDs, 169  
**CS\_VAL\_MEMORY\_INF\_EID**  
   CFS Checksum Event IDs, 170  
**CS\_VAL\_MEMORY\_RANGE\_ERR\_EID**  
   CFS Checksum Event IDs, 170  
**CS\_VAL\_MEMORY\_STATE\_ERR\_EID**  
   CFS Checksum Event IDs, 170  
**CS\_VAL\_TABLES\_DEF\_TBL\_DUPL\_ERR\_EID**  
   CFS Checksum Event IDs, 170  
**CS\_VAL\_TABLES\_DEF\_TBL\_ZERO\_NAME\_ERR\_EID**  
   CFS Checksum Event IDs, 171  
**CS\_VAL\_TABLES\_INF\_EID**  
   CFS Checksum Event IDs, 171  
**CS\_VAL\_TABLES\_STATE\_ERR\_EID**  
   CFS Checksum Event IDs, 171  
**CS\_ValidateAppChecksumDefinitionTable**  
   cs\_table\_processing.c, 861  
   cs\_tbldefs.h, 720  
**CS\_ValidateEepromChecksumDefinitionTable**  
   cs\_table\_processing.c, 862  
   cs\_tbldefs.h, 721  
**CS\_ValidateMemoryChecksumDefinitionTable**  
   cs\_table\_processing.c, 863  
   cs\_tbldefs.h, 722  
**CS\_ValidateTablesChecksumDefinitionTable**  
   cs\_table\_processing.c, 864  
   cs\_tbldefs.h, 723  
**CS\_VerifyCmdLength**  
   cs\_utils.c, 881  
   cs\_utils.h, 901  
**CS\_WAKEUP\_TIMEOUT**  
   cs\_app.h, 738  
**CS\_ZeroAppTempValues**  
   cs\_utils.c, 882  
   cs\_utils.h, 902  
**CS\_ZeroCfeCoreTempValues**  
   cs\_utils.c, 883  
   cs\_utils.h, 903  
**CS\_ZeroEepromTempValues**  
   cs\_utils.c, 883  
   cs\_utils.h, 903  
**CS\_ZeroMemoryTempValues**  
   cs\_utils.c, 883  
   cs\_utils.h, 903  
**CS\_ZeroOSTempValues**  
   cs\_utils.c, 884  
   cs\_utils.h, 904  
**CS\_ZeroTablesTempValues**  
   cs\_utils.c, 884  
   cs\_utils.h, 904  
**CurrentCSTable**  
   CS\_HkPacket\_Payload\_t, 670  
**CurrentEntryInTable**  
   CS\_HkPacket\_Payload\_t, 670  
**CurrentLatch**  
   CFE\_TIME\_DiagnosticTlm\_Payload, 637  
**CurrentMET**

CFE\_TIME\_DiagnosticTlm\_Payload, 637  
CurrentQueueDepth  
    CFE\_SB\_PipeDepthStats, 595  
    CFE\_SB\_PipeInfoEntry, 597  
CurrentTAI  
    CFE\_TIME\_DiagnosticTlm\_Payload, 637  
CurrentUTC  
    CFE\_TIME\_DiagnosticTlm\_Payload, 637  
  
data\_address  
    OS\_module\_address\_t, 689  
data\_size  
    OS\_module\_address\_t, 689  
DataAddress  
    CFE\_ES\_AppInfo, 528  
DataFileName  
    CFE\_ES\_StopPerfCmd\_Payload, 561  
DataSize  
    CFE\_ES\_AppInfo, 528  
DataStoreHandle  
    CS\_AppData\_t, 657  
DataStoreStatus  
    CFE\_TIME\_DiagnosticTlm\_Payload, 638  
DefAppTableHandle  
    CS\_AppData\_t, 657  
DefAppTblPtr  
    CS\_AppData\_t, 657  
default\_cfe\_core\_api\_base\_msgids.h  
    CFE\_PLATFORM\_CMD\_MID\_BASE, 971  
    CFE\_PLATFORM\_CMD\_MID\_BASE\_GLOB, 971  
    CFE\_PLATFORM\_TLM\_MID\_BASE, 972  
default\_cfe\_core\_api\_interface\_cfg.h  
    CFE\_MISSION\_MAX\_API\_LEN, 972  
    CFE\_MISSION\_MAX\_FILE\_LEN, 972  
    CFE\_MISSION\_MAX\_NUM\_FILES, 973  
    CFE\_MISSION\_MAX\_PATH\_LEN, 973  
default\_cfe\_es\_extern\_typedefs.h  
    CFE\_ES\_AppId\_t, 1034  
    CFE\_ES\_AppInfo\_t, 1034  
    CFE\_ES\_AppState, 1038  
    CFE\_ES\_AppState\_EARLY\_INIT, 1038  
    CFE\_ES\_AppState\_Enum\_t, 1034  
    CFE\_ES\_AppState\_LATE\_INIT, 1038  
    CFE\_ES\_AppState\_MAX, 1038  
    CFE\_ES\_AppState\_RUNNING, 1038  
    CFE\_ES\_AppState\_STOPPED, 1038  
    CFE\_ES\_AppState\_UNDEFINED, 1038  
    CFE\_ES\_AppState\_WAITING, 1038  
    CFE\_ES\_AppType, 1038  
    CFE\_ES\_AppType\_CORE, 1038  
    CFE\_ES\_AppType\_Enum\_t, 1035  
    CFE\_ES\_AppType\_EXTERNAL, 1038  
    CFE\_ES\_AppType\_LIBRARY, 1038  
    CFE\_ES\_BlockStats\_t, 1035  
  
CFE\_ES\_CDSHandle\_t, 1035  
CFE\_ES\_CDSRegDumpRec\_t, 1035  
CFE\_ES\_CounterId\_t, 1035  
CFE\_ES\_ExceptionAction, 1038  
CFE\_ES\_ExceptionAction\_Enum\_t, 1035  
CFE\_ES\_ExceptionAction\_PROC\_RESTART, 1038  
CFE\_ES\_ExceptionAction\_RESTART\_APP, 1038  
CFE\_ES\_LibId\_t, 1035  
CFE\_ES\_LogEntryType, 1038  
CFE\_ES\_LogEntryType\_APPLICATION, 1038  
CFE\_ES\_LogEntryType\_CORE, 1038  
CFE\_ES\_LogEntryType\_Enum\_t, 1036  
CFE\_ES\_LogMode, 1039  
CFE\_ES\_LogMode\_DISCARD, 1039  
CFE\_ES\_LogMode\_Enum\_t, 1036  
CFE\_ES\_LogMode\_OVERWRITE, 1039  
CFE\_ES\_MEMADDRESS\_C, 1034  
CFE\_ES\_MemAddress\_t, 1036  
CFE\_ES\_MEMADDRESS\_TO\_PTR, 1034  
CFE\_ES\_MemHandle\_t, 1036  
CFE\_ES\_MEMOFFSET\_C, 1034  
CFE\_ES\_MemOffset\_t, 1036  
CFE\_ES\_MEMOFFSET\_TO\_SIZE, 1034  
CFE\_ES\_MemPoolStats\_t, 1036  
CFE\_ES\_RunStatus, 1039  
CFE\_ES\_RunStatus\_APP\_ERROR, 1039  
CFE\_ES\_RunStatus\_APP\_EXIT, 1039  
CFE\_ES\_RunStatus\_APP\_RUN, 1039  
CFE\_ES\_RunStatus\_CORE\_APP\_INIT\_ERROR, 1039  
CFE\_ES\_RunStatus\_CORE\_APP\_RUNTIME\_ERROR, 1039  
CFE\_ES\_RunStatus\_Enum\_t, 1037  
CFE\_ES\_RunStatus\_MAX, 1039  
CFE\_ES\_RunStatus\_SYS\_DELETE, 1039  
CFE\_ES\_RunStatus\_SYS\_EXCEPTION, 1039  
CFE\_ES\_RunStatus\_SYS\_RELOAD, 1039  
CFE\_ES\_RunStatus\_SYS\_RESTART, 1039  
CFE\_ES\_RunStatus\_UNDEFINED, 1039  
CFE\_ES\_SystemState, 1039  
CFE\_ES\_SystemState\_APPS\_INIT, 1039  
CFE\_ES\_SystemState\_CORE\_READY, 1039  
CFE\_ES\_SystemState\_CORE\_STARTUP, 1039  
CFE\_ES\_SystemState\_EARLY\_INIT, 1039  
CFE\_ES\_SystemState\_Enum\_t, 1037  
CFE\_ES\_SystemState\_MAX, 1039  
CFE\_ES\_SystemState\_OPERATIONAL, 1039  
CFE\_ES\_SystemState\_SHUTDOWN, 1039  
CFE\_ES\_SystemState\_UNDEFINED, 1039  
CFE\_ES\_TaskId\_t, 1037  
CFE\_ES\_TaskInfo\_t, 1037  
CFE\_ES\_TaskPriority\_Atom\_t, 1037  
default\_cfe\_es\_fncodes.h  
    CFE\_ES\_CLEAR\_ER\_LOG\_CC, 1040

CFE\_ES\_CLEAR\_SYSLOG\_CC, 1041  
CFE\_ES\_DELETE\_CDS\_CC, 1042  
CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC, 1042  
CFE\_ES\_NOOP\_CC, 1043  
CFE\_ES\_OVER\_WRITE\_SYSLOG\_CC, 1044  
CFE\_ES\_QUERY\_ALL\_CC, 1045  
CFE\_ES\_QUERY\_ALL\_TASKS\_CC, 1046  
CFE\_ES\_QUERY\_ONE\_CC, 1046  
CFE\_ES\_RELOAD\_APP\_CC, 1047  
CFE\_ES\_RESET\_COUNTERS\_CC, 1048  
CFE\_ES\_RESET\_PR\_COUNT\_CC, 1049  
CFE\_ES\_RESTART\_APP\_CC, 1049  
CFE\_ES\_RESTART\_CC, 1050  
CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC, 1051  
CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC, 1052  
CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC, 1053  
CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC, 1054  
CFE\_ES\_START\_APP\_CC, 1054  
CFE\_ES\_START\_PERF\_DATA\_CC, 1055  
CFE\_ES\_STOP\_APP\_CC, 1056  
CFE\_ES\_STOP\_PERF\_DATA\_CC, 1057  
CFE\_ES\_WRITE\_ER\_LOG\_CC, 1058  
CFE\_ES\_WRITE\_SYSLOG\_CC, 1059

default\_cfe\_es\_interface\_cfg.h

CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN, 1061  
CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH, 1061  
CFE\_MISSION\_ES\_CRC\_16, 1061  
CFE\_MISSION\_ES\_CRC\_32, 1061  
CFE\_MISSION\_ES\_CRC\_8, 1062  
CFE\_MISSION\_ES\_DEFAULT\_CRC, 1062  
CFE\_MISSION\_ES\_MAX\_APPLICATIONS, 1062  
CFE\_MISSION\_ES\_PERF\_MAX\_IDS, 1062  
CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS, 1062

default\_cfe\_es\_internal\_cfg.h

CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT, 1065  
CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE, 1065  
CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE, 1066  
CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES, 1066  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01, 1066  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02, 1066  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03, 1066  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04, 1066  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06, 1067

CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14, 1067  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15, 1068  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16, 1068  
CFE\_PLATFORM\_ES\_CDS\_SIZE, 1068  
CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE, 1068  
CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE, 1068  
CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE, 1069  
CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME, 1069  
CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE, 1069  
CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE, 1070  
CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, 1070  
CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE, 1070  
CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE, 1071  
CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES, 1071  
CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE, 1071  
CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS, 1072  
CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE, 1072  
CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS, 1072  
CFE\_PLATFORM\_ES\_MAX\_LIBRARIES, 1072  
CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS, 1073  
CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS, 1073  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01, 1073

- 1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10,  
1074  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11,  
1075  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12,  
1075  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13,  
1075  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14,  
1075  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15,  
1075  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16,  
1075  
CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN,  
1075  
CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING,  
1075  
CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE,  
1076  
CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE, 1076  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY,  
1076  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY,  
1076  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE,  
1077  
CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE,  
1077  
CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS,  
1077  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL, 1078  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT,  
1078  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE,  
1078  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL,  
1078  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT,
- 1079  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE,  
1079  
CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS,  
1079  
CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING,  
1079  
CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS,  
1080  
CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED,  
1080  
CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE,  
1080  
CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY,  
1081  
CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE,  
1081  
CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC,  
1081  
CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC,  
1082  
CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE, 1082  
CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE,  
1082  
CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE,  
1083  
default\_cfe\_es\_msgids.h  
CFE\_ES\_APP\_TLM\_MID, 1084  
CFE\_ES\_CMD\_MID, 1084  
CFE\_ES\_HK\_TLM\_MID, 1085  
CFE\_ES\_MEMSTATS\_TLM\_MID, 1085  
CFE\_ES\_SEND\_HK\_MID, 1085  
default\_cfe\_es\_msgstruct.h  
CFE\_ES\_AppNameCmd\_Payload\_t, 1088  
CFE\_ES\_AppNameCmd\_t, 1088  
CFE\_ES\_AppReloadCmd\_Payload\_t, 1088  
CFE\_ES\_ClearERLogCmd\_t, 1089  
CFE\_ES\_ClearSysLogCmd\_t, 1089  
CFE\_ES\_DeleteCDSCmd\_Payload\_t, 1089  
CFE\_ES\_DeleteCDSCmd\_t, 1089  
CFE\_ES\_DumpCDSRegistryCmd\_Payload\_t, 1089  
CFE\_ES\_DumpCDSRegistryCmd\_t, 1089  
CFE\_ES\_FileNameCmd\_Payload\_t, 1089  
CFE\_ES\_FileNameCmd\_t, 1089  
CFE\_ES\_HousekeepingTlm\_Payload\_t, 1089  
CFE\_ES\_HousekeepingTlm\_t, 1089  
CFE\_ES\_MemStatsTlm\_t, 1089  
CFE\_ES\_NoArgsCmd\_t, 1089  
CFE\_ES\_NoopCmd\_t, 1090  
CFE\_ES\_OneAppTlm\_Payload\_t, 1090  
CFE\_ES\_OneAppTlm\_t, 1090  
CFE\_ES\_OverWriteSysLogCmd\_Payload\_t, 1090  
CFE\_ES\_OverWriteSysLogCmd\_t, 1090  
CFE\_ES\_PoolStatsTlm\_Payload\_t, 1090

CFE\_ES\_QueryAllCmd\_t, 1090  
 CFE\_ES\_QueryAllTasksCmd\_t, 1090  
 CFE\_ES\_QueryOneCmd\_t, 1090  
 CFE\_ES\_ReloadAppCmd\_t, 1090  
 CFE\_ES\_ResetCountersCmd\_t, 1090  
 CFE\_ES\_ResetPRCountCmd\_t, 1091  
 CFE\_ES.RestartAppCmd\_t, 1091  
 CFE\_ES.RestartCmd\_Payload\_t, 1091  
 CFE\_ES.RestartCmd\_t, 1091  
 CFE\_ES.SendHkCmd\_t, 1091  
 CFE\_ES.SendMemPoolStatsCmd\_Payload\_t, 1091  
 CFE\_ES.SendMemPoolStatsCmd\_t, 1091  
 CFE\_ES\_SetMaxPRCountCmd\_Payload\_t, 1091  
 CFE\_ES\_SetMaxPRCountCmd\_t, 1091  
 CFE\_ES\_SetPerfFilterMaskCmd\_Payload\_t, 1091  
 CFE\_ES\_SetPerfFilterMaskCmd\_t, 1091  
 CFE\_ES\_SetPerfTriggerMaskCmd\_t, 1091  
 CFE\_ES\_SetPerfTrigMaskCmd\_Payload\_t, 1092  
 CFE\_ES\_StartAppCmd\_Payload\_t, 1092  
 CFE\_ES\_StartAppCmd\_t, 1092  
 CFE\_ES\_StartPerfCmd\_Payload\_t, 1092  
 CFE\_ES\_StartPerfDataCmd\_t, 1092  
 CFE\_ES\_StopAppCmd\_t, 1092  
 CFE\_ES\_StopPerfCmd\_Payload\_t, 1092  
 CFE\_ES\_StopPerfDataCmd\_t, 1092  
 CFE\_ES\_WriteERLogCmd\_t, 1092  
 CFE\_ES\_WriteSysLogCmd\_t, 1092

default\_cfe\_es\_topicids.h  
 CFE\_MISSION\_ES\_APP\_TLM\_MSG, 1093  
 CFE\_MISSION\_ES\_CMD\_MSG, 1093  
 CFE\_MISSION\_ES\_HK\_TLM\_MSG, 1093  
 CFE\_MISSION\_ES\_MEMSTATS\_TLM\_MSG, 1094  
 CFE\_MISSION\_ES\_SEND\_HK\_MSG, 1094

default\_cfe\_evs\_extern\_typedefs.h  
 CFE\_EVS\_EventFilter, 1120  
 CFE\_EVS\_EventFilter\_BINARY, 1120  
 CFE\_EVS\_EventFilter\_Enum\_t, 1119  
 CFE\_EVS\_EventOutput, 1121  
 CFE\_EVS\_EventOutput\_Enum\_t, 1120  
 CFE\_EVS\_EventOutput\_PORT1, 1121  
 CFE\_EVS\_EventOutput\_PORT2, 1121  
 CFE\_EVS\_EventOutput\_PORT3, 1121  
 CFE\_EVS\_EventOutput\_PORT4, 1121  
 CFE\_EVS\_EventType, 1121  
 CFE\_EVS\_EventType\_CRITICAL, 1121  
 CFE\_EVS\_EventType\_DEBUG, 1121  
 CFE\_EVS\_EventType\_Enum\_t, 1120  
 CFE\_EVS\_EventType\_ERROR, 1121  
 CFE\_EVS\_EventType\_INFORMATION, 1121  
 CFE\_EVS\_LogMode, 1121  
 CFE\_EVS\_LogMode\_DISCARD, 1121  
 CFE\_EVS\_LogMode\_Enum\_t, 1120  
 CFE\_EVS\_LogMode\_OVERWRITE, 1121  
 CFE\_EVS\_MsgFormat, 1121

CFE\_EVS\_MsgFormat\_Enum\_t, 1120  
 CFE\_EVS\_MsgFormat\_LONG, 1121  
 CFE\_EVS\_MsgFormat\_SHORT, 1121

default\_cfe\_evs\_fcncodes.h  
 CFE\_EVS\_ADD\_EVENT\_FILTER\_CC, 1122  
 CFE\_EVS\_CLEAR\_LOG\_CC, 1123  
 CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC, 1124  
 CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC, 1124  
 CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC, 1125  
 CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC, 1126  
 CFE\_EVS\_DISABLE\_PORTS\_CC, 1127  
 CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC, 1128  
 CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC, 1129  
 CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC, 1130  
 CFE\_EVS\_ENABLE\_PORTS\_CC, 1131  
 CFE\_EVS\_NOOP\_CC, 1132  
 CFE\_EVS\_RESET\_ALL\_FILTERS\_CC, 1132  
 CFE\_EVS\_RESET\_APP\_COUNTER\_CC, 1133  
 CFE\_EVS\_RESET\_COUNTERS\_CC, 1134  
 CFE\_EVS\_RESET\_FILTER\_CC, 1135  
 CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC, 1135  
 CFE\_EVS\_SET\_FILTER\_CC, 1136  
 CFE\_EVS\_SET\_LOG\_MODE\_CC, 1137  
 CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC, 1138  
 CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC, 1139

default\_cfe\_evs\_interface\_cfg.h  
 CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH, 1140

default\_cfe\_evs\_internal\_cfg.h  
 CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC, 1141  
 CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE, 1141  
 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE, 1142  
 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE, 1142  
 CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE, 1142  
 CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG, 1142  
 CFE\_PLATFORM\_EVS\_LOG\_MAX, 1143  
 CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST, 1143  
 CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS, 1143  
 CFE\_PLATFORM\_EVS\_PORT\_DEFAULT, 1144  
 CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY, 1144  
 CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE, 1144

default\_cfe\_evs\_msgdefs.h  
 CFE\_EVS\_CRITICAL\_BIT, 1146

CFE\_EVS\_DEBUG\_BIT, 1146  
CFE\_EVS\_ERROR\_BIT, 1146  
CFE\_EVS\_INFORMATION\_BIT, 1146  
CFE\_EVS\_PORT1\_BIT, 1146  
CFE\_EVS\_PORT2\_BIT, 1146  
CFE\_EVS\_PORT3\_BIT, 1146  
CFE\_EVS\_PORT4\_BIT, 1146  
default\_cfe\_evs\_msgids.h  
CFE\_EVS\_CMD\_MID, 1147  
CFE\_EVS\_HK\_TLM\_MID, 1147  
CFE\_EVS\_LONG\_EVENT\_MSG\_MID, 1147  
CFE\_EVS\_SEND\_HK\_MID, 1147  
CFE\_EVS\_SHORT\_EVENT\_MSG\_MID, 1147  
default\_cfe\_evs\_msgstruct.h  
CFE\_EVS\_AddEventFilterCmd\_t, 1150  
CFE\_EVS\_AppDataCmd\_Payload\_t, 1150  
CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t, 1150  
CFE\_EVS\_AppNameBitMaskCmd\_t, 1150  
CFE\_EVS\_AppNameCmd\_Payload\_t, 1150  
CFE\_EVS\_AppNameCmd\_t, 1150  
CFE\_EVS\_AppNameEventIDCmd\_Payload\_t, 1150  
CFE\_EVS\_AppNameEventIDCmd\_t, 1150  
CFE\_EVS\_AppNameEventIDMaskCmd\_Payload\_t, 1150  
CFE\_EVS\_AppNameEventIDMaskCmd\_t, 1151  
CFE\_EVS\_AppTlmData\_t, 1151  
CFE\_EVS\_BitMaskCmd\_Payload\_t, 1151  
CFE\_EVS\_BitMaskCmd\_t, 1151  
CFE\_EVS\_ClearLogCmd\_t, 1151  
CFE\_EVS\_DeleteEventFilterCmd\_t, 1151  
CFE\_EVS\_DisableAppEventsCmd\_t, 1151  
CFE\_EVS\_DisableAppEventTypeCmd\_t, 1151  
CFE\_EVS\_DisableEventTypeCmd\_t, 1151  
CFE\_EVS\_DisablePortsCmd\_t, 1151  
CFE\_EVS\_EnableAppEventsCmd\_t, 1151  
CFE\_EVS\_EnableAppEventTypeCmd\_t, 1151  
CFE\_EVS\_EnableEventCmd\_t, 1152  
CFE\_EVS\_EnablePortsCmd\_t, 1152  
CFE\_EVS\_HousekeepingTlm\_Payload\_t, 1152  
CFE\_EVS\_HousekeepingTlm\_t, 1152  
CFE\_EVS\_LogFileCmd\_Payload\_t, 1152  
CFE\_EVS\_LongEventTlm\_Payload\_t, 1152  
CFE\_EVS\_LongEventTlm\_t, 1152  
CFE\_EVS\_NoArgsCmd\_t, 1152  
CFE\_EVS\_NoopCmd\_t, 1152  
CFE\_EVS\_PacketID\_t, 1152  
CFE\_EVS\_ResetAllFiltersCmd\_t, 1152  
CFE\_EVS\_ResetAppCounterCmd\_t, 1152  
CFE\_EVS\_ResetCountersCmd\_t, 1152  
CFE\_EVS\_ResetFilterCmd\_t, 1153  
CFE\_EVS\_SendHkCmd\_t, 1153  
CFE\_EVS\_SetEventFormatMode\_Payload\_t, 1153  
CFE\_EVS\_SetEventFormatModeCmd\_t, 1153  
CFE\_EVS\_SetFilterCmd\_t, 1153  
CFE\_EVS\_SetLogMode\_Payload\_t, 1153  
CFE\_EVS\_SetLogModeCmd\_t, 1153  
CFE\_EVS\_ShortEventTlm\_Payload\_t, 1153  
CFE\_EVS\_ShortEventTlm\_t, 1153  
CFE\_EVS\_WriteAppDataFileCmd\_t, 1153  
CFE\_EVS\_WriteLogDataFileCmd\_t, 1153  
default\_cfe\_evs\_topicids.h  
CFE\_MISSION\_EVS\_CMD\_MSG, 1154  
CFE\_MISSION\_EVS\_HK\_TLM\_MSG, 1154  
CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_MSG, 1155  
CFE\_MISSION\_EVS\_SEND\_HK\_MSG, 1155  
CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_MSG, 1155  
default\_cfe\_fs\_filedef.h  
CFE\_FS\_Header\_t, 1168  
CFE\_FS\_SubType, 1168  
CFE\_FS\_SubType\_Enum\_t, 1168  
CFE\_FS\_SubType\_ES\_CDS\_REG, 1169  
CFE\_FS\_SubType\_ES\_ERLOG, 1168  
CFE\_FS\_SubType\_ES\_PERFDATA, 1169  
CFE\_FS\_SubType\_ES\_QUERYALL, 1168  
CFE\_FS\_SubType\_ES\_QUERYALLTASKS, 1169  
CFE\_FS\_SubType\_ES\_SYSLOG, 1168  
CFE\_FS\_SubType\_EVS\_APPDATA, 1169  
CFE\_FS\_SubType\_EVS\_EVENTLOG, 1169  
CFE\_FS\_SubType\_SB\_MAPDATA, 1169  
CFE\_FS\_SubType\_SB\_PIPEDATA, 1169  
CFE\_FS\_SubType\_SB\_ROUTEDATA, 1169  
CFE\_FS\_SubType\_TBL\_IMG, 1169  
CFE\_FS\_SubType\_TBL\_REG, 1169  
default\_cfe\_fs\_interface\_cfg.h  
CFE\_FS\_FILE\_CONTENT\_ID, 1169  
CFE\_FS\_HDR\_DESC\_MAX\_LEN, 1170  
default\_cfe\_sb\_extern\_typedefs.h  
CFE\_SB\_MsgId\_Atom\_t, 1173  
CFE\_SB\_Pipeld\_t, 1173  
CFE\_SB\_QosPriority, 1174  
CFE\_SB\_QosPriority\_Enum\_t, 1173  
CFE\_SB\_QosPriority\_HIGH, 1174  
CFE\_SB\_QosPriority\_LOW, 1174  
CFE\_SB\_QosReliability, 1174  
CFE\_SB\_QosReliability\_Enum\_t, 1173  
CFE\_SB\_QosReliability\_HIGH, 1174  
CFE\_SB\_QosReliability\_LOW, 1174  
CFE\_SB\_Routeld\_Atom\_t, 1174  
CFE\_SB\_SUB\_ENTRIES\_PER\_PKT, 1173  
default\_cfe\_sb\_fcncodes.h  
CFE\_SB\_DISABLE\_ROUTE\_CC, 1175  
CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC, 1175  
CFE\_SB\_ENABLE\_ROUTE\_CC, 1176  
CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC, 1177  
CFE\_SB\_NOOP\_CC, 1178  
CFE\_SB\_RESET\_COUNTERS\_CC, 1178

CFE\_SB\_SEND\_PREV\_SUBS\_CC, [1179](#)  
 CFE\_SB\_SEND\_SB\_STATS\_CC, [1180](#)  
 CFE\_SB\_WRITE\_MAP\_INFO\_CC, [1181](#)  
 CFE\_SB\_WRITE\_PIPE\_INFO\_CC, [1182](#)  
 CFE\_SB\_WRITE\_ROUTING\_INFO\_CC, [1183](#)  
 default\_cfe\_sb\_interface\_cfg.h  
     CFE\_MISSION\_SB\_MAX\_PIPES, [1184](#)  
     CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE, [1184](#)  
 default\_cfe\_sb\_internal\_cfg.h  
     CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES,  
         [1186](#)  
     CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME,  
         [1186](#)  
     CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT, [1187](#)  
     CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME,  
         [1187](#)  
     CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME,  
         [1187](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK1, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK2, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK3, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK4, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK5, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK6, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK7, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTER\_MASK8, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT1, [1188](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT2, [1189](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT3, [1189](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT4, [1189](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT5, [1189](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT6, [1189](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT7, [1189](#)  
     CFE\_PLATFORM\_SB\_FILTERED\_EVENT8, [1189](#)  
     CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID,  
         [1189](#)  
     CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE, [1190](#)  
     CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT,  
         [1190](#)  
     CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS, [1190](#)  
     CFE\_PLATFORM\_SB\_MAX\_PIPES, [1191](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01,  
         [1191](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02,  
         [1191](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03,  
         [1191](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04,  
         [1191](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05,  
         [1191](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16,  
         [1192](#)  
     CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY,  
         [1193](#)  
     CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE,  
         [1193](#)  
 default\_cfe\_sb\_msgids.h  
     CFE\_SB\_ALLSUBS\_TLM\_MID, [1195](#)  
     CFE\_SB\_CMD\_MID, [1195](#)  
     CFE\_SB\_HK\_TLM\_MID, [1195](#)  
     CFE\_SB\_ONESUB\_TLM\_MID, [1195](#)  
     CFE\_SB\_SEND\_HK\_MID, [1195](#)  
     CFE\_SB\_STATS\_TLM\_MID, [1195](#)  
     CFE\_SB\_SUB\_RPT\_CTRL\_MID, [1195](#)  
 default\_cfe\_sb\_msgstruct.h  
     CFE\_SB\_AllSubscriptionsTlm\_Payload\_t, [1197](#)  
     CFE\_SB\_AllSubscriptionsTlm\_t, [1197](#)  
     CFE\_SB\_DisableRouteCmd\_t, [1197](#)  
     CFE\_SB\_DisableSubReportingCmd\_t, [1197](#)  
     CFE\_SB\_EnableRouteCmd\_t, [1197](#)  
     CFE\_SB\_EnableSubReportingCmd\_t, [1197](#)  
     CFE\_SB\_HousekeepingTlm\_Payload\_t, [1197](#)  
     CFE\_SB\_HousekeepingTlm\_t, [1198](#)  
     CFE\_SB\_MsgMapFileEntry\_t, [1198](#)  
     CFE\_SB\_NoopCmd\_t, [1198](#)  
     CFE\_SB\_PipeDepthStats\_t, [1198](#)  
     CFE\_SB\_PipeInfoEntry\_t, [1198](#)  
     CFE\_SB\_ResetCountersCmd\_t, [1198](#)  
     CFE\_SB\_RouteCmd\_Payload\_t, [1198](#)  
     CFE\_SB\_RouteCmd\_t, [1198](#)  
     CFE\_SB\_RoutingFileEntry\_t, [1198](#)  
     CFE\_SB\_SendHkCmd\_t, [1198](#)  
     CFE\_SB\_SendPrevSubsCmd\_t, [1198](#)  
     CFE\_SB\_SendSbStatsCmd\_t, [1199](#)  
     CFE\_SB\_SingleSubscriptionTlm\_Payload\_t, [1199](#)  
     CFE\_SB\_SingleSubscriptionTlm\_t, [1199](#)  
     CFE\_SB\_StatsTlm\_Payload\_t, [1199](#)

CFE\_SB\_StatsTlm\_t, 1199  
CFE\_SB\_SubEntries\_t, 1199  
CFE\_SB\_WriteFileInfoCmd\_Payload\_t, 1199  
CFE\_SB\_WriteFileInfoCmd\_t, 1199  
CFE\_SB\_WriteMapInfoCmd\_t, 1199  
CFE\_SB\_WritePipeInfoCmd\_t, 1199  
CFE\_SB\_WriteRoutingInfoCmd\_t, 1200  
default\_cfe\_sb\_topicids.h  
CFE\_MISSION\_SB\_ALLSUBS\_TLM\_MSG, 1200  
CFE\_MISSION\_SB\_CMD\_MSG, 1200  
CFE\_MISSION\_SB\_HK\_TLM\_MSG, 1201  
CFE\_MISSION\_SB\_ONESUB\_TLM\_MSG, 1201  
CFE\_MISSION\_SB\_SEND\_HK\_MSG, 1201  
CFE\_MISSION\_SB\_STATS\_TLM\_MSG, 1201  
CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_MSG, 1201  
default\_cfe\_tbl\_extern\_typedefs.h  
CFE\_TBL\_BufferSelect, 1221  
CFE\_TBL\_BufferSelect\_ACTIVE, 1221  
CFE\_TBL\_BufferSelect\_Enum\_t, 1221  
CFE\_TBL\_BufferSelect\_INACTIVE, 1221  
CFE\_TBL\_File\_Hdr\_t, 1221  
default\_cfe\_tbl\_fcncodes.h  
CFE\_TBL\_ABORT\_LOAD\_CC, 1222  
CFE\_TBL\_ACTIVATE\_CC, 1223  
CFE\_TBL\_DELETE\_CDS\_CC, 1224  
CFE\_TBL\_DUMP\_CC, 1224  
CFE\_TBL\_DUMP\_REGISTRY\_CC, 1225  
CFE\_TBL\_LOAD\_CC, 1226  
CFE\_TBL\_NOOP\_CC, 1227  
CFE\_TBL\_RESET\_COUNTERS\_CC, 1228  
CFE\_TBL\_SEND\_REGISTRY\_CC, 1229  
CFE\_TBL\_VALIDATE\_CC, 1230  
default\_cfe\_tbl\_interface\_cfg.h  
CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN, 1231  
CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, 1232  
default\_cfe\_tbl\_internal\_cfg.h  
CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES, 1233  
CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE, 1233  
CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES, 1233  
CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE, 1234  
CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES, 1234  
CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES, 1234  
CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS, 1234  
CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS, 1235  
CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE, 1235  
CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY, 1235  
CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE, 1236  
CFE\_PLATFORM\_TBL\_U32FROM4CHARS, 1236  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_1, 1236  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_2, 1236  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_3, 1237  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_4, 1237  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT, 1237  
CFE\_PLATFORM\_TBL\_VALID\_SCID\_1, 1237  
CFE\_PLATFORM\_TBL\_VALID\_SCID\_2, 1237  
CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT, 1237  
default\_cfe\_tbl\_msgids.h  
CFE\_TBL\_CMD\_MID, 1239  
CFE\_TBL\_HK\_TLM\_MID, 1239  
CFE\_TBL\_REG\_TLM\_MID, 1239  
CFE\_TBL\_SEND\_HK\_MID, 1239  
default\_cfe\_tbl\_msgstruct.h  
CFE\_TBL\_AbortLoadCmd\_Payload\_t, 1242  
CFE\_TBL\_AbortLoadCmd\_t, 1242  
CFE\_TBL\_ActivateCmd\_Payload\_t, 1242  
CFE\_TBL\_ActivateCmd\_t, 1242  
CFE\_TBL\_DeICDSCmd\_Payload\_t, 1242  
CFE\_TBL\_DeleteCDSCmd\_t, 1242  
CFE\_TBL\_DumpCmd\_Payload\_t, 1242  
CFE\_TBL\_DumpCmd\_t, 1242  
CFE\_TBL\_DumpRegistryCmd\_Payload\_t, 1242  
CFE\_TBL\_DumpRegistryCmd\_t, 1242  
CFE\_TBL\_HousekeepingTlm\_Payload\_t, 1242  
CFE\_TBL\_HousekeepingTlm\_t, 1243  
CFE\_TBL\_LoadCmd\_Payload\_t, 1243  
CFE\_TBL\_LoadCmd\_t, 1243  
CFE\_TBL\_NoArgsCmd\_t, 1243  
CFE\_TBL\_NoopCmd\_t, 1243  
CFE\_TBL\_NotifyCmd\_Payload\_t, 1243  
CFE\_TBL\_NotifyCmd\_t, 1243  
CFE\_TBL\_ResetCountersCmd\_t, 1243  
CFE\_TBL\_SendHkCmd\_t, 1243  
CFE\_TBL\_SendRegistryCmd\_Payload\_t, 1243  
CFE\_TBL\_SendRegistryCmd\_t, 1244  
CFE\_TBL\_TableRegistryTlm\_t, 1244  
CFE\_TBL\_TblRegPacket\_Payload\_t, 1244  
CFE\_TBL\_ValidateCmd\_Payload\_t, 1244  
CFE\_TBL\_ValidateCmd\_t, 1244  
default\_cfe\_tbl\_topicids.h  
CFE\_MISSION\_TBL\_CMD\_MSG, 1245  
CFE\_MISSION\_TBL\_HK\_TLM\_MSG, 1245  
CFE\_MISSION\_TBL\_REG\_TLM\_MSG, 1245  
CFE\_MISSION\_TBL\_SEND\_HK\_MSG, 1245  
default\_cfe\_time\_extern\_typedefs.h  
CFE\_TIME\_AdjustDirection, 1268  
CFE\_TIME\_AdjustDirection\_ADD, 1268  
CFE\_TIME\_AdjustDirection\_Enum\_t, 1267

CFE\_TIME\_AdjustDirection\_SUBTRACT, 1268  
 CFE\_TIME\_ClockState, 1268  
 CFE\_TIME\_ClockState\_Enum\_t, 1267  
 CFE\_TIME\_ClockState\_FLYWHEEL, 1269  
 CFE\_TIME\_ClockState\_INVALID, 1269  
 CFE\_TIME\_ClockState\_VALID, 1269  
 CFE\_TIME\_FlagBit, 1269  
 CFE\_TIME\_FlagBit\_ADD1HZ, 1269  
 CFE\_TIME\_FlagBit\_ADDADJ, 1269  
 CFE\_TIME\_FlagBit\_ADDTCL, 1269  
 CFE\_TIME\_FlagBit\_CLKSET, 1269  
 CFE\_TIME\_FlagBit\_CMDFLY, 1269  
 CFE\_TIME\_FlagBit\_Enum\_t, 1267  
 CFE\_TIME\_FlagBit\_FLYING, 1269  
 CFE\_TIME\_FlagBit\_GDTONE, 1269  
 CFE\_TIME\_FlagBit\_SERVER, 1269  
 CFE\_TIME\_FlagBit\_SIGPRI, 1269  
 CFE\_TIME\_FlagBit\_SRCINT, 1269  
 CFE\_TIME\_FlagBit\_SRVFLY, 1269  
 CFE\_TIME\_FlywheelState, 1269  
 CFE\_TIME\_FlywheelState\_Enum\_t, 1267  
 CFE\_TIME\_FlywheelState\_IS\_FLY, 1269  
 CFE\_TIME\_FlywheelState\_NO\_FLY, 1269  
 CFE\_TIME\_SetState, 1269  
 CFE\_TIME\_SetState\_Enum\_t, 1267  
 CFE\_TIME\_SetState\_NOT\_SET, 1270  
 CFE\_TIME\_SetState\_WAS\_SET, 1270  
 CFE\_TIME\_SourceSelect, 1270  
 CFE\_TIME\_SourceSelect\_Enum\_t, 1268  
 CFE\_TIME\_SourceSelect\_EXTERNAL, 1270  
 CFE\_TIME\_SourceSelect\_INTERNAL, 1270  
 CFE\_TIME\_SysTime\_t, 1268  
 CFE\_TIME\_ToneSignalSelect, 1270  
 CFE\_TIME\_ToneSignalSelect\_Enum\_t, 1268  
 CFE\_TIME\_ToneSignalSelect\_PRIMARY, 1270  
 CFE\_TIME\_ToneSignalSelect\_REDUNDANT, 1270

default\_cfe\_time\_fcncodes.h  
 CFE\_TIME\_ADD\_1HZ\_ADJUSTMENT\_CC, 1271  
 CFE\_TIME\_ADD\_ADJUST\_CC, 1272  
 CFE\_TIME\_ADD\_DELAY\_CC, 1272  
 CFE\_TIME\_NOOP\_CC, 1273  
 CFE\_TIME\_RESET\_COUNTERS\_CC, 1274  
 CFE\_TIME\_SEND\_DIAGNOSTIC\_TLM\_CC, 1275  
 CFE\_TIME\_SET\_LEAP\_SECONDS\_CC, 1276  
 CFE\_TIME\_SET\_MET\_CC, 1277  
 CFE\_TIME\_SET\_SIGNAL\_CC, 1278  
 CFE\_TIME\_SET\_SOURCE\_CC, 1278  
 CFE\_TIME\_SET\_STATE\_CC, 1279  
 CFE\_TIME\_SET\_STCF\_CC, 1281  
 CFE\_TIME\_SET\_TIME\_CC, 1281  
 CFE\_TIME\_SUB\_1HZ\_ADJUSTMENT\_CC, 1282  
 CFE\_TIME\_SUB\_ADJUST\_CC, 1283  
 CFE\_TIME\_SUB\_DELAY\_CC, 1284

default\_cfe\_time\_interface\_cfg.h  
 CFE\_MISSION\_TIME\_AT\_TONE\_WAS, 1286  
 CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE, 1286  
 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI, 1287  
 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC, 1287  
 CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE, 1287  
 CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS, 1287  
 CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS, 1287  
 CFE\_MISSION\_TIME\_DEF\_LEAPS, 1287  
 CFE\_MISSION\_TIME\_DEF\_MET\_SECS, 1287  
 CFE\_MISSION\_TIME\_DEF\_MET\_SUBS, 1288  
 CFE\_MISSION\_TIME\_DEF\_STCF\_SECS, 1288  
 CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS, 1288  
 CFE\_MISSION\_TIME\_EPOCH\_DAY, 1288  
 CFE\_MISSION\_TIME\_EPOCH\_HOUR, 1288  
 CFE\_MISSION\_TIME\_EPOCH\_MICROS, 1288  
 CFE\_MISSION\_TIME\_EPOCH\_MINUTE, 1288  
 CFE\_MISSION\_TIME\_EPOCH\_SECOND, 1288  
 CFE\_MISSION\_TIME\_EPOCH\_YEAR, 1289  
 CFE\_MISSION\_TIME\_FS\_FACTOR, 1289  
 CFE\_MISSION\_TIME\_MAX\_ELAPSED, 1289  
 CFE\_MISSION\_TIME\_MIN\_ELAPSED, 1289

default\_cfe\_time\_internal\_cfg.h  
 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY, 1291  
 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE, 1291  
 CFE\_PLATFORM\_TIME\_CFG\_CLIENT, 1291  
 CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY, 1291  
 CFE\_PLATFORM\_TIME\_CFG\_SERVER, 1291  
 CFE\_PLATFORM\_TIME\_CFG\_SIGNAL, 1291  
 CFE\_PLATFORM\_TIME\_CFG\_SOURCE, 1292  
 CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS, 1292  
 CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET, 1292  
 CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME, 1292  
 CFE\_PLATFORM\_TIME\_CFG\_START\_FLY, 1293  
 CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT, 1293  
 CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL, 1293  
 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS, 1293  
 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS, 1294  
 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS, 1294  
 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS, 1294  
 CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY, 1294  
 CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE, 1294  
 CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY, 1295  
 CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE, 1295

default\_cfe\_time\_msgids.h  
 CFE\_TIME\_1HZ\_CMD\_MID, 1297  
 CFE\_TIME\_CMD\_MID, 1297  
 CFE\_TIME\_DATA\_CMD\_MID, 1297  
 CFE\_TIME\_DIAG\_TLM\_MID, 1297

CFE\_TIME\_HK\_TLM\_MID, 1297  
CFE\_TIME\_SEND\_CMD\_MID, 1298  
CFE\_TIME\_SEND\_HK\_MID, 1298  
CFE\_TIME\_TONE\_CMD\_MID, 1298  
default\_cfe\_time\_msgstruct.h  
    CFE\_TIME\_1HzCmd\_t, 1300  
    CFE\_TIME\_Add1HZAdjustmentCmd\_t, 1300  
    CFE\_TIME\_AddAdjustCmd\_t, 1300  
    CFE\_TIME\_AddDelayCmd\_t, 1300  
    CFE\_TIME\_DiagnosticTlm\_Payload\_t, 1300  
    CFE\_TIME\_DiagnosticTlm\_t, 1300  
    CFE\_TIME\_FakeToneCmd\_t, 1300  
    CFE\_TIME\_HousekeepingTlm\_Payload\_t, 1301  
    CFE\_TIME\_HousekeepingTlm\_t, 1301  
    CFE\_TIME\_LeapsCmd\_Payload\_t, 1301  
    CFE\_TIME\_NoArgsCmd\_t, 1301  
    CFE\_TIME\_NoopCmd\_t, 1301  
    CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t, 1301  
    CFE\_TIME\_OneHzAdjustmentCmd\_t, 1301  
    CFE\_TIME\_ResetCountersCmd\_t, 1301  
    CFE\_TIME\_SendDiagnosticCmd\_t, 1301  
    CFE\_TIME\_SendHkCmd\_t, 1301  
    CFE\_TIME\_SetLeapSecondsCmd\_t, 1301  
    CFE\_TIME\_SetMETCmd\_t, 1301  
    CFE\_TIME\_SetSignalCmd\_t, 1302  
    CFE\_TIME\_SetSourceCmd\_t, 1302  
    CFE\_TIME\_SetStateCmd\_t, 1302  
    CFE\_TIME\_SetSTCFCmd\_t, 1302  
    CFE\_TIME\_SetTimeCmd\_t, 1302  
    CFE\_TIME\_SignalCmd\_Payload\_t, 1302  
    CFE\_TIME\_SourceCmd\_Payload\_t, 1302  
    CFE\_TIME\_StateCmd\_Payload\_t, 1302  
    CFE\_TIME\_Sub1HZAdjustmentCmd\_t, 1302  
    CFE\_TIME\_SubAdjustCmd\_t, 1302  
    CFE\_TIME\_SubDelayCmd\_t, 1302  
    CFE\_TIME\_TimeCmd\_Payload\_t, 1302  
    CFE\_TIME\_TimeCmd\_t, 1302  
    CFE\_TIME\_ToneDataCmd\_Payload\_t, 1303  
    CFE\_TIME\_ToneDataCmd\_t, 1303  
    CFE\_TIME\_ToneSignalCmd\_t, 1303  
default\_cfe\_time\_topicids.h  
    CFE\_MISSION\_TIME\_1HZ\_CMD\_MSG, 1303  
    CFE\_MISSION\_TIME\_CMD\_MSG, 1304  
    CFE\_MISSION\_TIME\_DATA\_CMD\_MSG, 1304  
    CFE\_MISSION\_TIME\_DIAG\_TLM\_MSG, 1304  
    CFE\_MISSION\_TIME\_HK\_TLM\_MSG, 1304  
    CFE\_MISSION\_TIME\_SEND\_CMD\_MSG, 1304  
    CFE\_MISSION\_TIME\_SEND\_HK\_MSG, 1304  
    CFE\_MISSION\_TIME\_TONE\_CMD\_MSG, 1305  
DefaultAppDefTable  
    CS\_AppData\_t, 657  
DefaultEepromDefTable  
    CS\_AppData\_t, 658  
DefaultMemoryDefTable  
    CS\_AppData\_t, 658  
DefaultTablesDefTable  
    CS\_AppData\_t, 658  
DefEepromTableHandle  
    CS\_AppData\_t, 658  
DefEepromTblPtr  
    CS\_AppData\_t, 658  
DefMemoryTableHandle  
    CS\_AppData\_t, 658  
DefMemoryTblPtr  
    CS\_AppData\_t, 658  
DefTablesTableHandle  
    CS\_AppData\_t, 659  
DefTablesTblPtr  
    CS\_AppData\_t, 659  
DelayDirection  
    CFE\_TIME\_DiagnosticTlm\_Payload, 638  
Description  
    CFE\_FS\_FileWriteMetaData, 585  
    CFE\_FS\_Header, 586  
    CFE\_TBL\_FileDef, 616  
DoubleBuffered  
    CFE\_TBL\_Info, 622  
    CFE\_TBL\_TblRegPacket\_Payload, 630  
DumpFilename  
    CFE\_ES\_DumpCDSRegistryCmd\_Payload, 536  
    CFE\_TBL\_DumpCmd\_Payload, 613  
    CFE\_TBL\_DumpRegistryCmd\_Payload, 614  
DumpOnly  
    CFE\_TBL\_Info, 622  
    CFE\_TBL\_TblRegPacket\_Payload, 630  
DuplicateSubscriptionsCounter  
    CFE\_SB\_HousekeepingTlm\_Payload, 591  
EepResTablesTblPtr  
    CS\_AppData\_t, 659  
EepromBaseline  
    CS\_HkPacket\_Payload\_t, 671  
EepromCSErrCounter  
    CS\_HkPacket\_Payload\_t, 671  
EepromCSState  
    CS\_HkPacket\_Payload\_t, 671  
Entries  
    CFE\_SB\_AllSubscriptionsTlm\_Payload, 588  
Entry  
    CFE\_SB\_AllSubscriptionsTlm\_Payload, 588  
entry\_point  
    OS\_module\_prop\_t, 690  
EntryID  
    CS\_EntryCmd\_Payload\_t, 666  
EntryPoint  
    CFE\_ES\_AppInfo, 528  
ERLogEntries  
    CFE\_ES\_HousekeepingTlm\_Payload, 541

ERLogIndex  
 CFE\_ES\_HousekeepingTIm\_Payload, 541

EventID  
 CFE\_EVS\_AppNameEventIDCmd\_Payload, 568  
 CFE\_EVS\_AppNameEventIDMaskCmd\_Payload,  
 569  
 CFE\_EVS\_BinFilter, 571  
 CFE\_EVS\_PacketID, 579

EventType  
 CFE\_EVS\_PacketID, 579

example\_mission\_cfg.h  
 CFE\_FS\_FILE\_CONTENT\_ID, 915  
 CFE\_FS\_HDR\_DESC\_MAX\_LEN, 915  
 CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN,  
 915  
 CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH,  
 915  
 CFE\_MISSION\_ES\_CRC\_16, 916  
 CFE\_MISSION\_ES\_CRC\_32, 916  
 CFE\_MISSION\_ES\_CRC\_8, 916  
 CFE\_MISSION\_ES\_DEFAULT\_CRC, 916  
 CFE\_MISSION\_ES\_MAX\_APPLICATIONS, 916  
 CFE\_MISSION\_ES\_PERF\_MAX\_IDS, 916  
 CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS, 917  
 CFE\_MISSION\_ES\_MAX\_MESSAGE\_LENGTH,  
 917  
 CFE\_MISSION\_MAX\_API\_LEN, 917  
 CFE\_MISSION\_MAX\_FILE\_LEN, 918  
 CFE\_MISSION\_MAX\_NUM\_FILES, 918  
 CFE\_MISSION\_MAX\_PATH\_LEN, 919  
 CFE\_MISSION\_SB\_MAX\_PIPES, 919  
 CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE, 919  
 CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN, 920  
 CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, 920  
 CFE\_MISSION\_TIME\_AT\_TONE\_WAS, 920  
 CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE, 921  
 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI, 921  
 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC, 921  
 CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE, 921  
 CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS, 922  
 CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS, 922  
 CFE\_MISSION\_TIME\_DEF\_LEAPS, 922  
 CFE\_MISSION\_TIME\_DEF\_MET\_SECS, 922  
 CFE\_MISSION\_TIME\_DEF\_MET\_SUBS, 922  
 CFE\_MISSION\_TIME\_DEF\_STCF\_SECS, 922  
 CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS, 923  
 CFE\_MISSION\_TIME\_EPOCH\_DAY, 923  
 CFE\_MISSION\_TIME\_EPOCH\_HOUR, 923  
 CFE\_MISSION\_TIME\_EPOCH\_MICROS, 923  
 CFE\_MISSION\_TIME\_EPOCH\_MINUTE, 923  
 CFE\_MISSION\_TIME\_EPOCH\_SECOND, 923  
 CFE\_MISSION\_TIME\_EPOCH\_YEAR, 923  
 CFE\_MISSION\_TIME\_FS\_FACTOR, 923  
 CFE\_MISSION\_TIME\_MAX\_ELAPSED, 924

CFE\_MISSION\_TIME\_MIN\_ELAPSED, 924

example\_platform\_cfg.h  
 CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC,  
 928  
 CFE\_PLATFORM\_ENDIAN, 929  
 CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT, 929  
 CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE, 929  
 CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE,  
 930  
 CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES,  
 930  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01,  
 930  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02,  
 930  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12,  
 931  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13,  
 932  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14,  
 932  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15,  
 932  
 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16,  
 932  
 CFE\_PLATFORM\_ES\_CDS\_SIZE, 932  
 CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE,  
 932  
 CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE,  
 933  
 CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE,  
 933  
 CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME,  
 933  
 CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE,  
 933

CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE, 934  
CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE, 934  
CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE, 934  
CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE, 935  
CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES, 935  
CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE, 935  
CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS, 936  
CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE, 936  
CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS, 936  
CFE\_PLATFORM\_ES\_MAX\_LIBRARIES, 936  
CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS, 937  
CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS, 937  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01, 937  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10, 938  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15, 939  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16, 939  
CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN, 939  
CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING, 939  
CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE, 940  
CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE, 940  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY, 940  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY, 940  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE, 941  
CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE, 941  
CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS, 941  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL, 942  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT, 942  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE, 942  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL, 942  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT, 943  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE, 943  
CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS, 943  
CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING, 943  
CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS, 944  
CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED, 944  
CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE, 944  
CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY, 945  
CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE, 945  
CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC, 945  
CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC, 946  
CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE, 946  
CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE, 946  
CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE, 947  
CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC, 947  
CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE, 947  
CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE, 948  
CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE, 948  
CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE, 948  
CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG, 949  
CFE\_PLATFORM\_EVS\_LOG\_MAX, 949  
CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST, 949  
CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS, 950  
CFE\_PLATFORM\_EVS\_PORT\_DEFAULT, 950  
CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY, 950  
CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE, 950  
CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES, 951  
CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME, 951  
CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT, 951  
CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME,

CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME, 952  
 CFE\_PLATFORM\_SB\_FILTER\_MASK1, 952  
 CFE\_PLATFORM\_SB\_FILTER\_MASK2, 952  
 CFE\_PLATFORM\_SB\_FILTER\_MASK3, 953  
 CFE\_PLATFORM\_SB\_FILTER\_MASK4, 953  
 CFE\_PLATFORM\_SB\_FILTER\_MASK5, 953  
 CFE\_PLATFORM\_SB\_FILTER\_MASK6, 953  
 CFE\_PLATFORM\_SB\_FILTER\_MASK7, 953  
 CFE\_PLATFORM\_SB\_FILTER\_MASK8, 953  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT1, 953  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT2, 953  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT3, 953  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT4, 953  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT5, 954  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT6, 954  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT7, 954  
 CFE\_PLATFORM\_SB\_FILTERED\_EVENT8, 954  
 CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID, 954  
 CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE, 954  
 CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT, 954  
 CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS, 955  
 CFE\_PLATFORM\_SB\_MAX\_PIPES, 955  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01, 955  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10, 956  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11, 957  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12, 957  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13, 957  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14, 957  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15, 957  
 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16, 957  
 CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY, 957  
 CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE, 957  
 CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES, 958  
 CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE, 958  
 CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES, 958  
 CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE, 959  
 CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES, 959  
 CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES, 959  
 CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS, 960  
 CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS, 960  
 CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE, 960  
 CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY, 961  
 CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE, 961  
 CFE\_PLATFORM\_TBL\_U32FROM4CHARS, 961  
 CFE\_PLATFORM\_TBL\_VALID\_PRID\_1, 962  
 CFE\_PLATFORM\_TBL\_VALID\_PRID\_2, 962  
 CFE\_PLATFORM\_TBL\_VALID\_PRID\_3, 962  
 CFE\_PLATFORM\_TBL\_VALID\_PRID\_4, 962  
 CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT, 962  
 CFE\_PLATFORM\_TBL\_VALID\_SCID\_1, 962  
 CFE\_PLATFORM\_TBL\_VALID\_SCID\_2, 963  
 CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT, 963  
 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_PRIORITY, 963  
 CFE\_PLATFORM\_TIME\_1HZ\_TASK\_STACK\_SIZE, 963  
 CFE\_PLATFORM\_TIME\_CFG\_CLIENT, 963  
 CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY, 963  
 CFE\_PLATFORM\_TIME\_CFG\_SERVER, 964  
 CFE\_PLATFORM\_TIME\_CFG\_SIGNAL, 964  
 CFE\_PLATFORM\_TIME\_CFG\_SOURCE, 964  
 CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS, 965  
 CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET, 965  
 CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME, 965  
 CFE\_PLATFORM\_TIME\_CFG\_START\_FLY, 965  
 CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT, 966  
 CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL, 966  
 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS, 966  
 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS, 967  
 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS, 967  
 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS, 967  
 CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY, 967  
 CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE, 967  
 CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY, 968  
 CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE, 968  
 ExceptionAction  
     CFE\_ES\_AppInfo, 528  
     CFE\_ES\_StartAppCmd\_Payload, 560  
 ExecutionCounter  
     CFE\_ES\_AppInfo, 528  
     CFE\_ES\_TaskInfo, 563

FailedValCounter  
    CFE\_TBL\_HousekeepingTlm\_Payload, 619

FileCreateTimeSecs  
    CFE\_TBL\_Info, 623  
    CFE\_TBL\_TblRegPacket\_Payload, 630

FileCreateTimeSubSecs  
    CFE\_TBL\_Info, 623  
    CFE\_TBL\_TblRegPacket\_Payload, 630

FileModeBits  
    os\_fstat\_t, 688

FileName  
    CFE\_ES\_AppInfo, 528  
    CFE\_ES\_FileNameCmd\_Payload, 537  
    CFE\_FS\_FileWriteMetaData, 585  
    os\_dirent\_t, 685

Filename  
    CFE\_SB\_WriteFileInfoCmd\_Payload, 608

filename  
    OS\_module\_prop\_t, 690

FileSize  
    os\_fstat\_t, 688

FileSubType  
    CFE\_FS\_FileWriteMetaData, 585

FileTime  
    os\_fstat\_t, 688

Filler16  
    CS\_Def\_EepromMemory\_Table\_Entry\_t, 664

Filler32  
    CS\_Res\_EepromMemory\_Table\_Entry\_t, 679

Filler8  
    CS\_HkPacket\_Payload\_t, 671  
    CS\_Res\_Tables\_Table\_Entry\_t, 681

FilterMask  
    CFE\_ES\_SetPerfFilterMaskCmd\_Payload, 557

FilterMaskNum  
    CFE\_ES\_SetPerfFilterMaskCmd\_Payload, 557

flags  
    OS\_module\_address\_t, 689

Forced2Fly  
    CFE\_TIME\_DiagnosticTlm\_Payload, 638

free\_blocks  
    OS\_heap\_prop\_t, 688

free\_bytes  
    OS\_heap\_prop\_t, 688

FreeFds  
    os\_fsinfo\_t, 687

freerun\_time  
    OS\_timebase\_prop\_t, 697

FreeVolumes  
    os\_fsinfo\_t, 687

GetData  
    CFE\_FS\_FileWriteMetaData, 585

GetPipeIdByNameErrorCounter

                CFE\_SB\_HousekeepingTlm\_Payload, 591

Handle  
    CFE\_ES\_CDSRegDumpRec, 533

HeapBlocksFree  
    CFE\_ES\_HousekeepingTlm\_Payload, 541

HeapBytesFree  
    CFE\_ES\_HousekeepingTlm\_Payload, 541

HeapMaxBlockSize  
    CFE\_ES\_HousekeepingTlm\_Payload, 541

HkPacket  
    CS\_AppData\_t, 659

host\_module\_id  
    OS\_module\_prop\_t, 690

InactiveBufferAddr  
    CFE\_TBL\_TblRegPacket\_Payload, 630

Index  
    CFE\_SB\_MsgMapFileEntry, 595

int16  
    common\_types.h, 1318

int32  
    common\_types.h, 1318

int64  
    common\_types.h, 1318

int8  
    common\_types.h, 1318

InternalErrorCounter  
    CFE\_SB\_HousekeepingTlm\_Payload, 591

interval\_time  
    OS\_timer\_prop\_t, 698

intptr  
    common\_types.h, 1318

IsCSOwner  
    CS\_Res\_Tables\_Table\_Entry\_t, 681

IsPending  
    CFE\_FS\_FileWriteMetaData, 585

IsValid  
    OS\_file\_prop\_t, 686

largest\_free\_block  
    OS\_heap\_prop\_t, 688

LastFileDumped  
    CFE\_TBL\_HousekeepingTlm\_Payload, 619

LastFileLoaded  
    CFE\_TBL\_HousekeepingTlm\_Payload, 619  
    CFE\_TBL\_Info, 623  
    CFE\_TBL\_TblRegPacket\_Payload, 630

LastOneShotAddress  
    CS\_HkPacket\_Payload\_t, 671

LastOneShotChecksum  
    CS\_HkPacket\_Payload\_t, 671

LastOneShotMaxBytesPerCycle  
    CS\_HkPacket\_Payload\_t, 671

LastOneShotSize

CS\_HkPacket\_Payload\_t, 671  
 LastTableLoaded  
     CFE\_TBL\_HousekeepingTlm\_Payload, 619  
 LastUpdatedTable  
     CFE\_TBL\_HousekeepingTlm\_Payload, 620  
 LastUpdateTime  
     CFE\_TBL\_HousekeepingTlm\_Payload, 620  
 LastValCrc  
     CFE\_TBL\_HousekeepingTlm\_Payload, 620  
 LastValStatus  
     CFE\_TBL\_HousekeepingTlm\_Payload, 620  
 LastValTableName  
     CFE\_TBL\_HousekeepingTlm\_Payload, 620  
 LeapSeconds  
     CFE\_TIME\_HousekeepingTlm\_Payload, 643  
     CFE\_TIME\_LeapsCmd\_Payload, 645  
 Length  
     CCSDS\_PrimaryHeader, 526  
     CFE\_FS\_Header, 586  
 LENGTHCHECK  
     osapi-macros.h, 1339  
 LoadFilename  
     CFE\_TBL\_LoadCmd\_Payload, 624  
 LoadPending  
     CFE\_TBL\_TblRegPacket\_Payload, 630  
 LocalIntCounter  
     CFE\_TIME\_DiagnosticTlm\_Payload, 638  
 LocalTaskCounter  
     CFE\_TIME\_DiagnosticTlm\_Payload, 638  
 LogEnabled  
     CFE\_EVS\_HousekeepingTlm\_Payload, 574  
 LogFilename  
     CFE\_EVS\_LogFileCmd\_Payload, 576  
 LogFullFlag  
     CFE\_EVS\_HousekeepingTlm\_Payload, 574  
 LogMode  
     CFE\_EVS\_HousekeepingTlm\_Payload, 574  
     CFE\_EVS\_SetLogMode\_Payload, 581  
 LogOverflowCounter  
     CFE\_EVS\_HousekeepingTlm\_Payload, 574  
 LongDouble  
     CFE\_ES\_PoolAlign, 551  
     CFE\_SB\_Msg, 593  
 LongInt  
     CFE\_ES\_PoolAlign, 551  
     CFE\_SB\_Msg, 593  
 MainTaskId  
     CFE\_ES\_AppInfo, 529  
 MainTaskName  
     CFE\_ES\_AppInfo, 529  
 Mask  
     CFE\_EVS\_AppNameEventIDMaskCmd\_Payload,  
         569  
             CFE\_EVS\_BinFilter, 571  
 MaxBytesPerCycle  
     CS\_AppData\_t, 659  
     CS\_OneShotCmd\_Payload\_t, 675  
 MaxElapsed  
     CFE\_TIME\_DiagnosticTlm\_Payload, 638  
 MaxFds  
     os\_finfo\_t, 687  
 MaxLocalClock  
     CFE\_TIME\_DiagnosticTlm\_Payload, 638  
 MaxMemAllowed  
     CFE\_SB\_StatsTlm\_Payload, 604  
 MaxMsgIdsAllowed  
     CFE\_SB\_StatsTlm\_Payload, 604  
 MaxPipeDepthAllowed  
     CFE\_SB\_StatsTlm\_Payload, 604  
 MaxPipesAllowed  
     CFE\_SB\_StatsTlm\_Payload, 605  
 MaxPRCount  
     CFE\_ES\_SetMaxPRCountCmd\_Payload, 556  
 MaxProcessorResets  
     CFE\_ES\_HousekeepingTlm\_Payload, 541  
 MaxQueueDepth  
     CFE\_SB\_PipeDepthStats, 595  
     CFE\_SB\_PipeInfoEntry, 597  
 MaxSubscriptionsAllowed  
     CFE\_SB\_StatsTlm\_Payload, 605  
 MaxVolumes  
     os\_finfo\_t, 687  
 MemInUse  
     CFE\_SB\_HousekeepingTlm\_Payload, 591  
     CFE\_SB\_StatsTlm\_Payload, 605  
 MemoryCSErrCounter  
     CS\_HkPacket\_Payload\_t, 672  
 MemoryCSState  
     CS\_HkPacket\_Payload\_t, 672  
 MemPoolHandle  
     CFE\_SB\_HousekeepingTlm\_Payload, 591  
     CFE\_TBL\_HousekeepingTlm\_Payload, 620  
 MemResTablesTblPtr  
     CS\_AppData\_t, 660  
 Message  
     CFE\_EVS\_LongEventTlm\_Payload, 577  
 MessageFormatMode  
     CFE\_EVS\_HousekeepingTlm\_Payload, 575  
 MessageSendCounter  
     CFE\_EVS\_HousekeepingTlm\_Payload, 575  
 MessageTruncCounter  
     CFE\_EVS\_HousekeepingTlm\_Payload, 575  
 MicroSeconds  
     CFE\_TIME\_TimeCmd\_Payload, 653  
 MinElapsed  
     CFE\_TIME\_DiagnosticTlm\_Payload, 639  
 Mode

CFE\_ES\_OverWriteSysLogCmd\_Payload, 550  
Module  
  OS\_static\_symbol\_record\_t, 694  
Msg  
  CFE\_SB\_Msg, 593  
MsgCnt  
  CFE\_SB\_RoutingFileEntry, 600  
MsgFormat  
  CFE\_EVS\_SetEventFormatCode\_Payload, 580  
MsgId  
  CFE\_SB\_MsgMapFileEntry, 595  
  CFE\_SB\_RouteCmd\_Payload, 599  
  CFE\_SB\_RoutingFileEntry, 601  
  CFE\_SB\_SingleSubscriptionTlm\_Payload, 602  
  CFE\_SB\_SubEntries, 607  
MsgIdsInUse  
  CFE\_SB\_StatsTlm\_Payload, 605  
MsgLimitErrorCounter  
  CFE\_SB\_HousekeepingTlm\_Payload, 591  
MsgReceiveErrorCounter  
  CFE\_SB\_HousekeepingTlm\_Payload, 592  
MsgSendErrorCounter  
  CFE\_SB\_HousekeepingTlm\_Payload, 592

Name  
  CFE\_ES\_AppInfo, 529  
  CFE\_ES\_CDSRegDumpRec, 533  
  CFE\_TBL\_TblRegPacket\_Payload, 631  
  CS\_AppNameCmd\_Payload\_t, 662  
  CS\_Def\_App\_Table\_Entry\_t, 663  
  CS\_Def\_Tables\_Table\_Entry\_t, 665  
  CS\_Res\_App\_Table\_Entry\_t, 677  
  CS\_Res\_Tables\_Table\_Entry\_t, 681  
  CS\_TableNameCmd\_Payload\_t, 682  
  OS\_static\_symbol\_record\_t, 694  
name  
  OS\_bin\_sem\_prop\_t, 683  
  OS\_condvar\_prop\_t, 684  
  OS\_count\_sem\_prop\_t, 684  
  OS\_module\_prop\_t, 690  
  OS\_mut\_sem\_prop\_t, 691  
  OS\_queue\_prop\_t, 691  
  OS\_socket\_prop\_t, 693  
  OS\_task\_prop\_t, 695  
  OS\_timebase\_prop\_t, 697  
  OS\_timer\_prop\_t, 698  
nominal\_interval\_time  
  OS\_timebase\_prop\_t, 697  
NoSubscribersCounter  
  CFE\_SB\_HousekeepingTlm\_Payload, 592  
NumBlocksRequested  
  CFE\_ES\_MemPoolStats, 546  
NumBytes  
  CFE\_TBL\_File\_Hdr, 615  
NumBytesToChecksum  
  CS\_Def\_EepromMemory\_Table\_Entry\_t, 664  
  CS\_Res\_App\_Table\_Entry\_t, 677  
  CS\_Res\_EepromMemory\_Table\_Entry\_t, 679  
  CS\_Res\_Tables\_Table\_Entry\_t, 681  
NumCreated  
  CFE\_ES\_BlockStats, 532  
NumFree  
  CFE\_ES\_BlockStats, 532  
NumFreeBytes  
  CFE\_ES\_MemPoolStats, 546  
NumFreeSharedBufs  
  CFE\_TBL\_HousekeepingTlm\_Payload, 620  
NumLoadPending  
  CFE\_TBL\_HousekeepingTlm\_Payload, 621  
NumOfChildTasks  
  CFE\_ES\_AppInfo, 529  
NumTables  
  CFE\_TBL\_HousekeepingTlm\_Payload, 621  
NumUsers  
  CFE\_TBL\_Info, 623  
NumValRequests  
  CFE\_TBL\_HousekeepingTlm\_Payload, 621

object\_ids  
  OS\_FdSet, 685  
ObjectName  
  CFE\_TBL\_FileDef, 616  
ObjectSize  
  CFE\_TBL\_FileDef, 616  
Offset  
  CFE\_TBL\_File\_Hdr, 615  
OneHzAdjust  
  CFE\_TIME\_DiagnosticTlm\_Payload, 639  
OneHzDirection  
  CFE\_TIME\_DiagnosticTlm\_Payload, 639  
OneShotInProgress  
  CS\_HkPacket\_Payload\_t, 672  
OneTimeAdjust  
  CFE\_TIME\_DiagnosticTlm\_Payload, 639  
OneTimeDirection  
  CFE\_TIME\_DiagnosticTlm\_Payload, 639  
OnEvent  
  CFE\_FS\_FileWriteMetaData, 585  
Opts  
  CFE\_SB\_PipeInfoEntry, 597  
OS\_ADD\_TASK\_FLAGS  
  osconfig.h, 909  
OS\_API\_Init  
  OSAL Core Operation APIs, 419  
OS\_API\_Teardown  
  OSAL Core Operation APIs, 420  
OS\_Application\_Run  
  OSAL Core Operation APIs, 420

OS\_Application\_Startup  
     OSAL Core Operation APIs, 420

OS\_ApplicationExit  
     OSAL Core Operation APIs, 420

OS\_ApplicationShutdown  
     OSAL Core Operation APIs, 420

OS\_ArgCallback\_t  
     common\_types.h, 1318

OS\_bin\_sem\_prop\_t, 683  
     creator, 683  
     name, 683  
     value, 683

OS\_BinSemCreate  
     OSAL Binary Semaphore APIs, 402

OS\_BinSemDelete  
     OSAL Binary Semaphore APIs, 403

OS\_BinSemFlush  
     OSAL Binary Semaphore APIs, 403

OS\_BinSemGetIdByName  
     OSAL Binary Semaphore APIs, 404

OS\_BinSemGetInfo  
     OSAL Binary Semaphore APIs, 404

OS\_BinSemGive  
     OSAL Binary Semaphore APIs, 405

OS\_BinSemTake  
     OSAL Binary Semaphore APIs, 405

OS\_BinSemTimedWait  
     OSAL Binary Semaphore APIs, 406

OS\_BSP\_GetArgC  
     OSAL BSP low level access APIs, 407

OS\_BSP\_GetArgV  
     OSAL BSP low level access APIs, 407

OS\_BSP\_GetResourceTypeConfig  
     OSAL BSP low level access APIs, 407

OS\_BSP\_SetExitCode  
     OSAL BSP low level access APIs, 407

OS\_BSP\_SetResourceTypeConfig  
     OSAL BSP low level access APIs, 407

OS\_BUFFER\_MSG\_DEPTH  
     osconfig.h, 909

OS\_BUFFER\_SIZE  
     osconfig.h, 909

OS\_BUILD\_BASELINE  
     osapi-version.h, 1351

OS\_BUILD\_NUMBER  
     osapi-version.h, 1352

OS\_CHECK  
     osapi-constants.h, 1326

OS\_CHK\_ONLY  
     osapi-fs.h, 1335

OS\_chkfs  
     OSAL File System Level APIs, 459

OS\_chmod  
     OSAL Standard File APIs, 448

OS\_close  
     OSAL Standard File APIs, 449

OS\_CloseAllFiles  
     OSAL Standard File APIs, 449

OS\_CloseFileByName  
     OSAL Standard File APIs, 450

OS\_condvar\_prop\_t, 683  
     creator, 684  
     name, 684

OS\_CondVarBroadcast  
     OSAL Condition Variable APIs, 422

OS\_CondVarCreate  
     OSAL Condition Variable APIs, 423

OS\_CondVarDelete  
     OSAL Condition Variable APIs, 424

OS\_CondVarGetIdByName  
     OSAL Condition Variable APIs, 424

OS\_CondVarGetInfo  
     OSAL Condition Variable APIs, 425

OS\_CondVarLock  
     OSAL Condition Variable APIs, 425

OS\_CondVarSignal  
     OSAL Condition Variable APIs, 425

OS\_CondVarTimedWait  
     OSAL Condition Variable APIs, 426

OS\_CondVarUnlock  
     OSAL Condition Variable APIs, 426

OS\_CondVarWait  
     OSAL Condition Variable APIs, 427

OS\_ConvertToArrayIndex  
     OSAL Object ID Utility APIs, 471

OS\_count\_sem\_prop\_t, 684  
     creator, 684  
     name, 684  
     value, 684

OS\_CountSemCreate  
     OSAL Counting Semaphore APIs, 428

OS\_CountSemDelete  
     OSAL Counting Semaphore APIs, 429

OS\_CountSemGetIdByName  
     OSAL Counting Semaphore APIs, 429

OS\_CountSemGetInfo  
     OSAL Counting Semaphore APIs, 430

OS\_CountSemGive  
     OSAL Counting Semaphore APIs, 430

OS\_CountSemTake  
     OSAL Counting Semaphore APIs, 431

OS\_CountSemTimedWait  
     OSAL Counting Semaphore APIs, 431

OS\_cp  
     OSAL Standard File APIs, 450

OS\_DeleteAllObjects  
     OSAL Core Operation APIs, 421

OS\_DirectoryClose

OSAL Directory APIs, 433  
OS\_DirectoryOpen  
    OSAL Directory APIs, 433  
OS\_DirectoryRead  
    OSAL Directory APIs, 434  
OS\_DirectoryRewind  
    OSAL Directory APIs, 434  
os\_dirent\_t, 685  
    FileName, 685  
OS\_DIRENTRY\_NAME  
    osapi-dir.h, 1328  
OS\_ERR\_BAD\_ADDRESS  
    OSAL Return Code Defines, 439  
OS\_ERR\_FILE  
    OSAL Return Code Defines, 439  
OS\_ERR\_INCORRECT\_OBJ\_STATE  
    OSAL Return Code Defines, 439  
OS\_ERR\_INCORRECT\_OBJ\_TYPE  
    OSAL Return Code Defines, 439  
OS\_ERR\_INVALID\_ARGUMENT  
    OSAL Return Code Defines, 439  
OS\_ERR\_INVALID\_ID  
    OSAL Return Code Defines, 439  
OS\_ERR\_INVALID\_PRIORITY  
    OSAL Return Code Defines, 440  
OS\_ERR\_INVALID\_SIZE  
    OSAL Return Code Defines, 440  
OS\_ERR\_NAME\_NOT\_FOUND  
    OSAL Return Code Defines, 440  
os\_err\_name\_t  
    osapi-error.h, 1331  
OS\_ERR\_NAME\_TAKEN  
    OSAL Return Code Defines, 440  
OS\_ERR\_NAME\_TOO\_LONG  
    OSAL Return Code Defines, 440  
OS\_ERR\_NO\_FREE\_IDS  
    OSAL Return Code Defines, 440  
OS\_ERR\_NOT\_IMPLEMENTED  
    OSAL Return Code Defines, 440  
OS\_ERR\_OBJECT\_IN\_USE  
    OSAL Return Code Defines, 440  
OS\_ERR\_OPERATION\_NOT\_SUPPORTED  
    OSAL Return Code Defines, 440  
OS\_ERR\_OUTPUT\_TOO\_LARGE  
    OSAL Return Code Defines, 440  
OS\_ERR\_SEM\_NOT\_FULL  
    OSAL Return Code Defines, 441  
OS\_ERR\_STREAM\_DISCONNECTED  
    OSAL Return Code Defines, 441  
OS\_ERROR  
    OSAL Return Code Defines, 441  
OS\_ERROR\_ADDRESS\_MISALIGNED  
    OSAL Return Code Defines, 441  
OS\_ERROR\_NAME\_LENGTH  
    osapi-error.h, 1331  
OS\_ERROR\_TIMEOUT  
    OSAL Return Code Defines, 441  
OS\_EVENT\_MAX  
    osapi-common.h, 1325  
OS\_EVENT\_RESERVED  
    osapi-common.h, 1324  
OS\_EVENT\_RESOURCE\_ALLOCATED  
    osapi-common.h, 1324  
OS\_EVENT\_RESOURCE\_CREATED  
    osapi-common.h, 1325  
OS\_EVENT\_RESOURCE\_DELETED  
    osapi-common.h, 1325  
OS\_Event\_t  
    osapi-common.h, 1324  
OS\_EVENT\_TASK\_STARTUP  
    osapi-common.h, 1325  
OS\_EventHandler\_t  
    osapi-common.h, 1324  
OS\_FDGetInfo  
    OSAL Standard File APIs, 451  
OS\_FdSet, 685  
    object\_ids, 685  
OS\_FILE\_FLAG\_CREATE  
    osapi-file.h, 1334  
OS\_FILE\_FLAG\_NONE  
    osapi-file.h, 1334  
OS\_file\_flag\_t  
    osapi-file.h, 1334  
OS\_FILE\_FLAG\_TRUNCATE  
    osapi-file.h, 1334  
OS\_file\_prop\_t, 686  
    IsValid, 686  
    Path, 686  
    User, 686  
OS\_FileOpenCheck  
    OSAL Standard File APIs, 451  
OS\_FILESTAT\_EXEC  
    osapi-file.h, 1333  
OS\_FILESTAT\_ISDIR  
    osapi-file.h, 1333  
OS\_FILESTAT\_MODE  
    osapi-file.h, 1333  
OS\_FILESTAT\_MODE\_DIR  
    osapi-file.h, 1334  
OS\_FILESTAT\_MODE\_EXEC  
    osapi-file.h, 1334  
OS\_FILESTAT\_MODE\_READ  
    osapi-file.h, 1334  
OS\_FILESTAT\_MODE\_WRITE  
    osapi-file.h, 1334  
OS\_FILESTAT\_READ  
    osapi-file.h, 1333  
OS\_FILESTAT\_SIZE

osapi-file.h, 1333  
OS\_FILESTAT\_TIME  
    osapi-file.h, 1333  
OS\_FILESTAT\_WRITE  
    osapi-file.h, 1334  
OS\_FileSysAddFixedMap  
    OSAL File System Level APIs, 460  
OS\_FileSysStatVolume  
    OSAL File System Level APIs, 460  
OS\_ForEachObject  
    OSAL Object ID Utility APIs, 472  
OS\_ForEachObjectType  
    OSAL Object ID Utility APIs, 472  
OS\_FP\_ENABLED  
    osapi-task.h, 1348  
OS\_FS\_DEV\_NAME\_LEN  
    osconfig.h, 909  
OS\_FS\_ERR\_DEVICE\_NOT\_FREE  
    OSAL Return Code Defines, 441  
OS\_FS\_ERR\_DRIVE\_NOT\_CREATED  
    OSAL Return Code Defines, 441  
OS\_FS\_ERR\_NAME\_TOO\_LONG  
    OSAL Return Code Defines, 441  
OS\_FS\_ERR\_PATH\_INVALID  
    OSAL Return Code Defines, 441  
OS\_FS\_ERR\_PATH\_TOO\_LONG  
    OSAL Return Code Defines, 441  
OS\_FS\_GetPhysDriveName  
    OSAL File System Level APIs, 461  
OS\_FS\_PHYS\_NAME\_LEN  
    osconfig.h, 909  
OS\_FS\_VOL\_NAME\_LEN  
    osconfig.h, 909  
os\_fsinfo\_t, 686  
    FreeFds, 687  
    FreeVolumes, 687  
    MaxFds, 687  
    MaxVolumes, 687  
os\_fstat\_t, 687  
     FileModeBits, 688  
    FileSize, 688  
    FileTime, 688  
OS\_GetBuildNumber  
    osapi-version.h, 1353  
OS\_GetErrorName  
    OSAL Error Info APIs, 444  
OS\_GetFsInfo  
    OSAL File System Level APIs, 462  
OS\_GetLocalTime  
    OSAL Real Time Clock APIs, 409  
OS\_GetResourceName  
    OSAL Object ID Utility APIs, 472  
OS\_GetVersionCodeName  
    osapi-version.h, 1353  
OS\_GetVersionNumber  
    osapi-version.h, 1353  
OS\_GetVersionString  
    osapi-version.h, 1354  
OS\_heap\_prop\_t, 688  
    free\_blocks, 688  
    free\_bytes, 688  
    largest\_free\_block, 688  
OS\_HeapGetInfo  
    OSAL Heap APIs, 467  
OS\_IdentifyObject  
    OSAL Object ID Utility APIs, 473  
OS\_IdleLoop  
    OSAL Core Operation APIs, 421  
OS\_initfs  
    OSAL File System Level APIs, 462  
OS\_INVALID\_INT\_NUM  
    OSAL Return Code Defines, 442  
OS\_INVALID\_POINTER  
    OSAL Return Code Defines, 442  
OS\_INVALID\_SEM\_VALUE  
    OSAL Return Code Defines, 442  
OS\_Iseek  
    OSAL Standard File APIs, 452  
OS\_MAJOR\_VERSION  
    osapi-version.h, 1352  
OS\_MAX\_API\_NAME  
    osconfig.h, 909  
OS\_MAX\_BIN\_SEMAPHORES  
    osconfig.h, 910  
OS\_MAX\_CMD\_LEN  
    osconfig.h, 910  
OS\_MAX\_CONDVARSS  
    osconfig.h, 910  
OS\_MAX\_CONSOLES  
    osconfig.h, 910  
OS\_MAX\_COUNT\_SEMAPHORES  
    osconfig.h, 910  
OS\_MAX\_FILE\_NAME  
    osconfig.h, 910  
OS\_MAX\_FILE\_SYSTEMS  
    osconfig.h, 910  
OS\_MAX\_LOCAL\_PATH\_LEN  
    osapi-constants.h, 1326  
OS\_MAX\_MODULES  
    osconfig.h, 911  
OS\_MAX\_MUTEXES  
    osconfig.h, 911  
OS\_MAX\_NUM\_OPEN\_DIRS  
    osconfig.h, 911  
OS\_MAX\_NUM\_OPEN\_FILES  
    osconfig.h, 911  
OS\_MAX\_PATH\_LEN  
    osconfig.h, 911

OS\_MAX\_QUEUES  
    osconfig.h, 911

OS\_MAX\_SYM\_LEN  
    osconfig.h, 911

OS\_MAX\_TASK\_PRIORITY  
    osapi-task.h, 1348

OS\_MAX\_TASKS  
    osconfig.h, 912

OS\_MAX\_TIMEBASES  
    osconfig.h, 912

OS\_MAX\_TIMERS  
    osconfig.h, 912

OS\_MINOR\_VERSION  
    osapi-version.h, 1352

OS\_MISSION\_REV  
    osapi-version.h, 1352

OS\_mkdir  
    OSAL Directory APIs, 435

OS\_mkfs  
    OSAL File System Level APIs, 463

OS\_module\_address\_t, 689

- bss\_address, 689
- bss\_size, 689
- code\_address, 689
- code\_size, 689
- data\_address, 689
- data\_size, 689
- flags, 689
- valid, 689

OS\_MODULE\_FILE\_EXTENSION  
    osconfig.h, 912

OS\_MODULE\_FLAG\_GLOBAL\_SYMBOLS  
    osapi-module.h, 1340

OS\_MODULE\_FLAG\_LOCAL\_SYMBOLS  
    osapi-module.h, 1340

OS\_module\_prop\_t, 690

- addr, 690
- entry\_point, 690
- filename, 690
- host\_module\_id, 690
- name, 690

OS\_ModuleInfo  
    OSAL Dynamic Loader and Symbol APIs, 476

OS\_ModuleLoad  
    OSAL Dynamic Loader and Symbol APIs, 476

OS\_ModuleSymbolLookup  
    OSAL Dynamic Loader and Symbol APIs, 477

OS\_ModuleUnload  
    OSAL Dynamic Loader and Symbol APIs, 478

OS\_mount  
    OSAL File System Level APIs, 463

OS\_mut\_sem\_prop\_t, 690

- creator, 691
- name, 691

OS\_MutSemCreate  
    OSAL Mutex APIs, 480

OS\_MutSemDelete  
    OSAL Mutex APIs, 480

OS\_MutSemGetIdByName  
    OSAL Mutex APIs, 481

OS\_MutSemGetInfo  
    OSAL Mutex APIs, 481

OS\_MutSemGive  
    OSAL Mutex APIs, 482

OS\_MutSemTake  
    OSAL Mutex APIs, 482

OS\_mv  
    OSAL Standard File APIs, 452

OS\_NetworkGetHostName  
    OSAL Network ID APIs, 484

OS\_NetworkGetID  
    OSAL Network ID APIs, 484

OS\_OBJECT\_CREATOR\_ANY  
    osapi-constants.h, 1326

OS\_OBJECT\_ID\_UNDEFINED  
    osapi-constants.h, 1326

OS\_OBJECT\_INDEX\_MASK  
    osapi-idmap.h, 1337

OS\_OBJECT\_TYPE\_OS\_BINSEM  
    OSAL Object Type Defines, 468

OS\_OBJECT\_TYPE\_OS\_CONDVAR  
    OSAL Object Type Defines, 468

OS\_OBJECT\_TYPE\_OS\_CONSOLE  
    OSAL Object Type Defines, 468

OS\_OBJECT\_TYPE\_OS\_COUNTSEM  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_DIR  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_FILESYS  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_MODULE  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_MUTEX  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_QUEUE  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_STREAM  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_TASK  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_TIMEBASE  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_OS\_TIMECB  
    OSAL Object Type Defines, 469

OS\_OBJECT\_TYPE\_SHIFT  
    osapi-idmap.h, 1337

OS\_OBJECT\_TYPE\_UNDEFINED  
    OSAL Object Type Defines, 470

OS\_OBJECT\_TYPE\_USER  
     OSAL Object Type Defines, [470](#)

OS\_ObjectIdDefined  
     OSAL Object ID Utility APIs, [473](#)

OS\_ObjectIdEqual  
     OSAL Object ID Utility APIs, [474](#)

OS\_ObjectIdFromInteger  
     OSAL Object ID Utility APIs, [474](#)

OS\_ObjectIdToArrayIndex  
     OSAL Object ID Utility APIs, [474](#)

OS\_ObjectIdToInteger  
     OSAL Object ID Utility APIs, [475](#)

OS\_OpenCreate  
     OSAL Standard File APIs, [453](#)

OS\_PEND  
     osapi-constants.h, [1327](#)

OS\_PRINTF  
     cfe\_es.h, [991](#)  
     common\_types.h, [1317](#)

OS\_printf  
     OSAL Printf APIs, [486](#)

OS\_PRINTF\_CONSOLE\_NAME  
     osconfig.h, [912](#)

OS\_printf\_disable  
     OSAL Printf APIs, [486](#)

OS\_printf\_enable  
     OSAL Printf APIs, [486](#)

OS\_QUEUE\_EMPTY  
     OSAL Return Code Defines, [442](#)

OS\_QUEUE\_FULL  
     OSAL Return Code Defines, [442](#)

OS\_QUEUE\_ID\_ERROR  
     OSAL Return Code Defines, [442](#)

OS\_QUEUE\_INVALID\_SIZE  
     OSAL Return Code Defines, [442](#)

OS\_QUEUE\_MAX\_DEPTH  
     osconfig.h, [912](#)

OS\_queue\_prop\_t, [691](#)  
     creator, [691](#)  
     name, [691](#)

OS\_QUEUE\_TIMEOUT  
     OSAL Return Code Defines, [442](#)

OS\_QueueCreate  
     OSAL Message Queue APIs, [487](#)

OS\_QueueDelete  
     OSAL Message Queue APIs, [488](#)

OS\_QueueGet  
     OSAL Message Queue APIs, [488](#)

OS\_QueueGetByName  
     OSAL Message Queue APIs, [489](#)

OS\_QueueGetInfo  
     OSAL Message Queue APIs, [489](#)

OS\_QueuePut  
     OSAL Message Queue APIs, [490](#)

OS\_read  
     OSAL Standard File APIs, [454](#)

OS\_READ\_ONLY  
     OSAL File Access Option Defines, [446](#)

OS\_READ\_WRITE  
     OSAL File Access Option Defines, [446](#)

OS\_RegisterEventHandler  
     OSAL Core Operation APIs, [421](#)

OS\_remove  
     OSAL Standard File APIs, [454](#)

OS\_rename  
     OSAL Standard File APIs, [455](#)

OS\_REPAIR  
     osapi-filesystem.h, [1335](#)

OS\_REVISION  
     osapi-version.h, [1352](#)

OS\_rmdir  
     OSAL Directory APIs, [435](#)

OS\_rmfs  
     OSAL File System Level APIs, [464](#)

OS\_SEEK\_CUR  
     OSAL Reference Point For Seek Offset Defines, [447](#)

OS\_SEEK\_END  
     OSAL Reference Point For Seek Offset Defines, [447](#)

OS\_SEEK\_SET  
     OSAL Reference Point For Seek Offset Defines, [447](#)

OS\_SelectFdAdd  
     OSAL Select APIs, [491](#)

OS\_SelectFdClear  
     OSAL Select APIs, [491](#)

OS\_SelectFdIsSet  
     OSAL Select APIs, [492](#)

OS\_SelectFdZero  
     OSAL Select APIs, [492](#)

OS\_SelectMultiple  
     OSAL Select APIs, [493](#)

OS\_SelectSingle  
     OSAL Select APIs, [494](#)

OS\_SEM\_EMPTY  
     OSAL Semaphore State Defines, [401](#)

OS\_SEM\_FAILURE  
     OSAL Return Code Defines, [442](#)

OS\_SEM\_FULL  
     OSAL Semaphore State Defines, [401](#)

OS\_SEM\_TIMEOUT  
     OSAL Return Code Defines, [442](#)

OS\_SetLocalTime  
     OSAL Real Time Clock APIs, [409](#)

OS\_SHELL\_CMD\_INPUT\_FILE\_NAME  
     osconfig.h, [912](#)

OS\_ShellOutputToFile  
     OSAL Shell APIs, [495](#)

OS SOCKADDR\_MAX\_LEN  
     osapi-sockets.h, [1346](#)

osconfig.h, 912  
OS\_SockAddr\_t, 692  
  ActualLength, 692  
  AddrData, 692  
OS\_SockAddrData\_t, 692  
  AlignPtr, 693  
  AlignU32, 693  
  Buffer, 693  
OS\_socket\_prop\_t, 693  
  creator, 693  
  name, 693  
OS\_SocketAccept  
  OSAL Socket Management APIs, 500  
OS\_SocketAddrFromString  
  OSAL Socket Address APIs, 496  
OS\_SocketAddrGetPort  
  OSAL Socket Address APIs, 497  
OS\_SocketAddrInit  
  OSAL Socket Address APIs, 497  
OS\_SocketAddrSetPort  
  OSAL Socket Address APIs, 498  
OS\_SocketAddrToString  
  OSAL Socket Address APIs, 498  
OS\_SocketBind  
  OSAL Socket Management APIs, 501  
OS\_SocketBindAddress  
  OSAL Socket Management APIs, 502  
OS\_SocketConnect  
  OSAL Socket Management APIs, 502  
OS\_SocketDomain\_INET  
  osapi-sockets.h, 1346  
OS\_SocketDomain\_INET6  
  osapi-sockets.h, 1346  
OS\_SocketDomain\_INVALID  
  osapi-sockets.h, 1346  
OS\_SocketDomain\_MAX  
  osapi-sockets.h, 1346  
OS\_SocketDomain\_t  
  osapi-sockets.h, 1346  
OS\_SocketGetIdByName  
  OSAL Socket Management APIs, 503  
OS\_SocketGetInfo  
  OSAL Socket Management APIs, 503  
OS\_SocketListen  
  OSAL Socket Management APIs, 504  
OS\_SocketOpen  
  OSAL Socket Management APIs, 504  
OS\_SocketRecvFrom  
  OSAL Socket Management APIs, 505  
OS\_SocketSendTo  
  OSAL Socket Management APIs, 505  
OS\_SocketShutdown  
  OSAL Socket Management APIs, 506  
OS\_SocketShutdownMode\_NONE  
  osapi-sockets.h, 1346  
OS\_SocketShutdownMode\_SHUT\_READ  
  osapi-sockets.h, 1346  
OS\_SocketShutdownMode\_SHUT\_READWRITE  
  osapi-sockets.h, 1346  
OS\_SocketShutdownMode\_SHUT\_WRITE  
  osapi-sockets.h, 1346  
OS\_SocketShutdownMode\_t  
  osapi-sockets.h, 1346  
OS\_SocketType\_DATAGRAM  
  osapi-sockets.h, 1346  
OS\_SocketType\_INVALID  
  osapi-sockets.h, 1346  
OS\_SocketType\_MAX  
  osapi-sockets.h, 1346  
OS\_SocketType\_STREAM  
  osapi-sockets.h, 1346  
OS\_SocketType\_t  
  osapi-sockets.h, 1346  
OS\_stat  
  OSAL Standard File APIs, 455  
OS\_static\_symbol\_record\_t, 694  
  Address, 694  
  Module, 694  
  Name, 694  
OS\_STATUS\_STRING\_LENGTH  
  osapi-error.h, 1331  
os\_status\_string\_t  
  osapi-error.h, 1331  
OS\_StatusToInteger  
  OSAL Error Info APIs, 444  
OS\_StatusToString  
  OSAL Error Info APIs, 445  
OS\_statvfs\_t, 694  
  block\_size, 695  
  blocks\_free, 695  
  total\_blocks, 695  
OS\_STR  
  osapi-version.h, 1352  
OS\_STR\_HELPER  
  osapi-version.h, 1352  
OS\_STREAM\_STATE\_BOUND  
  osapi-select.h, 1344  
OS\_STREAM\_STATE\_CONNECTED  
  osapi-select.h, 1344  
OS\_STREAM\_STATE\_LISTENING  
  osapi-select.h, 1344  
OS\_STREAM\_STATE\_READABLE  
  osapi-select.h, 1344  
OS\_STREAM\_STATE\_WRITABLE  
  osapi-select.h, 1344  
OS\_StreamState\_t  
  osapi-select.h, 1343  
OS\_SUCCESS

OSAL Return Code Defines, 443  
**OS\_SymbolLookup**  
 OSAL Dynamic Loader and Symbol APIs, 478  
**OS\_SymbolTableDump**  
 OSAL Dynamic Loader and Symbol APIs, 479  
**OS\_task\_prop\_t**, 695  
 creator, 695  
 name, 695  
 priority, 695  
 stack\_size, 695  
**OS\_TaskCreate**  
 OSAL Task APIs, 508  
**OS\_TaskDelay**  
 OSAL Task APIs, 509  
**OS\_TaskDelete**  
 OSAL Task APIs, 509  
**OS\_TaskExit**  
 OSAL Task APIs, 510  
**OS\_TaskFindIdBySystemData**  
 OSAL Task APIs, 510  
**OS\_TaskGetId**  
 OSAL Task APIs, 511  
**OS\_TaskGetIdByName**  
 OSAL Task APIs, 511  
**OS\_TaskGetInfo**  
 OSAL Task APIs, 511  
**OS\_TaskInstallDeleteHandler**  
 OSAL Task APIs, 512  
**OS\_TaskSetPriority**  
 OSAL Task APIs, 512  
**OS\_time\_t**, 696  
 ticks, 696  
**OS\_TIME\_TICK\_RESOLUTION\_NS**  
 osapi-clock.h, 1323  
**OS\_TIME TICKS\_PER\_MSEC**  
 osapi-clock.h, 1323  
**OS\_TIME TICKS\_PER\_SECOND**  
 osapi-clock.h, 1323  
**OS\_TIME TICKS\_PER\_USEC**  
 osapi-clock.h, 1323  
**OS\_TimeAdd**  
 OSAL Real Time Clock APIs, 410  
**OS\_TimeAssembleFromMicroseconds**  
 OSAL Real Time Clock APIs, 410  
**OS\_TimeAssembleFromMilliseconds**  
 OSAL Real Time Clock APIs, 410  
**OS\_TimeAssembleFromNanoseconds**  
 OSAL Real Time Clock APIs, 411  
**OS\_TimeAssembleFromSubseconds**  
 OSAL Real Time Clock APIs, 411  
**OS\_timebase\_prop\_t**, 696  
 accuracy, 697  
 creator, 697  
 freerun\_time, 697  
 name, 697  
 nominal\_interval\_time, 697  
**OS\_TimeBaseCreate**  
 OSAL Time Base APIs, 514  
**OS\_TimeBaseDelete**  
 OSAL Time Base APIs, 515  
**OS\_TimeBaseGetFreeRun**  
 OSAL Time Base APIs, 515  
**OS\_TimeBaseGetIdByName**  
 OSAL Time Base APIs, 516  
**OS\_TimeBaseGetInfo**  
 OSAL Time Base APIs, 517  
**OS\_TimeBaseSet**  
 OSAL Time Base APIs, 517  
**OS\_TimedRead**  
 OSAL Standard File APIs, 456  
**OS\_TimedWrite**  
 OSAL Standard File APIs, 457  
**OS\_TimeFromTotalMicroseconds**  
 OSAL Real Time Clock APIs, 412  
**OS\_TimeFromTotalMilliseconds**  
 OSAL Real Time Clock APIs, 412  
**OS\_TimeFromTotalNanoseconds**  
 OSAL Real Time Clock APIs, 413  
**OS\_TimeFromTotalSeconds**  
 OSAL Real Time Clock APIs, 413  
**OS\_TimeGetFractionalPart**  
 OSAL Real Time Clock APIs, 413  
**OS\_TimeGetMicrosecondsPart**  
 OSAL Real Time Clock APIs, 414  
**OS\_TimeGetMillisecondsPart**  
 OSAL Real Time Clock APIs, 414  
**OS\_TimeGetNanosecondsPart**  
 OSAL Real Time Clock APIs, 415  
**OS\_TimeGetSubsecondsPart**  
 OSAL Real Time Clock APIs, 416  
**OS\_TimeGetTotalMicroseconds**  
 OSAL Real Time Clock APIs, 416  
**OS\_TimeGetTotalMilliseconds**  
 OSAL Real Time Clock APIs, 417  
**OS\_TimeGetTotalNanoseconds**  
 OSAL Real Time Clock APIs, 417  
**OS\_TimeGetTotalSeconds**  
 OSAL Real Time Clock APIs, 418  
**OS\_TIMER\_ERR\_INTERNAL**  
 OSAL Return Code Defines, 443  
**OS\_TIMER\_ERR\_INVALID\_ARGS**  
 OSAL Return Code Defines, 443  
**OS\_TIMER\_ERR\_TIMER\_ID**  
 OSAL Return Code Defines, 443  
**OS\_TIMER\_ERR\_UNAVAILABLE**  
 OSAL Return Code Defines, 443  
**OS\_timer\_prop\_t**, 697  
 accuracy, 697

creator, 698  
interval\_time, 698  
name, 698  
start\_time, 698  
**OS\_TimerAdd**  
    OSAL Timer APIs, 519  
**OS\_TimerCallback\_t**  
    osapi-timer.h, 1350  
**OS\_TimerCreate**  
    OSAL Timer APIs, 520  
**OS\_TimerDelete**  
    OSAL Timer APIs, 521  
**OS\_TimerGetIdByName**  
    OSAL Timer APIs, 522  
**OS\_TimerGetInfo**  
    OSAL Timer APIs, 522  
**OS\_TimerSet**  
    OSAL Timer APIs, 523  
**OS\_TimerSync\_t**  
    osapi-timebase.h, 1350  
**OS\_TimeSubtract**  
    OSAL Real Time Clock APIs, 418  
**OS\_TranslatePath**  
    OSAL File System Level APIs, 464  
**OS\_unmount**  
    OSAL File System Level APIs, 465  
**OS\_USED**  
    common\_types.h, 1317  
**OS\_UTILITYTASK\_PRIORITY**  
    osconfig.h, 913  
**OS\_UTILITYTASK\_STACK\_SIZE**  
    osconfig.h, 913  
**OS\_VERSION**  
    osapi-version.h, 1352  
**OS\_VERSION\_CODENAME**  
    osapi-version.h, 1352  
**OS\_VERSION\_STRING**  
    osapi-version.h, 1353  
**OS\_write**  
    OSAL Standard File APIs, 457  
**OS\_WRITE\_ONLY**  
    OSAL File Access Option Defines, 446  
**OSAL Binary Semaphore APIs**, 402  
    **OS\_BinSemCreate**, 402  
    **OS\_BinSemDelete**, 403  
    **OS\_BinSemFlush**, 403  
    **OS\_BinSemGetIdByName**, 404  
    **OS\_BinSemGetInfo**, 404  
    **OS\_BinSemGive**, 405  
    **OS\_BinSemTake**, 405  
    **OS\_BinSemTimedWait**, 406  
**OSAL BSP low level access APIs**, 407  
    **OS\_BSP\_GetArgC**, 407  
    **OS\_BSP\_GetArgV**, 407  
    **OS\_BSP\_GetResourceTypeConfig**, 407  
    **OS\_BSP\_SetExitCode**, 407  
    **OS\_BSP\_SetResourceTypeConfig**, 407  
**OSAL Condition Variable APIs**, 422  
    **OS\_CondVarBroadcast**, 422  
    **OS\_CondVarCreate**, 423  
    **OS\_CondVarDelete**, 424  
    **OS\_CondVarGetIdByName**, 424  
    **OS\_CondVarGetInfo**, 425  
    **OS\_CondVarLock**, 425  
    **OS\_CondVarSignal**, 425  
    **OS\_CondVarTimedWait**, 426  
    **OS\_CondVarUnlock**, 426  
    **OS\_CondVarWait**, 427  
**OSAL Core Operation APIs**, 419  
    **OS\_API\_Init**, 419  
    **OS\_API\_Teardown**, 420  
    **OS\_Application\_Run**, 420  
    **OS\_Application\_Startup**, 420  
    **OS\_ApplicationExit**, 420  
    **OS\_ApplicationShutdown**, 420  
    **OS\_DeleteAllObjects**, 421  
    **OS\_IdleLoop**, 421  
    **OS\_RegisterEventHandler**, 421  
**OSAL Counting Semaphore APIs**, 428  
    **OS\_CountSemCreate**, 428  
    **OS\_CountSemDelete**, 429  
    **OS\_CountSemGetIdByName**, 429  
    **OS\_CountSemGetInfo**, 430  
    **OS\_CountSemGive**, 430  
    **OS\_CountSemTake**, 431  
    **OS\_CountSemTimedWait**, 431  
**OSAL Directory APIs**, 433  
    **OS\_DirectoryClose**, 433  
    **OS\_DirectoryOpen**, 433  
    **OS\_DirectoryRead**, 434  
    **OS\_DirectoryRewind**, 434  
    **OS\_mkdir**, 435  
    **OS\_rmdir**, 435  
**OSAL Dynamic Loader and Symbol APIs**, 476  
    **OS\_ModuleInfo**, 476  
    **OS\_ModuleLoad**, 476  
    **OS\_ModuleSymbolLookup**, 477  
    **OS\_ModuleUnload**, 478  
    **OS\_SymbolLookup**, 478  
    **OS\_SymbolTableDump**, 479  
**OSAL Error Info APIs**, 444  
    **OS\_GetErrorName**, 444  
    **OS\_StatusToInteger**, 444  
    **OS\_StatusToString**, 445  
**OSAL File Access Option Defines**, 446  
    **OS\_READ\_ONLY**, 446  
    **OS\_READ\_WRITE**, 446  
    **OS\_WRITE\_ONLY**, 446

OSAL File System Level APIs, 459  
 OS\_chkfs, 459  
 OS\_FileSysAddFixedMap, 460  
 OS\_FileSysStatVolume, 460  
 OS\_FS\_GetPhysDriveName, 461  
 OS\_GetFsInfo, 462  
 OS\_initfs, 462  
 OS\_mkfs, 463  
 OS\_mount, 463  
 OS\_rmfs, 464  
 OS\_TranslatePath, 464  
 OS\_unmount, 465  
 OSAL Heap APIs, 467  
 OS\_HeapGetInfo, 467  
 OSAL Message Queue APIs, 487  
 OS\_QueueCreate, 487  
 OS\_QueueDelete, 488  
 OS\_QueueGet, 488  
 OS\_QueueGetIdByName, 489  
 OS\_QueueGetInfo, 489  
 OS\_QueuePut, 490  
 OSAL Mutex APIs, 480  
 OS\_MutSemCreate, 480  
 OS\_MutSemDelete, 480  
 OS\_MutSemGetIdByName, 481  
 OS\_MutSemGetInfo, 481  
 OS\_MutSemGive, 482  
 OS\_MutSemTake, 482  
 OSAL Network ID APIs, 484  
 OS\_NetworkGetHostName, 484  
 OS\_NetworkGetID, 484  
 OSAL Object ID Utility APIs, 471  
 OS\_ConvertToArrayIndex, 471  
 OS\_ForEachObject, 472  
 OS\_ForEachObjectOfType, 472  
 OS\_GetResourceName, 472  
 OS\_IdentifyObject, 473  
 OS\_ObjectIdDefined, 473  
 OS\_ObjectIdEqual, 474  
 OS\_ObjectIdFromInteger, 474  
 OS\_ObjectIdToArrayIndex, 474  
 OS\_ObjectIdToInteger, 475  
 OSAL Object Type Defines, 468  
 OS\_OBJECT\_TYPE\_OS\_BINSEM, 468  
 OS\_OBJECT\_TYPE\_OS\_CONDVAR, 468  
 OS\_OBJECT\_TYPE\_OS\_CONSOLE, 468  
 OS\_OBJECT\_TYPE\_OS\_COUNTSEM, 469  
 OS\_OBJECT\_TYPE\_OS\_DIR, 469  
 OS\_OBJECT\_TYPE\_OS\_FILESYS, 469  
 OS\_OBJECT\_TYPE\_OS\_MODULE, 469  
 OS\_OBJECT\_TYPE\_OS\_MUTEX, 469  
 OS\_OBJECT\_TYPE\_OS\_QUEUE, 469  
 OS\_OBJECT\_TYPE\_OS\_STREAM, 469  
 OS\_OBJECT\_TYPE\_OS\_TASK, 469  
 OS\_OBJECT\_TYPE\_OS\_TIMEBASE, 469  
 OS\_OBJECT\_TYPE\_OS\_TIMECB, 469  
 OS\_OBJECT\_TYPE\_UNDEFINED, 470  
 OS\_OBJECT\_TYPE\_USER, 470  
 OSAL Printf APIs, 486  
 OS\_printf, 486  
 OS\_printf\_disable, 486  
 OS\_printf\_enable, 486  
 OSAL Real Time Clock APIs, 408  
 OS\_GetLocalTime, 409  
 OS\_SetLocalTime, 409  
 OS\_TimeAdd, 410  
 OS\_TimeAssembleFromMicroseconds, 410  
 OS\_TimeAssembleFromMilliseconds, 410  
 OS\_TimeAssembleFromNanoseconds, 411  
 OS\_TimeAssembleFromSubseconds, 411  
 OS\_TimeFromTotalMicroseconds, 412  
 OS\_TimeFromTotalMilliseconds, 412  
 OS\_TimeFromTotalNanoseconds, 413  
 OS\_TimeFromTotalSeconds, 413  
 OS\_TimeGetFractionalPart, 413  
 OS\_TimeGetMicrosecondsPart, 414  
 OS\_TimeGetMillisecondsPart, 414  
 OS\_TimeGetNanosecondsPart, 415  
 OS\_TimeGetSubsecondsPart, 416  
 OS\_TimeGetTotalMicroseconds, 416  
 OS\_TimeGetTotalMilliseconds, 417  
 OS\_TimeGetTotalNanoseconds, 417  
 OS\_TimeGetTotalSeconds, 418  
 OS\_TimeSubtract, 418  
 OSAL Reference Point For Seek Offset Defines, 447  
 OS\_SEEK\_CUR, 447  
 OS\_SEEK\_END, 447  
 OS\_SEEK\_SET, 447  
 OSAL Return Code Defines, 437  
 OS\_ERR\_BAD\_ADDRESS, 439  
 OS\_ERR\_FILE, 439  
 OS\_ERR\_INCORRECT\_OBJ\_STATE, 439  
 OS\_ERR\_INCORRECT\_OBJ\_TYPE, 439  
 OS\_ERR\_INVALID\_ARGUMENT, 439  
 OS\_ERR\_INVALID\_ID, 439  
 OS\_ERR\_INVALID\_PRIORITY, 440  
 OS\_ERR\_INVALID\_SIZE, 440  
 OS\_ERR\_NAME\_NOT\_FOUND, 440  
 OS\_ERR\_NAME\_TAKEN, 440  
 OS\_ERR\_NAME\_TOO\_LONG, 440  
 OS\_ERR\_NO\_FREE\_IDS, 440  
 OS\_ERR\_NOT\_IMPLEMENTED, 440  
 OS\_ERR\_OBJECT\_IN\_USE, 440  
 OS\_ERR\_OPERATION\_NOT\_SUPPORTED, 440  
 OS\_ERR\_OUTPUT\_TOO\_LARGE, 440  
 OS\_ERR\_SEM\_NOT\_FULL, 441  
 OS\_ERR\_STREAM\_DISCONNECTED, 441  
 OS\_ERROR, 441

OS\_ERROR\_ADDRESS\_MISALIGNED, 441  
OS\_ERROR\_TIMEOUT, 441  
OS\_FS\_ERR\_DEVICE\_NOT\_FREE, 441  
OS\_FS\_ERR\_DRIVE\_NOT\_CREATED, 441  
OS\_FS\_ERR\_NAME\_TOO\_LONG, 441  
OS\_FS\_ERR\_PATH\_INVALID, 441  
OS\_FS\_ERR\_PATH\_TOO\_LONG, 441  
OS\_INVALID\_INT\_NUM, 442  
OS\_INVALID\_POINTER, 442  
OS\_INVALID\_SEM\_VALUE, 442  
OS\_QUEUE\_EMPTY, 442  
OS\_QUEUE\_FULL, 442  
OS\_QUEUE\_ID\_ERROR, 442  
OS\_QUEUE\_INVALID\_SIZE, 442  
OS\_QUEUE\_TIMEOUT, 442  
OS\_SEM\_FAILURE, 442  
OS\_SEM\_TIMEOUT, 442  
OS\_SUCCESS, 443  
OS\_TIMER\_ERR\_INTERNAL, 443  
OS\_TIMER\_ERR\_INVALID\_ARGS, 443  
OS\_TIMER\_ERR\_TIMER\_ID, 443  
OS\_TIMER\_ERR\_UNAVAILABLE, 443  
OSAL Select APIs, 491  
  OS\_SelectFdAdd, 491  
  OS\_SelectFdClear, 491  
  OS\_SelectFdlsSet, 492  
  OS\_SelectFdZero, 492  
  OS\_SelectMultiple, 493  
  OS\_SelectSingle, 494  
OSAL Semaphore State Defines, 401  
  OS\_SEM\_EMPTY, 401  
  OS\_SEM\_FULL, 401  
OSAL Shell APIs, 495  
  OS\_ShellOutputToFile, 495  
OSAL Socket Address APIs, 496  
  OS\_SocketAddrFromString, 496  
  OS\_SocketAddrGetPort, 497  
  OS\_SocketAddrInit, 497  
  OS\_SocketAddrSetPort, 498  
  OS\_SocketAddrToString, 498  
OSAL Socket Management APIs, 500  
  OS\_SocketAccept, 500  
  OS\_SocketBind, 501  
  OS\_SocketBindAddress, 502  
  OS\_SocketConnect, 502  
  OS\_SocketGetIdByName, 503  
  OS\_SocketGetInfo, 503  
  OS\_SocketListen, 504  
  OS\_SocketOpen, 504  
  OS\_SocketRecvFrom, 505  
  OS\_SocketSendTo, 505  
  OS\_SocketShutdown, 506  
OSAL Standard File APIs, 448  
  OS\_chmod, 448  
            OS\_close, 449  
            OS\_CloseAllFiles, 449  
            OS\_CloseFileByName, 450  
            OS\_cp, 450  
            OS\_FDGetInfo, 451  
            OS\_FileOpenCheck, 451  
            OS\_Iseek, 452  
            OS\_mv, 452  
            OS\_OpenCreate, 453  
            OS\_read, 454  
            OS\_remove, 454  
            OS\_rename, 455  
            OS\_stat, 455  
            OS\_TimedRead, 456  
            OS\_TimedWrite, 457  
            OS\_write, 457  
OSAL Task APIs, 508  
  OS\_TaskCreate, 508  
  OS\_TaskDelay, 509  
  OS\_TaskDelete, 509  
  OS\_TaskExit, 510  
  OS\_TaskFindIdBySystemData, 510  
  OS\_TaskGetId, 511  
  OS\_TaskGetIdByName, 511  
  OS\_TaskGetInfo, 511  
  OS\_TaskInstallDeleteHandler, 512  
  OS\_TaskSetPriority, 512  
OSAL Time Base APIs, 514  
  OS\_TimeBaseCreate, 514  
  OS\_TimeBaseDelete, 515  
  OS\_TimeBaseGetFreeRun, 515  
  OS\_TimeBaseGetIdByName, 516  
  OS\_TimeBaseGetInfo, 517  
  OS\_TimeBaseSet, 517  
OSAL Timer APIs, 519  
  OS\_TimerAdd, 519  
  OS\_TimerCreate, 520  
  OS\_TimerDelete, 521  
  OS\_TimerGetIdByName, 522  
  OS\_TimerGetInfo, 522  
  OS\_TimerSet, 523  
osal/docs/src/osal\_frontpage.dox, 1315  
osal/docs/src/osal\_fs.dox, 1315  
osal/docs/src/osal\_timer.dox, 1315  
osal/src/os/inc/common\_types.h, 1315  
osal/src/os/inc/osapi-binsem.h, 1320  
osal/src/os/inc/osapi-bsp.h, 1321  
osal/src/os/inc/osapi-clock.h, 1321  
osal/src/os/inc/osapi-common.h, 1323  
osal/src/os/inc/osapi-condvar.h, 1325  
osal/src/os/inc/osapi-constants.h, 1326  
osal/src/os/inc/osapi-countsem.h, 1327  
osal/src/os/inc/osapi-dir.h, 1327  
osal/src/os/inc/osapi-error.h, 1328

osal/src/os/inc/osapi-file.h, 1331  
 osal/src/os/inc/osapi-filesystem.h, 1334  
 osal/src/os/inc/osapi-heap.h, 1336  
 osal/src/os/inc/osapi-idmap.h, 1336  
 osal/src/os/inc/osapi-macros.h, 1338  
 osal/src/os/inc/osapi-module.h, 1339  
 osal/src/os/inc/osapi-mutex.h, 1341  
 osal/src/os/inc/osapi-network.h, 1341  
 osal/src/os/inc/osapi-printf.h, 1342  
 osal/src/os/inc/osapi-queue.h, 1342  
 osal/src/os/inc/osapi-select.h, 1343  
 osal/src/os/inc/osapi-shell.h, 1344  
 osal/src/os/inc/osapi-sockets.h, 1344  
 osal/src/os/inc/osapi-task.h, 1347  
 osal/src/os/inc/osapi-timebase.h, 1349  
 osal/src/os/inc/osapi-timer.h, 1350  
 osal/src/os/inc/osapi-version.h, 1351  
 osal/src/os/inc/osapi.h, 1354  
**OSAL\_API\_VERSION**  
 osapi-version.h, 1353  
**OSAL\_BLOCKCOUNT\_C**  
 common\_types.h, 1317  
**osal\_blockcount\_t**  
 common\_types.h, 1318  
**OSAL\_CONFIG\_CONSOLE\_ASYNC**  
 osconfig.h, 913  
**OSAL\_CONFIG\_INCLUDE\_DYNAMIC\_LOADER**  
 osconfig.h, 913  
**OSAL\_CONFIG\_INCLUDE\_NETWORK**  
 osconfig.h, 913  
**OSAL\_CONFIG\_INCLUDE\_STATIC\_LOADER**  
 osconfig.h, 913  
**osal\_id\_t**  
 common\_types.h, 1318  
**OSAL\_INDEX\_C**  
 common\_types.h, 1317  
**osal\_index\_t**  
 common\_types.h, 1318  
**OSAL\_OBJTYPE\_C**  
 common\_types.h, 1317  
**osal\_objtype\_t**  
 common\_types.h, 1319  
**OSAL\_PRIORITY\_C**  
 osapi-task.h, 1348  
**osal\_priority\_t**  
 osapi-task.h, 1348  
**OSAL\_SIZE\_C**  
 common\_types.h, 1317  
**OSAL\_STACKPTR\_C**  
 osapi-task.h, 1348  
**osal\_stackptr\_t**  
 osapi-task.h, 1348  
**OSAL\_STATUS\_C**  
 common\_types.h, 1317

osal\_status\_t  
 common\_types.h, 1319  
**osal\_task**  
 osapi-task.h, 1349  
**OSAL\_TASK\_STACK\_ALLOCATE**  
 osapi-task.h, 1348  
**OSALMajorVersion**  
 CFE\_ES\_HousekeepingTlm\_Payload, 541  
**OSALMinorVersion**  
 CFE\_ES\_HousekeepingTlm\_Payload, 542  
**OSALMissionRevision**  
 CFE\_ES\_HousekeepingTlm\_Payload, 542  
**OSALRevision**  
 CFE\_ES\_HousekeepingTlm\_Payload, 542  
**osapi-clock.h**  
 OS\_TIME\_TICK\_RESOLUTION\_NS, 1323  
 OS\_TIME TICKS\_PER\_MSEC, 1323  
 OS\_TIME TICKS\_PER\_SECOND, 1323  
 OS\_TIME TICKS\_PER\_USEC, 1323  
**osapi-common.h**  
 OS\_EVENT\_MAX, 1325  
 OS\_EVENT\_RESERVED, 1324  
 OS\_EVENT\_RESOURCE\_ALLOCATED, 1324  
 OS\_EVENT\_RESOURCE\_CREATED, 1325  
 OS\_EVENT\_RESOURCE\_DELETED, 1325  
 OS\_Event\_t, 1324  
 OS\_EVENT\_TASK\_STARTUP, 1325  
 OS\_EventHandler\_t, 1324  
**osapi-constants.h**  
 OS\_CHECK, 1326  
 OS\_MAX\_LOCAL\_PATH\_LEN, 1326  
 OS\_OBJECT\_CREATOR\_ANY, 1326  
 OS\_OBJECT\_ID\_UNDEFINED, 1326  
 OS\_PEND, 1327  
**osapi-dir.h**  
 OS\_DIRENTRY\_NAME, 1328  
**osapi-error.h**  
 os\_err\_name\_t, 1331  
 OS\_ERROR\_NAME\_LENGTH, 1331  
 OS\_STATUS\_STRING\_LENGTH, 1331  
 os\_status\_string\_t, 1331  
**osapi-file.h**  
 OS\_FILE\_FLAG\_CREATE, 1334  
 OS\_FILE\_FLAG\_NONE, 1334  
 OS\_file\_flag\_t, 1334  
 OS\_FILE\_FLAG\_TRUNCATE, 1334  
 OS\_FILESTAT\_EXEC, 1333  
 OS\_FILESTAT\_ISDIR, 1333  
 OS\_FILESTAT\_MODE, 1333  
 OS\_FILESTAT\_MODE\_DIR, 1334  
 OS\_FILESTAT\_MODE\_EXEC, 1334  
 OS\_FILESTAT\_MODE\_READ, 1334  
 OS\_FILESTAT\_MODE\_WRITE, 1334  
 OS\_FILESTAT\_READ, 1333

OS\_FILESTAT\_SIZE, 1333  
OS\_FILESTAT\_TIME, 1333  
OS\_FILESTAT\_WRITE, 1334  
osapi-fs.h  
  OS\_CHK\_ONLY, 1335  
  OS\_REPAIR, 1335  
osapi-idmap.h  
  OS\_OBJECT\_INDEX\_MASK, 1337  
  OS\_OBJECT\_TYPE\_SHIFT, 1337  
osapi-macros.h  
  ARGCHECK, 1338  
  BUGCHECK, 1338  
  BUGCHECK\_VOID, 1339  
  BUGREPORT, 1339  
  LENGTHCHECK, 1339  
osapi-module.h  
  OS\_MODULE\_FLAG\_GLOBAL\_SYMBOLS, 1340  
  OS\_MODULE\_FLAG\_LOCAL\_SYMBOLS, 1340  
osapi-select.h  
  OS\_STREAM\_STATE\_BOUND, 1344  
  OS\_STREAM\_STATE\_CONNECTED, 1344  
  OS\_STREAM\_STATE\_LISTENING, 1344  
  OS\_STREAM\_STATE\_READABLE, 1344  
  OS\_STREAM\_STATE\_WRITABLE, 1344  
  OS\_StreamState\_t, 1343  
osapi-sockets.h  
  OS\_SOCKADDR\_MAX\_LEN, 1346  
  OS\_SocketDomain\_INET, 1346  
  OS\_SocketDomain\_INET6, 1346  
  OS\_SocketDomain\_INVALID, 1346  
  OS\_SocketDomain\_MAX, 1346  
  OS\_SocketDomain\_t, 1346  
  OS\_SocketShutdownMode\_NONE, 1346  
  OS\_SocketShutdownMode\_SHUT\_READ, 1346  
  OS\_SocketShutdownMode\_SHUT\_READWRITE, 1346  
  OS\_SocketShutdownMode\_SHUT\_WRITE, 1346  
  OS\_SocketShutdownMode\_t, 1346  
  OS\_SocketType\_DATAGRAM, 1346  
  OS\_SocketType\_INVALID, 1346  
  OS\_SocketType\_MAX, 1346  
  OS\_SocketType\_STREAM, 1346  
  OS\_SocketType\_t, 1346  
osapi-task.h  
  OS\_FP\_ENABLED, 1348  
  OS\_MAX\_TASK\_PRIORITY, 1348  
  OSAL\_PRIORITY\_C, 1348  
  osal\_priority\_t, 1348  
  OSAL\_STACKPTR\_C, 1348  
  osal\_stackptr\_t, 1348  
  osal\_task, 1349  
  OSAL\_TASK\_STACK\_ALLOCATE, 1348  
osapi-timebase.h  
  OS\_TimerSync\_t, 1350  
osapi-timer.h  
  OS\_TimerCallback\_t, 1350  
osapi-version.h  
  OS\_BUILD\_BASELINE, 1351  
  OS\_BUILD\_NUMBER, 1352  
  OS\_GetBuildNumber, 1353  
  OS\_GetVersionCodeName, 1353  
  OS\_GetVersionNumber, 1353  
  OS\_GetVersionString, 1354  
  OS\_MAJOR\_VERSION, 1352  
  OS\_MINOR\_VERSION, 1352  
  OS\_MISSION\_REV, 1352  
  OS\_REVISION, 1352  
  OS\_STR, 1352  
  OS\_STR\_HELPER, 1352  
  OS\_VERSION, 1352  
  OS\_VERSION\_CODENAME, 1352  
  OS\_VERSION\_STRING, 1353  
  OSAL\_API\_VERSION, 1353  
OSBaseline  
  CS\_HkPacket\_Payload\_t, 672  
OSCodeSeg  
  CS\_AppData\_t, 660  
osconfig.h  
  OS\_ADD\_TASK\_FLAGS, 909  
  OS\_BUFFER\_MSG\_DEPTH, 909  
  OS\_BUFFER\_SIZE, 909  
  OS\_FS\_DEV\_NAME\_LEN, 909  
  OS\_FS\_PHYS\_NAME\_LEN, 909  
  OS\_FS\_VOL\_NAME\_LEN, 909  
  OS\_MAX\_API\_NAME, 909  
  OS\_MAX\_BIN\_SEMAPHORES, 910  
  OS\_MAX\_CMD\_LEN, 910  
  OS\_MAX\_CONDVARS, 910  
  OS\_MAX\_CONSOLES, 910  
  OS\_MAX\_COUNT\_SEMAPHORES, 910  
  OS\_MAX\_FILE\_NAME, 910  
  OS\_MAX\_FILE\_SYSTEMS, 910  
  OS\_MAX\_MODULES, 911  
  OS\_MAX\_MUTEXES, 911  
  OS\_MAX\_NUM\_OPEN\_DIRS, 911  
  OS\_MAX\_NUM\_OPEN\_FILES, 911  
  OS\_MAX\_PATH\_LEN, 911  
  OS\_MAX\_QUEUES, 911  
  OS\_MAX\_SYM\_LEN, 911  
  OS\_MAX\_TASKS, 912  
  OS\_MAX\_TIMEBASES, 912  
  OS\_MAX\_TIMERS, 912  
  OS\_MODULE\_FILE\_EXTENSION, 912  
  OS\_PRINTF\_CONSOLE\_NAME, 912  
  OS\_QUEUE\_MAX\_DEPTH, 912  
  OS\_SHELL\_CMD\_INPUT\_FILE\_NAME, 912  
  OS SOCKADDR\_MAX\_LEN, 912  
  OS\_UTILITYTASK\_PRIORITY, 913

OS.UtilityTask\_Stack\_Size, 913  
 OSAL\_Config\_Console\_Async, 913  
 OSAL\_Config\_Include\_Dynamic\_Loader, 913  
 OSAL\_Config\_Include\_Network, 913  
 OSAL\_Config\_Include\_Static\_Loader, 913  
**OSCSErrCounter**  
 CS\_HkPacket\_Payload\_t, 672  
**OSCSState**  
 CS\_HkPacket\_Payload\_t, 672  
**OutputPort**  
 CFE\_EVS\_HousekeepingTlm\_Payload, 575  
**OwnerAppName**  
 CFE\_TBL\_TblRegPacket\_Payload, 631  
**PacketID**  
 CFE\_EVS\_LongEventTlm\_Payload, 577  
 CFE\_EVS\_ShortEventTlm\_Payload, 583  
**Parameter**  
 CFE\_TBL\_NotifyCmd\_Payload, 626  
**PassCounter**  
 CS\_HkPacket\_Payload\_t, 672  
**Path**  
 OS\_file\_prop\_t, 686  
**Payload**  
 CFE\_ES\_AppNameCmd, 530  
 CFE\_ES\_DeleteCDSCmd, 534  
 CFE\_ES\_DumpCDSRegistryCmd, 535  
 CFE\_ES\_FileNameCmd, 536  
 CFE\_ES\_HousekeepingTlm, 537  
 CFE\_ES\_MemStatsTlm, 547  
 CFE\_ES\_OneAppTlm, 548  
 CFE\_ES\_OverWriteSysLogCmd, 550  
 CFE\_ES\_ReloadAppCmd, 552  
 CFE\_ES\_RestartCmd, 553  
 CFE\_ES\_SendMemPoolStatsCmd, 554  
 CFE\_ES\_SetMaxPRCountCmd, 555  
 CFE\_ES\_SetPerfFilterMaskCmd, 556  
 CFE\_ES\_SetPerfTriggerMaskCmd, 557  
 CFE\_ES\_StartApp, 559  
 CFE\_ES\_StartPerfDataCmd, 561  
 CFE\_ES\_StopPerfDataCmd, 562  
 CFE\_EVS\_AppNameBitMaskCmd, 565  
 CFE\_EVS\_AppNameCmd, 566  
 CFE\_EVS\_AppNameEventIDCmd, 567  
 CFE\_EVS\_AppNameEventIDMaskCmd, 568  
 CFE\_EVS\_BitMaskCmd, 571  
 CFE\_EVS\_HousekeepingTlm, 572  
 CFE\_EVS\_LongEventTlm, 577  
 CFE\_EVS\_SetEventFormatModeCmd, 581  
 CFE\_EVS\_SetLogModeCmd, 582  
 CFE\_EVS\_ShortEventTlm, 582  
 CFE\_EVS\_WriteAppDataFileCmd, 584  
 CFE\_EVS\_WriteLogDataFileCmd, 584  
 CFE\_SB\_AllSubscriptionsTlm, 587  
 CFE\_SB\_HousekeepingTlm, 589  
 CFE\_SB\_RouteCmd, 599  
 CFE\_SB\_SingleSubscriptionTlm, 601  
 CFE\_SB\_StatsTlm, 603  
 CFE\_SB\_WriteFileInfoCmd, 608  
 CFE\_TBL\_AbortLoadCmd, 609  
 CFE\_TBL\_ActivateCmd, 610  
 CFE\_TBL\_DeleteCDSCmd, 612  
 CFE\_TBL\_DumpCmd, 612  
 CFE\_TBL\_DumpRegistryCmd, 614  
 CFE\_TBL\_HousekeepingTlm, 617  
 CFE\_TBL\_LoadCmd, 624  
 CFE\_TBL\_NotifyCmd, 626  
 CFE\_TBL\_SendRegistryCmd, 627  
 CFE\_TBL\_TableRegistryTlm, 628  
 CFE\_TBL\_ValidateCmd, 632  
 CFE\_TIME\_DiagnosticTlm, 633  
 CFE\_TIME\_HousekeepingTlm, 642  
 CFE\_TIME\_OneHzAdjustmentCmd, 646  
 CFE\_TIME\_SetLeapSecondsCmd, 647  
 CFE\_TIME\_SetSignalCmd, 648  
 CFE\_TIME\_SetSourceCmd, 649  
 CFE\_TIME\_SetStateCmd, 649  
 CFE\_TIME\_TimeCmd, 652  
 CFE\_TIME\_ToneDataCmd, 653  
 CS\_AppNameCmd\_t, 663  
 CS\_EntryCmd\_t, 666  
 CS\_GetEntryIDCmd\_t, 667  
 CS\_HkPacket\_t, 673  
 CS\_OneShotCmd\_t, 676  
 CS\_TableNameCmd\_t, 683  
**PeakMemInUse**  
 CFE\_SB\_StatsTlm\_Payload, 605  
**PeakMsgIdsInUse**  
 CFE\_SB\_StatsTlm\_Payload, 605  
**PeakPipesInUse**  
 CFE\_SB\_StatsTlm\_Payload, 605  
**PeakQueueDepth**  
 CFE\_SB\_PipeDepthStats, 596  
 CFE\_SB\_PipeInfoEntry, 597  
**PeakSBBuffersInUse**  
 CFE\_SB\_StatsTlm\_Payload, 606  
**PeakSubscriptionsInUse**  
 CFE\_SB\_StatsTlm\_Payload, 606  
**PerfDataCount**  
 CFE\_ES\_HousekeepingTlm\_Payload, 542  
**PerfDataEnd**  
 CFE\_ES\_HousekeepingTlm\_Payload, 542  
**PerfDataStart**  
 CFE\_ES\_HousekeepingTlm\_Payload, 542  
**PerfDataToWrite**  
 CFE\_ES\_HousekeepingTlm\_Payload, 542  
**PerfFilterMask**

CFE\_ES\_HousekeepingTlm\_Payload, 543  
PerfMode  
    CFE\_ES\_HousekeepingTlm\_Payload, 543  
PerfState  
    CFE\_ES\_HousekeepingTlm\_Payload, 543  
PerfTriggerCount  
    CFE\_ES\_HousekeepingTlm\_Payload, 543  
PerfTriggerMask  
    CFE\_ES\_HousekeepingTlm\_Payload, 543  
Pipe  
    CFE\_SB\_RouteCmd\_Payload, 600  
    CFE\_SB\_SingleSubscriptionTlm\_Payload, 602  
    CFE\_SB\_SubEntries, 607  
PipeDepth  
    CS\_AppData\_t, 660  
PipeDepthStats  
    CFE\_SB\_StatsTlm\_Payload, 606  
Pipeld  
    CFE\_SB\_PipeDepthStats, 596  
    CFE\_SB\_PipeInfoEntry, 597  
    CFE\_SB\_RoutingFileEntry, 601  
PipeName  
    CFE\_SB\_PipeInfoEntry, 597  
    CFE\_SB\_RoutingFileEntry, 601  
    CS\_AppData\_t, 660  
PipeOptsErrorCounter  
    CFE\_SB\_HousekeepingTlm\_Payload, 592  
PipeOverflowErrorCounter  
    CFE\_SB\_HousekeepingTlm\_Payload, 592  
PipesInUse  
    CFE\_SB\_StatsTlm\_Payload, 606  
PktSegment  
    CFE\_SB\_AllSubscriptionsTlm\_Payload, 588  
PoolHandle  
    CFE\_ES\_PoolStatsTlm\_Payload, 551  
    CFE\_ES\_SendMemPoolStatsCmd\_Payload, 554  
PoolSize  
    CFE\_ES\_MemPoolStats, 547  
PoolStats  
    CFE\_ES\_PoolStatsTlm\_Payload, 551  
Priority  
    CFE\_ES\_AppInfo, 529  
    CFE\_ES\_StartAppCmd\_Payload, 560  
    CFE\_ES\_TaskInfo, 563  
    CFE\_SB\_Qos\_t, 598  
priority  
    OS\_task\_prop\_t, 695  
ProcessorID  
    CFE\_EVS\_PacketID, 579  
    CFE\_FS\_Header, 587  
ProcessorResets  
    CFE\_ES\_HousekeepingTlm\_Payload, 543  
psp/fsw/inc/cfe\_psp.h, 1355  
psp/fsw/inc/cfe\_psp\_error.h, 1368  
PSPMajorVersion  
    CFE\_ES\_HousekeepingTlm\_Payload, 543  
PSPMinorVersion  
    CFE\_ES\_HousekeepingTlm\_Payload, 544  
PSPMissionRevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 544  
PSPRevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 544  
Ptr  
    CFE\_ES\_PoolAlign, 551  
Qos  
    CFE\_SB\_SingleSubscriptionTlm\_Payload, 602  
    CFE\_SB\_SubEntries, 607  
RecomputeAppEntryPtr  
    CS\_AppData\_t, 660  
RecomputeEepromMemoryEntryPtr  
    CS\_AppData\_t, 660  
RecomputeInProgress  
    CS\_HkPacket\_Payload\_t, 672  
RecomputeTablesEntryPtr  
    CS\_AppData\_t, 660  
RegisteredCoreApps  
    CFE\_ES\_HousekeepingTlm\_Payload, 544  
RegisteredExternalApps  
    CFE\_ES\_HousekeepingTlm\_Payload, 544  
RegisteredLibs  
    CFE\_ES\_HousekeepingTlm\_Payload, 544  
RegisteredTasks  
    CFE\_ES\_HousekeepingTlm\_Payload, 544  
Reliability  
    CFE\_SB\_Qos\_t, 598  
ResAppTableHandle  
    CS\_AppData\_t, 660  
ResAppTblPtr  
    CS\_AppData\_t, 661  
ResEepromTableHandle  
    CS\_AppData\_t, 661  
ResEepromTblPtr  
    CS\_AppData\_t, 661  
Reserved  
    CFE\_TBL\_File\_Hdr, 615  
ResetSubtype  
    CFE\_ES\_HousekeepingTlm\_Payload, 545  
ResetType  
    CFE\_ES\_HousekeepingTlm\_Payload, 545  
ResMemoryTableHandle  
    CS\_AppData\_t, 661  
ResMemoryTblPtr  
    CS\_AppData\_t, 661  
ResourceId  
    CFE\_ES\_AppInfo, 529  
ResTablesTableHandle  
    CS\_AppData\_t, 661

ResTablesTblPtr  
     CS\_AppData\_t, 661

RestartType  
     CFE\_ES\_RestartCmd\_Payload, 553

RunStatus  
     CS\_AppData\_t, 662

sample\_perfids.h  
     CFE\_MISSION\_ES\_MAIN\_PERF\_ID, 969  
     CFE\_MISSION\_ES\_PERF\_EXIT\_BIT, 969  
     CFE\_MISSION\_EVS\_MAIN\_PERF\_ID, 969  
     CFE\_MISSION\_SB\_MAIN\_PERF\_ID, 969  
     CFE\_MISSION\_SB\_MSG\_LIM\_PERF\_ID, 969  
     CFE\_MISSION\_SB\_PIPE\_OFLOW\_PERF\_ID, 969  
     CFE\_MISSION\_TBL\_MAIN\_PERF\_ID, 969  
     CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID,  
         969  
     CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID,  
         970  
     CFE\_MISSION\_TIME\_MAIN\_PERF\_ID, 970  
     CFE\_MISSION\_TIME\_SEDMET\_PERF\_ID, 970  
     CFE\_MISSION\_TIME\_TONE1HZISR\_PERF\_ID,  
         970  
     CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID,  
         970

SBBuffersInUse  
     CFE\_SB\_StatsTlm\_Payload, 606

Seconds  
     CFE\_TIME\_OneHzAdjustmentCmd\_Payload, 647  
     CFE\_TIME\_SysTime, 652  
     CFE\_TIME\_TimeCmd\_Payload, 653

Seconds1HzAdj  
     CFE\_TIME\_HousekeepingTlm\_Payload, 644

SecondsDelay  
     CFE\_TIME\_HousekeepingTlm\_Payload, 644

SecondsMET  
     CFE\_TIME\_HousekeepingTlm\_Payload, 644

SecondsSTCF  
     CFE\_TIME\_HousekeepingTlm\_Payload, 644

SendErrors  
     CFE\_SB\_PipeInfoEntry, 597

Sequence  
     CCSDS\_PrimaryHeader, 526

ServerFlyState  
     CFE\_TIME\_DiagnosticTlm\_Payload, 639

Size  
     CFE\_ES\_CDSRegDumpRec, 533  
     CFE\_TBL\_Info, 623  
     CFE\_TBL\_TblRegPacket\_Payload, 631  
     CS\_OneShotCmd\_Payload\_t, 675

SIZE\_BYTEx  
     cfe\_psp.h, 1360

SIZE\_HALFx  
     cfe\_psp.h, 1360

SIZE\_WORD  
     cfe\_psp.h, 1360

SpacecraftID  
     CFE\_EVS\_PacketID, 579  
     CFE\_FS\_Header, 587

Spare  
     CFE\_ES\_TaskInfo, 563  
     CFE\_EVS\_AppNameBitMaskCmd\_Payload, 565  
     CFE\_EVS\_BitMaskCmd\_Payload, 572  
     CFE\_EVS\_SetEventFormatCode\_Payload, 580  
     CFE\_EVS\_SetLogMode\_Payload, 581  
     CFE\_SB\_PipeDepthStats, 596  
     CFE\_SB\_PipeInfoEntry, 598  
     CFE\_SB\_RouteCmd\_Payload, 600

Spare1  
     CFE\_EVS\_HousekeepingTlm\_Payload, 575  
     CFE\_EVS\_LongEventTlm\_Payload, 578

Spare2  
     CFE\_EVS\_HousekeepingTlm\_Payload, 575  
     CFE\_EVS\_LongEventTlm\_Payload, 578

Spare2Align  
     CFE\_SB\_HousekeepingTlm\_Payload, 592

Spare3  
     CFE\_EVS\_HousekeepingTlm\_Payload, 575

stack\_size  
     OS\_task\_prop\_t, 695

StackSize  
     CFE\_ES\_AppInfo, 529  
     CFE\_ES\_StartAppCmd\_Payload, 560  
     CFE\_ES\_TaskInfo, 563

start\_time  
     OS\_timer\_prop\_t, 698

StartAddress  
     CFE\_ES\_AppInfo, 530  
     CS\_Def\_EepromMemory\_Table\_Entry\_t, 664  
     CS\_Res\_App\_Table\_Entry\_t, 677  
     CS\_Res\_EepromMemory\_Table\_Entry\_t, 679  
     CS\_Res\_Tables\_Table\_Entry\_t, 681

State  
     CFE\_SB\_RoutingFileEntry, 601  
     CS\_Def\_App\_Table\_Entry\_t, 663  
     CS\_Def\_EepromMemory\_Table\_Entry\_t, 664  
     CS\_Def\_Tables\_Table\_Entry\_t, 665  
     CS\_Res\_App\_Table\_Entry\_t, 677  
     CS\_Res\_EepromMemory\_Table\_Entry\_t, 679  
     CS\_Res\_Tables\_Table\_Entry\_t, 681

StreamId  
     CCSDS\_PrimaryHeader, 526

SubscribeErrorCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 592

SubscriptionsInUse  
     CFE\_SB\_StatsTlm\_Payload, 606

Subseconds  
     CFE\_TIME\_OneHzAdjustmentCmd\_Payload, 647

CFE\_TIME\_SysTime, 652  
Subsecs1HzAdj  
    CFE\_TIME\_HousekeepingTlm\_Payload, 644  
SubsecsDelay  
    CFE\_TIME\_HousekeepingTlm\_Payload, 644  
SubsecsMET  
    CFE\_TIME\_HousekeepingTlm\_Payload, 644  
SubsecsSTCF  
    CFE\_TIME\_HousekeepingTlm\_Payload, 645  
Subsystem  
    CCSDS\_ExtendedHeader, 525  
SubType  
    CFE\_FS\_Header, 587  
    CFE\_SB\_SingleSubscriptionTlm\_Payload, 602  
SuccessValCounter  
    CFE\_TBL\_HousekeepingTlm\_Payload, 621  
SysLogBytesUsed  
    CFE\_ES\_HousekeepingTlm\_Payload, 545  
SysLogEntries  
    CFE\_ES\_HousekeepingTlm\_Payload, 545  
SysLogMode  
    CFE\_ES\_HousekeepingTlm\_Payload, 545  
SysLogSize  
    CFE\_ES\_HousekeepingTlm\_Payload, 545  
SystemId  
    CCSDS\_ExtendedHeader, 525

Table  
    CFE\_ES\_CDSRegDumpRec, 533  
TableLoadedOnce  
    CFE\_TBL\_Info, 623  
    CFE\_TBL\_TblRegPacket\_Payload, 631  
TableName  
    CFE\_TBL\_AbortLoadCmd\_Payload, 609  
    CFE\_TBL\_ActivateCmd\_Payload, 610  
    CFE\_TBL\_DelCDSCmd\_Payload, 611  
    CFE\_TBL\_DumpCmd\_Payload, 613  
    CFE\_TBL\_File\_Hdr, 615  
    CFE\_TBL\_FileDef, 616  
    CFE\_TBL\_SendRegistryCmd\_Payload, 627  
    CFE\_TBL\_ValidateCmd\_Payload, 633  
TablesCSErrCounter  
    CS\_HkPacket\_Payload\_t, 673  
TablesCSState  
    CS\_HkPacket\_Payload\_t, 673  
TaskId  
    CFE\_ES\_TaskInfo, 563  
TaskName  
    CFE\_ES\_TaskInfo, 563  
TblHandle  
    CS\_Res\_Tables\_Table\_Entry\_t, 681  
TblResTablesTblPtr  
    CS\_AppData\_t, 662  
TelemetryHeader  
    CFE\_ES\_HousekeepingTlm, 537  
    CFE\_ES\_MemStatsTlm, 547  
    CFE\_ES\_OneAppTlm, 548  
    CFE\_EVS\_HousekeepingTlm, 573  
    CFE\_EVS\_LongEventTlm, 577  
    CFE\_EVS\_ShortEventTlm, 583  
    CFE\_SB\_AllSubscriptionsTlm, 588  
    CFE\_SB\_HousekeepingTlm, 589  
    CFE\_SB\_SingleSubscriptionTlm, 601  
    CFE\_SB\_StatsTlm, 603  
    CFE\_TBL\_HousekeepingTlm, 617  
    CFE\_TBL\_TableRegistryTlm, 628  
    CFE\_TIME\_DiagnosticTlm, 633  
    CFE\_TIME\_HousekeepingTlm, 642  
TempChecksumValue  
    CS\_Res\_App\_Table\_Entry\_t, 677  
    CS\_Res\_EepromMemory\_Table\_Entry\_t, 679  
    CS\_Res\_Tables\_Table\_Entry\_t, 681  
TgtFilename  
    CFE\_TBL\_FileDef, 617  
ticks  
    OS\_time\_t, 696  
TimeOfLastUpdate  
    CFE\_TBL\_Info, 623  
    CFE\_TBL\_TblRegPacket\_Payload, 631  
TimeSeconds  
    CFE\_FS\_Header, 587  
TimeSinceTone  
    CFE\_TIME\_DiagnosticTlm\_Payload, 639  
TimeSource  
    CFE\_TIME\_SourceCmd\_Payload, 650  
TimeSubSeconds  
    CFE\_FS\_Header, 587  
TlmHeader  
    CS\_HkPacket\_t, 674  
ToneDataCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 640  
ToneDataLatch  
    CFE\_TIME\_DiagnosticTlm\_Payload, 640  
ToneIntCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 640  
ToneIntErrorCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 640  
ToneMatchCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 640  
ToneMatchErrorCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 640  
ToneOverLimit  
    CFE\_TIME\_DiagnosticTlm\_Payload, 640  
ToneSignalCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 641  
ToneSignalLatch  
    CFE\_TIME\_DiagnosticTlm\_Payload, 641  
ToneSource

CFE\_TIME\_SignalCmd\_Payload, [650](#)  
ToneTaskCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, [641](#)  
ToneUnderLimit  
    CFE\_TIME\_DiagnosticTlm\_Payload, [641](#)  
total\_blocks  
    OS\_statvfs\_t, [695](#)  
TotalSegments  
    CFE\_SB\_AllSubscriptionsTlm\_Payload, [589](#)  
TriggerMask  
    CFE\_ES\_SetPerfTrigMaskCmd\_Payload, [558](#)  
TriggerMaskNum  
    CFE\_ES\_SetPerfTrigMaskCmd\_Payload, [558](#)  
TriggerMode  
    CFE\_ES\_StartPerfCmd\_Payload, [560](#)  
Type  
    CFE\_ES\_AppInfo, [530](#)

uint16  
    common\_types.h, [1319](#)  
uint32  
    common\_types.h, [1319](#)  
uint64  
    common\_types.h, [1319](#)  
uint8  
    common\_types.h, [1319](#)  
UnmarkedMem  
    CFE\_SB\_HousekeepingTlm\_Payload, [593](#)  
UnregisteredAppCounter  
    CFE\_EVS\_HousekeepingTlm\_Payload, [576](#)  
User  
    OS\_file\_prop\_t, [686](#)  
UserDefAddr  
    CFE\_TBL\_Info, [623](#)

valid  
    OS\_module\_address\_t, [689](#)  
ValidationCounter  
    CFE\_TBL\_HousekeepingTlm\_Payload, [621](#)  
ValidationFuncPtr  
    CFE\_TBL\_TblRegPacket\_Payload, [631](#)  
Value  
    CFE\_SB\_MsgId\_t, [594](#)  
value  
    OS\_bin\_sem\_prop\_t, [683](#)  
    OS\_count\_sem\_prop\_t, [684](#)  
VersionCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, [641](#)  
VirtualMET  
    CFE\_TIME\_DiagnosticTlm\_Payload, [641](#)