

## CF User's Guide

Generated by Doxygen 1.9.1

---

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

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

---

2.18.5 Effects of Processor Reset on Tables . . . . .	74
2.18.6 Frequently Asked Questions about Table Services . . . . .	74
2.19 cFE Table Services Commands . . . . .	76
2.20 cFE Table Services Telemetry . . . . .	76
2.21 cFE Table Services Configuration Parameters . . . . .	77
2.22 cFE Time Services Overview . . . . .	78
2.22.1 Time Components . . . . .	80
2.22.2 Time Structure . . . . .	81
2.22.3 Time Formats . . . . .	81
2.22.4 Time Configuration . . . . .	82
2.22.5 Time Format Selection . . . . .	86
2.22.6 Enabling Fake Tone Signal . . . . .	86
2.22.7 Selecting Tone and Data Ordering . . . . .	87
2.22.8 Specifying Tone and Data Window . . . . .	87
2.22.9 Specifying Time Server/Client . . . . .	87
2.22.10 Specifying Time Tone Byte Order . . . . .	88
2.22.11 Virtual MET . . . . .	88
2.22.12 Specifying Time Source . . . . .	88
2.22.13 Specifying Time Signal . . . . .	89
2.22.14 Time Services Paradigm(s) . . . . .	89
2.22.15 Flywheeling . . . . .	90
2.22.16 Time State . . . . .	91
2.22.17 Initialization . . . . .	91
2.22.18 Power-On Reset . . . . .	92
2.22.19 Processor Reset . . . . .	92
2.22.20 Normal Operation . . . . .	92
2.22.21 Client . . . . .	94
2.22.22 Server . . . . .	94
2.22.23 Setting Time . . . . .	95
2.22.24 Adjusting Time . . . . .	96
2.22.25 Setting MET . . . . .	96
2.22.26 Frequently Asked Questions about Time Services . . . . .	96
2.23 cFE Time Services Commands . . . . .	96
2.24 cFE Time Services Telemetry . . . . .	97
2.25 cFE Time Services Configuration Parameters . . . . .	98
2.26 cFE Event Message Cross Reference . . . . .	99
2.27 cFE Command Mnemonic Cross Reference . . . . .	99
2.28 cFE Telemetry Mnemonic Cross Reference . . . . .	103

<b>3 Glossary of Terms</b>	<b>114</b>
<b>4 cFE Application Programmer's Interface (API) Reference</b>	<b>115</b>
4.1 Executive Services API . . . . .	115
4.2 Events Services API . . . . .	117
4.3 File Services API . . . . .	117
4.4 Message API . . . . .	118
4.5 Resource ID API . . . . .	119
4.6 Software Bus Services API . . . . .	119
4.7 Table Services API . . . . .	120
4.8 Time Services API . . . . .	120
<b>5 Osal API Documentation</b>	<b>121</b>
5.1 OSAL Introduction . . . . .	123
5.2 File System Overview . . . . .	123
5.3 File Descriptors In Osal . . . . .	124
5.4 Timer Overview . . . . .	124
<b>6 cFE Mission Configuration Parameters</b>	<b>124</b>
<b>7 Module Index</b>	<b>125</b>
7.1 Modules . . . . .	125
<b>8 Data Structure Index</b>	<b>128</b>
8.1 Data Structures . . . . .	128
<b>9 File Index</b>	<b>144</b>
9.1 File List . . . . .	144
<b>10 Module Documentation</b>	<b>152</b>
10.1 CFS CFDP Event IDs . . . . .	152
10.1.1 Detailed Description . . . . .	158
10.1.2 Macro Definition Documentation . . . . .	158
10.2 CFS CFDP Command Codes . . . . .	193
10.2.1 Detailed Description . . . . .	194
10.2.2 Macro Definition Documentation . . . . .	194
10.3 CFS CFDP Platform Configuration . . . . .	209
10.3.1 Detailed Description . . . . .	211
10.3.2 Macro Definition Documentation . . . . .	211
10.4 CFS CFDP Mission Configuration . . . . .	217
10.4.1 Detailed Description . . . . .	218
10.4.2 Macro Definition Documentation . . . . .	218

---

10.5 CFS CFDP Version . . . . .	219
10.5.1 Detailed Description . . . . .	219
10.5.2 Macro Definition Documentation . . . . .	219
10.6 CFS CFDP Telemetry . . . . .	220
10.6.1 Detailed Description . . . . .	221
10.6.2 Typedef Documentation . . . . .	221
10.7 CFS CFDP Command Structures . . . . .	221
10.7.1 Detailed Description . . . . .	224
10.7.2 Typedef Documentation . . . . .	224
10.7.3 Enumeration Type Documentation . . . . .	227
10.8 CFS CFDP Command Message IDs . . . . .	228
10.8.1 Detailed Description . . . . .	228
10.8.2 Macro Definition Documentation . . . . .	228
10.9 CFS CFDP Telemetry Message IDs . . . . .	228
10.9.1 Detailed Description . . . . .	229
10.9.2 Macro Definition Documentation . . . . .	229
10.10 CFS CFDP Data Interface Message IDs . . . . .	229
10.10.1 Detailed Description . . . . .	229
10.10.2 Macro Definition Documentation . . . . .	229
10.11 cFE Return Code Defines . . . . .	229
10.11.1 Detailed Description . . . . .	235
10.11.2 Macro Definition Documentation . . . . .	235
10.12 cFE Resource ID APIs . . . . .	252
10.12.1 Detailed Description . . . . .	252
10.12.2 Function Documentation . . . . .	252
10.13 cFE Entry/Exit APIs . . . . .	254
10.13.1 Detailed Description . . . . .	255
10.13.2 Function Documentation . . . . .	255
10.14 cFE Application Control APIs . . . . .	256
10.14.1 Detailed Description . . . . .	256
10.14.2 Function Documentation . . . . .	256
10.15 cFE Application Behavior APIs . . . . .	258
10.15.1 Detailed Description . . . . .	259
10.15.2 Function Documentation . . . . .	259
10.16 cFE Information APIs . . . . .	262
10.16.1 Detailed Description . . . . .	262
10.16.2 Function Documentation . . . . .	262
10.17 cFE Child Task APIs . . . . .	270
10.17.1 Detailed Description . . . . .	271

10.17.2 Function Documentation . . . . .	271
10.18 cFE Miscellaneous APIs . . . . .	274
10.18.1 Detailed Description . . . . .	274
10.18.2 Function Documentation . . . . .	274
10.19 cFE Critical Data Store APIs . . . . .	276
10.19.1 Detailed Description . . . . .	277
10.19.2 Function Documentation . . . . .	277
10.20 cFE Memory Manager APIs . . . . .	281
10.20.1 Detailed Description . . . . .	281
10.20.2 Function Documentation . . . . .	281
10.21 cFE Performance Monitor APIs . . . . .	289
10.21.1 Detailed Description . . . . .	289
10.21.2 Macro Definition Documentation . . . . .	289
10.21.3 Function Documentation . . . . .	290
10.22 cFE Generic Counter APIs . . . . .	291
10.22.1 Detailed Description . . . . .	291
10.22.2 Function Documentation . . . . .	291
10.23 cFE Registration APIs . . . . .	296
10.23.1 Detailed Description . . . . .	296
10.23.2 Function Documentation . . . . .	296
10.24 cFE Send Event APIs . . . . .	297
10.24.1 Detailed Description . . . . .	297
10.24.2 Function Documentation . . . . .	297
10.25 cFE Reset Event Filter APIs . . . . .	301
10.25.1 Detailed Description . . . . .	301
10.25.2 Function Documentation . . . . .	301
10.26 cFE File Header Management APIs . . . . .	303
10.26.1 Detailed Description . . . . .	303
10.26.2 Function Documentation . . . . .	303
10.27 cFE File Utility APIs . . . . .	306
10.27.1 Detailed Description . . . . .	307
10.27.2 Function Documentation . . . . .	307
10.28 cFE Generic Message APIs . . . . .	311
10.28.1 Detailed Description . . . . .	311
10.28.2 Function Documentation . . . . .	311
10.29 cFE Message Primary Header APIs . . . . .	312
10.29.1 Detailed Description . . . . .	312
10.29.2 Function Documentation . . . . .	312
10.30 cFE Message Extended Header APIs . . . . .	320

---

10.30.1 Detailed Description . . . . .	321
10.30.2 Function Documentation . . . . .	321
10.31 cFE Message Secondary Header APIs . . . . .	326
10.31.1 Detailed Description . . . . .	326
10.31.2 Function Documentation . . . . .	326
10.32 cFE Message Id APIs . . . . .	330
10.32.1 Detailed Description . . . . .	330
10.32.2 Function Documentation . . . . .	330
10.33 cFE Message Integrity APIs . . . . .	332
10.33.1 Detailed Description . . . . .	332
10.33.2 Function Documentation . . . . .	332
10.34 cFE Pipe Management APIs . . . . .	334
10.34.1 Detailed Description . . . . .	334
10.34.2 Function Documentation . . . . .	334
10.35 cFE Message Subscription Control APIs . . . . .	339
10.35.1 Detailed Description . . . . .	339
10.35.2 Function Documentation . . . . .	339
10.36 cFE Send/Receive Message APIs . . . . .	343
10.36.1 Detailed Description . . . . .	343
10.36.2 Function Documentation . . . . .	343
10.37 cFE Zero Copy APIs . . . . .	345
10.37.1 Detailed Description . . . . .	346
10.37.2 Function Documentation . . . . .	346
10.38 cFE Message Characteristics APIs . . . . .	348
10.38.1 Detailed Description . . . . .	348
10.38.2 Function Documentation . . . . .	348
10.39 cFE Message ID APIs . . . . .	352
10.39.1 Detailed Description . . . . .	352
10.39.2 Function Documentation . . . . .	352
10.40 cFE SB Pipe options . . . . .	357
10.40.1 Detailed Description . . . . .	357
10.40.2 Macro Definition Documentation . . . . .	357
10.41 cFE Registration APIs . . . . .	357
10.41.1 Detailed Description . . . . .	358
10.41.2 Function Documentation . . . . .	358
10.42 cFE Manage Table Content APIs . . . . .	362
10.42.1 Detailed Description . . . . .	362
10.42.2 Function Documentation . . . . .	362
10.43 cFE Access Table Content APIs . . . . .	368

10.43.1 Detailed Description . . . . .	368
10.43.2 Function Documentation . . . . .	368
10.44 cFE Get Table Information APIs . . . . .	372
10.44.1 Detailed Description . . . . .	372
10.44.2 Function Documentation . . . . .	372
10.45 cFE Table Type Defines . . . . .	375
10.45.1 Detailed Description . . . . .	375
10.45.2 Macro Definition Documentation . . . . .	375
10.46 cFE Get Current Time APIs . . . . .	377
10.46.1 Detailed Description . . . . .	377
10.46.2 Function Documentation . . . . .	377
10.47 cFE Get Time Information APIs . . . . .	380
10.47.1 Detailed Description . . . . .	380
10.47.2 Function Documentation . . . . .	380
10.48 cFE Time Arithmetic APIs . . . . .	382
10.48.1 Detailed Description . . . . .	382
10.48.2 Function Documentation . . . . .	382
10.49 cFE Time Conversion APIs . . . . .	384
10.49.1 Detailed Description . . . . .	384
10.49.2 Function Documentation . . . . .	384
10.50 cFE External Time Source APIs . . . . .	386
10.50.1 Detailed Description . . . . .	386
10.50.2 Function Documentation . . . . .	386
10.51 cFE Miscellaneous Time APIs . . . . .	390
10.51.1 Detailed Description . . . . .	390
10.51.2 Function Documentation . . . . .	390
10.52 cFE Resource ID base values . . . . .	392
10.52.1 Detailed Description . . . . .	392
10.52.2 Enumeration Type Documentation . . . . .	392
10.53 cFE Clock State Flag Defines . . . . .	393
10.53.1 Detailed Description . . . . .	394
10.53.2 Macro Definition Documentation . . . . .	394
10.54 OSAL Semaphore State Defines . . . . .	395
10.54.1 Detailed Description . . . . .	395
10.54.2 Macro Definition Documentation . . . . .	395
10.55 OSAL Binary Semaphore APIs . . . . .	396
10.55.1 Detailed Description . . . . .	396
10.55.2 Function Documentation . . . . .	396
10.56 OSAL BSP low level access APIs . . . . .	400

---

10.56.1 Detailed Description . . . . .	400
10.56.2 Function Documentation . . . . .	401
10.57 OSAL Real Time Clock APIs . . . . .	401
10.57.1 Detailed Description . . . . .	402
10.57.2 Function Documentation . . . . .	402
10.58 OSAL Core Operation APIs . . . . .	415
10.58.1 Detailed Description . . . . .	416
10.58.2 Function Documentation . . . . .	416
10.59 OSAL Condition Variable APIs . . . . .	418
10.59.1 Detailed Description . . . . .	419
10.59.2 Function Documentation . . . . .	419
10.60 OSAL Counting Semaphore APIs . . . . .	424
10.60.1 Detailed Description . . . . .	425
10.60.2 Function Documentation . . . . .	425
10.61 OSAL Directory APIs . . . . .	428
10.61.1 Detailed Description . . . . .	429
10.61.2 Function Documentation . . . . .	429
10.62 OSAL Return Code Defines . . . . .	432
10.62.1 Detailed Description . . . . .	434
10.62.2 Macro Definition Documentation . . . . .	434
10.63 OSAL Error Info APIs . . . . .	439
10.63.1 Detailed Description . . . . .	439
10.63.2 Function Documentation . . . . .	439
10.64 OSAL File Access Option Defines . . . . .	440
10.64.1 Detailed Description . . . . .	440
10.64.2 Macro Definition Documentation . . . . .	440
10.65 OSAL Reference Point For Seek Offset Defines . . . . .	440
10.65.1 Detailed Description . . . . .	441
10.65.2 Macro Definition Documentation . . . . .	441
10.66 OSAL Standard File APIs . . . . .	441
10.66.1 Detailed Description . . . . .	442
10.66.2 Function Documentation . . . . .	442
10.67 OSAL File System Level APIs . . . . .	454
10.67.1 Detailed Description . . . . .	455
10.67.2 Function Documentation . . . . .	455
10.68 OSAL Heap APIs . . . . .	462
10.68.1 Detailed Description . . . . .	462
10.68.2 Function Documentation . . . . .	462
10.69 OSAL Object Type Defines . . . . .	462

---

10.69.1 Detailed Description . . . . .	463
10.69.2 Macro Definition Documentation . . . . .	463
10.70 OSAL Object ID Utility APIs . . . . .	465
10.70.1 Detailed Description . . . . .	465
10.70.2 Function Documentation . . . . .	465
10.71 OSAL Dynamic Loader and Symbol APIs . . . . .	470
10.71.1 Detailed Description . . . . .	470
10.71.2 Function Documentation . . . . .	470
10.72 OSAL Mutex APIs . . . . .	474
10.72.1 Detailed Description . . . . .	474
10.72.2 Function Documentation . . . . .	474
10.73 OSAL Network ID APIs . . . . .	477
10.73.1 Detailed Description . . . . .	477
10.73.2 Function Documentation . . . . .	477
10.74 OSAL Printf APIs . . . . .	478
10.74.1 Detailed Description . . . . .	478
10.74.2 Function Documentation . . . . .	478
10.75 OSAL Message Queue APIs . . . . .	479
10.75.1 Detailed Description . . . . .	479
10.75.2 Function Documentation . . . . .	479
10.76 OSAL RwLock APIs . . . . .	483
10.76.1 Detailed Description . . . . .	483
10.76.2 Function Documentation . . . . .	483
10.77 OSAL Select APIs . . . . .	487
10.77.1 Detailed Description . . . . .	488
10.77.2 Function Documentation . . . . .	488
10.78 OSAL Shell APIs . . . . .	493
10.78.1 Detailed Description . . . . .	493
10.78.2 Function Documentation . . . . .	493
10.79 OSAL Socket Address APIs . . . . .	494
10.79.1 Detailed Description . . . . .	494
10.79.2 Function Documentation . . . . .	494
10.80 OSAL Socket Management APIs . . . . .	497
10.80.1 Detailed Description . . . . .	498
10.80.2 Function Documentation . . . . .	498
10.81 OSAL Task APIs . . . . .	508
10.81.1 Detailed Description . . . . .	508
10.81.2 Function Documentation . . . . .	508
10.82 OSAL Time Base APIs . . . . .	513

---

10.82.1 Detailed Description . . . . .	513
10.82.2 Function Documentation . . . . .	513
10.83 OSAL Timer APIs . . . . .	517
10.83.1 Detailed Description . . . . .	518
10.83.2 Function Documentation . . . . .	518
<b>11 Data Structure Documentation</b> . . . . .	<b>523</b>
11.1 CCSDS_ExtendedHeader Struct Reference . . . . .	523
11.1.1 Detailed Description . . . . .	523
11.1.2 Field Documentation . . . . .	523
11.2 CCSDS_PrimaryHeader Struct Reference . . . . .	523
11.2.1 Detailed Description . . . . .	523
11.2.2 Field Documentation . . . . .	524
11.3 CF_AbandonCmd Struct Reference . . . . .	524
11.3.1 Detailed Description . . . . .	524
11.3.2 Field Documentation . . . . .	524
11.4 CF_AppData_t Struct Reference . . . . .	525
11.4.1 Detailed Description . . . . .	525
11.4.2 Field Documentation . . . . .	525
11.5 CF_CancelCmd Struct Reference . . . . .	526
11.5.1 Detailed Description . . . . .	526
11.5.2 Field Documentation . . . . .	526
11.6 CF_CFDP_FileDirectiveDispatchTable_t Struct Reference . . . . .	527
11.6.1 Detailed Description . . . . .	527
11.6.2 Field Documentation . . . . .	527
11.7 CF_CFDP_Inv Struct Reference . . . . .	527
11.7.1 Detailed Description . . . . .	527
11.7.2 Field Documentation . . . . .	528
11.8 CF_CFDP_PduAck Struct Reference . . . . .	528
11.8.1 Detailed Description . . . . .	528
11.8.2 Field Documentation . . . . .	528
11.9 CF_CFDP_PduEof Struct Reference . . . . .	528
11.9.1 Detailed Description . . . . .	529
11.9.2 Field Documentation . . . . .	529
11.10 CF_CFDP_PduFileDataContent Struct Reference . . . . .	529
11.10.1 Detailed Description . . . . .	529
11.10.2 Field Documentation . . . . .	529
11.11 CF_CFDP_PduFileDataHeader Struct Reference . . . . .	530
11.11.1 Detailed Description . . . . .	530

11.11.2 Field Documentation . . . . .	530
11.12 CF_CFDP_PduFileDirectiveHeader Struct Reference . . . . .	530
11.12.1 Detailed Description . . . . .	530
11.12.2 Field Documentation . . . . .	530
11.13 CF_CFDP_PduFin Struct Reference . . . . .	530
11.13.1 Detailed Description . . . . .	531
11.13.2 Field Documentation . . . . .	531
11.14 CF_CFDP_PduHeader Struct Reference . . . . .	531
11.14.1 Detailed Description . . . . .	531
11.14.2 Field Documentation . . . . .	531
11.15 CF_CFDP_PduMd Struct Reference . . . . .	532
11.15.1 Detailed Description . . . . .	532
11.15.2 Field Documentation . . . . .	532
11.16 CF_CFDP_PduNak Struct Reference . . . . .	533
11.16.1 Detailed Description . . . . .	533
11.16.2 Field Documentation . . . . .	533
11.17 CF_CFDP_R_SubstateDispatchTable_t Struct Reference . . . . .	533
11.17.1 Detailed Description . . . . .	533
11.17.2 Field Documentation . . . . .	533
11.18 CF_CFDP_S_SubstateRecvDispatchTable_t Struct Reference . . . . .	534
11.18.1 Detailed Description . . . . .	534
11.18.2 Field Documentation . . . . .	534
11.19 CF_CFDP_S_SubstateSendDispatchTable_t Struct Reference . . . . .	534
11.19.1 Detailed Description . . . . .	534
11.19.2 Field Documentation . . . . .	534
11.20 CF_CFDP_SegmentRequest Struct Reference . . . . .	535
11.20.1 Detailed Description . . . . .	535
11.20.2 Field Documentation . . . . .	535
11.21 CF_CFDP_Tick_args Struct Reference . . . . .	535
11.21.1 Detailed Description . . . . .	536
11.21.2 Field Documentation . . . . .	536
11.22 CF_CFDP_tlv Struct Reference . . . . .	536
11.22.1 Detailed Description . . . . .	536
11.22.2 Field Documentation . . . . .	536
11.23 CF_CFDP_TxnRecvDispatchTable_t Struct Reference . . . . .	537
11.23.1 Detailed Description . . . . .	537
11.23.2 Field Documentation . . . . .	537
11.24 CF_CFDP_TxnSendDispatchTable_t Struct Reference . . . . .	537
11.24.1 Detailed Description . . . . .	538

---

11.24.2 Field Documentation . . . . .	538
11.25 CF_CFDP_uint16_t Struct Reference . . . . .	538
11.25.1 Detailed Description . . . . .	538
11.25.2 Field Documentation . . . . .	538
11.26 CF_CFDP_uint32_t Struct Reference . . . . .	538
11.26.1 Detailed Description . . . . .	538
11.26.2 Field Documentation . . . . .	539
11.27 CF_CFDP_uint64_t Struct Reference . . . . .	539
11.27.1 Detailed Description . . . . .	539
11.27.2 Field Documentation . . . . .	539
11.28 CF_CFDP_uint8_t Struct Reference . . . . .	539
11.28.1 Detailed Description . . . . .	539
11.28.2 Field Documentation . . . . .	539
11.29 CF_ChanAction_BoolArg Struct Reference . . . . .	540
11.29.1 Detailed Description . . . . .	540
11.29.2 Field Documentation . . . . .	540
11.30 CF_ChanAction_BoolMsgArg Struct Reference . . . . .	540
11.30.1 Detailed Description . . . . .	540
11.30.2 Field Documentation . . . . .	540
11.31 CF_ChanAction_MsgArg Struct Reference . . . . .	541
11.31.1 Detailed Description . . . . .	541
11.31.2 Field Documentation . . . . .	541
11.32 CF_ChanAction_SuspResArg Struct Reference . . . . .	541
11.32.1 Detailed Description . . . . .	541
11.32.2 Field Documentation . . . . .	541
11.33 CF_Channel Struct Reference . . . . .	542
11.33.1 Detailed Description . . . . .	542
11.33.2 Field Documentation . . . . .	542
11.34 CF_ChannelConfig Struct Reference . . . . .	543
11.34.1 Detailed Description . . . . .	544
11.34.2 Field Documentation . . . . .	544
11.35 CF_Chunk Struct Reference . . . . .	546
11.35.1 Detailed Description . . . . .	546
11.35.2 Field Documentation . . . . .	546
11.36 CF_ChunkList Struct Reference . . . . .	547
11.36.1 Detailed Description . . . . .	547
11.36.2 Field Documentation . . . . .	547
11.37 CF_ChunkWrapper Struct Reference . . . . .	547
11.37.1 Detailed Description . . . . .	548

11.37.2 Field Documentation . . . . .	548
11.38 CF_CListNode Struct Reference . . . . .	548
11.38.1 Detailed Description . . . . .	548
11.38.2 Field Documentation . . . . .	548
11.39 CF_Codec_BitField Struct Reference . . . . .	549
11.39.1 Detailed Description . . . . .	549
11.39.2 Field Documentation . . . . .	549
11.40 CF_CodecState Struct Reference . . . . .	549
11.40.1 Detailed Description . . . . .	549
11.40.2 Field Documentation . . . . .	550
11.41 CF_ConfigTable Struct Reference . . . . .	550
11.41.1 Detailed Description . . . . .	550
11.41.2 Field Documentation . . . . .	551
11.42 CF_Crc Struct Reference . . . . .	552
11.42.1 Detailed Description . . . . .	552
11.42.2 Field Documentation . . . . .	552
11.43 CF_DecoderState Struct Reference . . . . .	552
11.43.1 Detailed Description . . . . .	552
11.43.2 Field Documentation . . . . .	553
11.44 CF_DisableDequeueCmd Struct Reference . . . . .	553
11.44.1 Detailed Description . . . . .	553
11.44.2 Field Documentation . . . . .	553
11.45 CF_DisableDirPollingCmd Struct Reference . . . . .	554
11.45.1 Detailed Description . . . . .	554
11.45.2 Field Documentation . . . . .	554
11.46 CF_DisableEngineCmd Struct Reference . . . . .	554
11.46.1 Detailed Description . . . . .	554
11.46.2 Field Documentation . . . . .	554
11.47 CF_EnableDequeueCmd Struct Reference . . . . .	555
11.47.1 Detailed Description . . . . .	555
11.47.2 Field Documentation . . . . .	555
11.48 CF_EnableDirPollingCmd Struct Reference . . . . .	555
11.48.1 Detailed Description . . . . .	556
11.48.2 Field Documentation . . . . .	556
11.49 CF_EnableEngineCmd Struct Reference . . . . .	556
11.49.1 Detailed Description . . . . .	556
11.49.2 Field Documentation . . . . .	556
11.50 CF_EncoderState Struct Reference . . . . .	556
11.50.1 Detailed Description . . . . .	557

---

11.50.2 Field Documentation . . . . .	557
11.51 CF_Engine Struct Reference . . . . .	557
11.51.1 Detailed Description . . . . .	557
11.51.2 Field Documentation . . . . .	558
11.52 CF_EotPacket Struct Reference . . . . .	559
11.52.1 Detailed Description . . . . .	559
11.52.2 Field Documentation . . . . .	559
11.53 CF_EotPacket_Payload Struct Reference . . . . .	559
11.53.1 Detailed Description . . . . .	560
11.53.2 Field Documentation . . . . .	560
11.54 CF_Flags_Common Struct Reference . . . . .	561
11.54.1 Detailed Description . . . . .	562
11.54.2 Field Documentation . . . . .	562
11.55 CF_Flags_Rx Struct Reference . . . . .	563
11.55.1 Detailed Description . . . . .	563
11.55.2 Field Documentation . . . . .	563
11.56 CF_Flags_Tx Struct Reference . . . . .	564
11.56.1 Detailed Description . . . . .	565
11.56.2 Field Documentation . . . . .	565
11.57 CF_FreezeCmd Struct Reference . . . . .	566
11.57.1 Detailed Description . . . . .	566
11.57.2 Field Documentation . . . . .	566
11.58 CF_GapComputeArgs_t Struct Reference . . . . .	566
11.58.1 Detailed Description . . . . .	567
11.58.2 Field Documentation . . . . .	567
11.59 CF_GetParam_Payload Struct Reference . . . . .	567
11.59.1 Detailed Description . . . . .	567
11.59.2 Field Documentation . . . . .	567
11.60 CF_GetParamCmd Struct Reference . . . . .	568
11.60.1 Detailed Description . . . . .	568
11.60.2 Field Documentation . . . . .	568
11.61 CF_History Struct Reference . . . . .	568
11.61.1 Detailed Description . . . . .	569
11.61.2 Field Documentation . . . . .	569
11.62 CF_HkChannel_Data Struct Reference . . . . .	570
11.62.1 Detailed Description . . . . .	571
11.62.2 Field Documentation . . . . .	571
11.63 CF_HkCmdCounters Struct Reference . . . . .	572
11.63.1 Detailed Description . . . . .	572

11.63.2 Field Documentation . . . . .	572
11.64 CF_HkCounters Struct Reference . . . . .	573
11.64.1 Detailed Description . . . . .	573
11.64.2 Field Documentation . . . . .	573
11.65 CF_HkFault Struct Reference . . . . .	573
11.65.1 Detailed Description . . . . .	574
11.65.2 Field Documentation . . . . .	574
11.66 CF_HkPacket Struct Reference . . . . .	576
11.66.1 Detailed Description . . . . .	576
11.66.2 Field Documentation . . . . .	576
11.67 CF_HkPacket_Payload Struct Reference . . . . .	577
11.67.1 Detailed Description . . . . .	577
11.67.2 Field Documentation . . . . .	577
11.68 CF_HkRecv Struct Reference . . . . .	578
11.68.1 Detailed Description . . . . .	578
11.68.2 Field Documentation . . . . .	578
11.69 CF_HkSent Struct Reference . . . . .	579
11.69.1 Detailed Description . . . . .	579
11.69.2 Field Documentation . . . . .	579
11.70 CF_Input Struct Reference . . . . .	580
11.70.1 Detailed Description . . . . .	580
11.70.2 Field Documentation . . . . .	580
11.71 CF_Logical_IntHeader Union Reference . . . . .	581
11.71.1 Detailed Description . . . . .	581
11.71.2 Field Documentation . . . . .	581
11.72 CF_Logical_Lv Struct Reference . . . . .	582
11.72.1 Detailed Description . . . . .	582
11.72.2 Field Documentation . . . . .	582
11.73 CF_Logical_PduAck Struct Reference . . . . .	583
11.73.1 Detailed Description . . . . .	583
11.73.2 Field Documentation . . . . .	583
11.74 CF_Logical_PduBuffer Struct Reference . . . . .	584
11.74.1 Detailed Description . . . . .	584
11.74.2 Field Documentation . . . . .	584
11.75 CF_Logical_PduEof Struct Reference . . . . .	585
11.75.1 Detailed Description . . . . .	586
11.75.2 Field Documentation . . . . .	586
11.76 CF_Logical_PduFileDataHeader Struct Reference . . . . .	586
11.76.1 Detailed Description . . . . .	587

---

11.76.2 Field Documentation . . . . .	587
11.77 CF_Logical_PduFileDirectiveHeader Struct Reference . . . . .	587
11.77.1 Detailed Description . . . . .	588
11.77.2 Field Documentation . . . . .	588
11.78 CF_Logical_PduFin Struct Reference . . . . .	588
11.78.1 Detailed Description . . . . .	588
11.78.2 Field Documentation . . . . .	588
11.79 CF_Logical_PduHeader Struct Reference . . . . .	589
11.79.1 Detailed Description . . . . .	590
11.79.2 Field Documentation . . . . .	590
11.80 CF_Logical_PduMd Struct Reference . . . . .	592
11.80.1 Detailed Description . . . . .	592
11.80.2 Field Documentation . . . . .	592
11.81 CF_Logical_PduNak Struct Reference . . . . .	593
11.81.1 Detailed Description . . . . .	593
11.81.2 Field Documentation . . . . .	593
11.82 CF_Logical_SegmentList Struct Reference . . . . .	594
11.82.1 Detailed Description . . . . .	594
11.82.2 Field Documentation . . . . .	594
11.83 CF_Logical_SegmentRequest Struct Reference . . . . .	594
11.83.1 Detailed Description . . . . .	595
11.83.2 Field Documentation . . . . .	595
11.84 CF_Logical_Tlv Struct Reference . . . . .	595
11.84.1 Detailed Description . . . . .	595
11.84.2 Field Documentation . . . . .	595
11.85 CF_Logical_TlvData Union Reference . . . . .	596
11.85.1 Detailed Description . . . . .	596
11.85.2 Field Documentation . . . . .	596
11.86 CF_Logical_TlvList Struct Reference . . . . .	597
11.86.1 Detailed Description . . . . .	597
11.86.2 Field Documentation . . . . .	597
11.87 CF_NoopCmd Struct Reference . . . . .	597
11.87.1 Detailed Description . . . . .	597
11.87.2 Field Documentation . . . . .	597
11.88 CF_Output Struct Reference . . . . .	598
11.88.1 Detailed Description . . . . .	598
11.88.2 Field Documentation . . . . .	598
11.89 CF_PduCmdMsg Struct Reference . . . . .	598
11.89.1 Detailed Description . . . . .	599

11.89.2 Field Documentation . . . . .	599
11.90 CF_PduTlmMsg Struct Reference . . . . .	599
11.90.1 Detailed Description . . . . .	599
11.90.2 Field Documentation . . . . .	600
11.91 CF_Playback Struct Reference . . . . .	600
11.91.1 Detailed Description . . . . .	600
11.91.2 Field Documentation . . . . .	600
11.92 CF_PlaybackDirCmd Struct Reference . . . . .	602
11.92.1 Detailed Description . . . . .	602
11.92.2 Field Documentation . . . . .	602
11.93 CF_Poll Struct Reference . . . . .	602
11.93.1 Detailed Description . . . . .	602
11.93.2 Field Documentation . . . . .	602
11.94 CF_PollDir Struct Reference . . . . .	603
11.94.1 Detailed Description . . . . .	603
11.94.2 Field Documentation . . . . .	603
11.95 CF_PurgeQueueCmd Struct Reference . . . . .	604
11.95.1 Detailed Description . . . . .	605
11.95.2 Field Documentation . . . . .	605
11.96 CF_ResetCountersCmd Struct Reference . . . . .	605
11.96.1 Detailed Description . . . . .	605
11.96.2 Field Documentation . . . . .	605
11.97 CF_ResumeCmd Struct Reference . . . . .	606
11.97.1 Detailed Description . . . . .	606
11.97.2 Field Documentation . . . . .	606
11.98 CF_SendHkCmd Struct Reference . . . . .	606
11.98.1 Detailed Description . . . . .	606
11.98.2 Field Documentation . . . . .	607
11.99 CF_SetParam_Payload Struct Reference . . . . .	607
11.99.1 Detailed Description . . . . .	607
11.99.2 Field Documentation . . . . .	607
11.100 CF_SetParamCmd Struct Reference . . . . .	608
11.100.1 Detailed Description . . . . .	608
11.100.2 Field Documentation . . . . .	608
11.101 CF_StateData Struct Reference . . . . .	608
11.101.1 Detailed Description . . . . .	609
11.101.2 Field Documentation . . . . .	609
11.102 CF_StateFlags Union Reference . . . . .	610
11.102.1 Detailed Description . . . . .	610

---

11.102.2 Field Documentation . . . . .	610
11.103 CF_SuspendCmd Struct Reference . . . . .	611
11.103.1 Detailed Description . . . . .	611
11.103.2 Field Documentation . . . . .	611
11.104 CF_ThawCmd Struct Reference . . . . .	612
11.104.1 Detailed Description . . . . .	612
11.104.2 Field Documentation . . . . .	612
11.105 CF_Timer Struct Reference . . . . .	612
11.105.1 Detailed Description . . . . .	613
11.105.2 Field Documentation . . . . .	613
11.106 CF_Transaction Struct Reference . . . . .	613
11.106.1 Detailed Description . . . . .	614
11.106.2 Field Documentation . . . . .	614
11.107 CF_Transaction_Payload Struct Reference . . . . .	617
11.107.1 Detailed Description . . . . .	617
11.107.2 Field Documentation . . . . .	617
11.108 CF_Traverse_PriorityArg Struct Reference . . . . .	618
11.108.1 Detailed Description . . . . .	618
11.108.2 Field Documentation . . . . .	618
11.109 CF_Traverse_TransSeqArg Struct Reference . . . . .	618
11.109.1 Detailed Description . . . . .	619
11.109.2 Field Documentation . . . . .	619
11.110 CF_Traverse_WriteHistoryFileArg Struct Reference . . . . .	619
11.110.1 Detailed Description . . . . .	620
11.110.2 Field Documentation . . . . .	620
11.111 CF_Traverse_WriteTxnFileArg Struct Reference . . . . .	620
11.111.1 Detailed Description . . . . .	620
11.111.2 Field Documentation . . . . .	621
11.112 CF_TraverseAll_Arg Struct Reference . . . . .	621
11.112.1 Detailed Description . . . . .	621
11.112.2 Field Documentation . . . . .	621
11.113 CF_TxFile_Payload Struct Reference . . . . .	622
11.113.1 Detailed Description . . . . .	622
11.113.2 Field Documentation . . . . .	622
11.114 CF_TxFileCmd Struct Reference . . . . .	623
11.114.1 Detailed Description . . . . .	624
11.114.2 Field Documentation . . . . .	624
11.115 CF_TxnFilenames Struct Reference . . . . .	624
11.115.1 Detailed Description . . . . .	624

---

11.115.2 Field Documentation . . . . .	624
11.116 CF_UnionArgs_Payload Union Reference . . . . .	625
11.116.1 Detailed Description . . . . .	625
11.116.2 Field Documentation . . . . .	625
11.117 CF_WakeupCmd Struct Reference . . . . .	625
11.117.1 Detailed Description . . . . .	626
11.117.2 Field Documentation . . . . .	626
11.118 CF_WriteQueue_Payload Struct Reference . . . . .	626
11.118.1 Detailed Description . . . . .	626
11.118.2 Field Documentation . . . . .	626
11.119 CF_WriteQueueCmd Struct Reference . . . . .	627
11.119.1 Detailed Description . . . . .	627
11.119.2 Field Documentation . . . . .	627
11.120 CFE_Config_ArrayValue Struct Reference . . . . .	628
11.120.1 Detailed Description . . . . .	628
11.120.2 Field Documentation . . . . .	628
11.121 CFE_Config_IdNameEntry Struct Reference . . . . .	628
11.121.1 Detailed Description . . . . .	628
11.121.2 Field Documentation . . . . .	628
11.122 CFE_Config_ValueBuffer Union Reference . . . . .	629
11.122.1 Detailed Description . . . . .	629
11.122.2 Field Documentation . . . . .	629
11.123 CFE_Config_ValueEntry Struct Reference . . . . .	629
11.123.1 Detailed Description . . . . .	629
11.123.2 Field Documentation . . . . .	629
11.124 CFE_ES_AppInfo Struct Reference . . . . .	630
11.124.1 Detailed Description . . . . .	631
11.124.2 Field Documentation . . . . .	631
11.125 CFE_ES_AppNameCmd_Payload Struct Reference . . . . .	634
11.125.1 Detailed Description . . . . .	634
11.125.2 Field Documentation . . . . .	634
11.126 CFE_ES_AppReloadCmd_Payload Struct Reference . . . . .	634
11.126.1 Detailed Description . . . . .	634
11.126.2 Field Documentation . . . . .	635
11.127 CFE_ES_BlockStats Struct Reference . . . . .	635
11.127.1 Detailed Description . . . . .	635
11.127.2 Field Documentation . . . . .	635
11.128 CFE_ES_CDSRegDumpRec Struct Reference . . . . .	636
11.128.1 Detailed Description . . . . .	636

---

11.128.2 Field Documentation . . . . .	636
11.129 CFE_ES_ClearERLogCmd Struct Reference . . . . .	637
11.129.1 Detailed Description . . . . .	637
11.129.2 Field Documentation . . . . .	637
11.130 CFE_ES_ClearSysLogCmd Struct Reference . . . . .	637
11.130.1 Detailed Description . . . . .	637
11.130.2 Field Documentation . . . . .	638
11.131 CFE_ES_DeleteCDSCmd Struct Reference . . . . .	638
11.131.1 Detailed Description . . . . .	638
11.131.2 Field Documentation . . . . .	638
11.132 CFE_ES_DeleteCDSCmd_Payload Struct Reference . . . . .	638
11.132.1 Detailed Description . . . . .	639
11.132.2 Field Documentation . . . . .	639
11.133 CFE_ES_DumpCDSRegistryCmd Struct Reference . . . . .	639
11.133.1 Detailed Description . . . . .	639
11.133.2 Field Documentation . . . . .	639
11.134 CFE_ES_DumpCDSRegistryCmd_Payload Struct Reference . . . . .	639
11.134.1 Detailed Description . . . . .	640
11.134.2 Field Documentation . . . . .	640
11.135 CFE_ES_FileNameCmd Struct Reference . . . . .	640
11.135.1 Detailed Description . . . . .	640
11.135.2 Field Documentation . . . . .	640
11.136 CFE_ES_FileNameCmd_Payload Struct Reference . . . . .	641
11.136.1 Detailed Description . . . . .	641
11.136.2 Field Documentation . . . . .	641
11.137 CFE_ES_HousekeepingTlm Struct Reference . . . . .	641
11.137.1 Detailed Description . . . . .	641
11.137.2 Field Documentation . . . . .	641
11.138 CFE_ES_HousekeepingTlm_Payload Struct Reference . . . . .	642
11.138.1 Detailed Description . . . . .	644
11.138.2 Field Documentation . . . . .	644
11.139 CFE_ES_MemAddress_t Struct Reference . . . . .	650
11.139.1 Detailed Description . . . . .	650
11.139.2 Field Documentation . . . . .	650
11.140 CFE_ES_MemOffset_t Struct Reference . . . . .	650
11.140.1 Detailed Description . . . . .	651
11.140.2 Field Documentation . . . . .	651
11.141 CFE_ES_MemPoolStats Struct Reference . . . . .	651
11.141.1 Detailed Description . . . . .	651

11.141.2 Field Documentation . . . . .	652
11.142 CFE_ES_MemStatsTlm Struct Reference . . . . .	652
11.142.1 Detailed Description . . . . .	653
11.142.2 Field Documentation . . . . .	653
11.143 CFE_ES_NoopCmd Struct Reference . . . . .	653
11.143.1 Detailed Description . . . . .	653
11.143.2 Field Documentation . . . . .	653
11.144 CFE_ES_OneAppTlm Struct Reference . . . . .	653
11.144.1 Detailed Description . . . . .	654
11.144.2 Field Documentation . . . . .	654
11.145 CFE_ES_OneAppTlm_Payload Struct Reference . . . . .	654
11.145.1 Detailed Description . . . . .	654
11.145.2 Field Documentation . . . . .	654
11.146 CFE_ES_OverWriteSysLogCmd Struct Reference . . . . .	654
11.146.1 Detailed Description . . . . .	655
11.146.2 Field Documentation . . . . .	655
11.147 CFE_ES_OverWriteSysLogCmd_Payload Struct Reference . . . . .	655
11.147.1 Detailed Description . . . . .	655
11.147.2 Field Documentation . . . . .	655
11.148 CFE_ES_PoolAlign Union Reference . . . . .	656
11.148.1 Detailed Description . . . . .	656
11.148.2 Field Documentation . . . . .	656
11.149 CFE_ES_PoolStatsTlm_Payload Struct Reference . . . . .	656
11.149.1 Detailed Description . . . . .	657
11.149.2 Field Documentation . . . . .	657
11.150 CFE_ES_QueryAllCmd Struct Reference . . . . .	657
11.150.1 Detailed Description . . . . .	657
11.150.2 Field Documentation . . . . .	657
11.151 CFE_ES_QueryAllTasksCmd Struct Reference . . . . .	658
11.151.1 Detailed Description . . . . .	658
11.151.2 Field Documentation . . . . .	658
11.152 CFE_ES_QueryOneCmd Struct Reference . . . . .	658
11.152.1 Detailed Description . . . . .	658
11.152.2 Field Documentation . . . . .	658
11.153 CFE_ES_ReloadAppCmd Struct Reference . . . . .	659
11.153.1 Detailed Description . . . . .	659
11.153.2 Field Documentation . . . . .	659
11.154 CFE_ES_ResetCountersCmd Struct Reference . . . . .	659
11.154.1 Detailed Description . . . . .	659

---

11.154.2 Field Documentation . . . . .	660
11.155 CFE_ES_ResetPRCountCmd Struct Reference . . . . .	660
11.155.1 Detailed Description . . . . .	660
11.155.2 Field Documentation . . . . .	660
11.156 CFE_ES_RestartAppCmd Struct Reference . . . . .	660
11.156.1 Detailed Description . . . . .	660
11.156.2 Field Documentation . . . . .	660
11.157 CFE_ES_RestartCmd Struct Reference . . . . .	661
11.157.1 Detailed Description . . . . .	661
11.157.2 Field Documentation . . . . .	661
11.158 CFE_ES_RestartCmd_Payload Struct Reference . . . . .	661
11.158.1 Detailed Description . . . . .	662
11.158.2 Field Documentation . . . . .	662
11.159 CFE_ES_SendHkCmd Struct Reference . . . . .	662
11.159.1 Detailed Description . . . . .	662
11.159.2 Field Documentation . . . . .	662
11.160 CFE_ES_SendMemPoolStatsCmd Struct Reference . . . . .	662
11.160.1 Detailed Description . . . . .	663
11.160.2 Field Documentation . . . . .	663
11.161 CFE_ES_SendMemPoolStatsCmd_Payload Struct Reference . . . . .	663
11.161.1 Detailed Description . . . . .	663
11.161.2 Field Documentation . . . . .	663
11.162 CFE_ES_SetMaxPRCountCmd Struct Reference . . . . .	664
11.162.1 Detailed Description . . . . .	664
11.162.2 Field Documentation . . . . .	664
11.163 CFE_ES_SetMaxPRCountCmd_Payload Struct Reference . . . . .	664
11.163.1 Detailed Description . . . . .	664
11.163.2 Field Documentation . . . . .	664
11.164 CFE_ES_SetPerfFilterMaskCmd Struct Reference . . . . .	665
11.164.1 Detailed Description . . . . .	665
11.164.2 Field Documentation . . . . .	665
11.165 CFE_ES_SetPerfFilterMaskCmd_Payload Struct Reference . . . . .	665
11.165.1 Detailed Description . . . . .	666
11.165.2 Field Documentation . . . . .	666
11.166 CFE_ES_SetPerfTriggerMaskCmd Struct Reference . . . . .	666
11.166.1 Detailed Description . . . . .	666
11.166.2 Field Documentation . . . . .	666
11.167 CFE_ES_SetPerfTrigMaskCmd_Payload Struct Reference . . . . .	667
11.167.1 Detailed Description . . . . .	667

11.167.2 Field Documentation . . . . .	667
11.168 CFE_ES_StartApp Struct Reference . . . . .	667
11.168.1 Detailed Description . . . . .	667
11.168.2 Field Documentation . . . . .	667
11.169 CFE_ES_StartAppCmd_Payload Struct Reference . . . . .	668
11.169.1 Detailed Description . . . . .	668
11.169.2 Field Documentation . . . . .	668
11.170 CFE_ES_StartPerfCmd_Payload Struct Reference . . . . .	669
11.170.1 Detailed Description . . . . .	669
11.170.2 Field Documentation . . . . .	669
11.171 CFE_ES_StartPerfDataCmd Struct Reference . . . . .	669
11.171.1 Detailed Description . . . . .	670
11.171.2 Field Documentation . . . . .	670
11.172 CFE_ES_StopAppCmd Struct Reference . . . . .	670
11.172.1 Detailed Description . . . . .	670
11.172.2 Field Documentation . . . . .	670
11.173 CFE_ES_StopPerfCmd_Payload Struct Reference . . . . .	671
11.173.1 Detailed Description . . . . .	671
11.173.2 Field Documentation . . . . .	671
11.174 CFE_ES_StopPerfDataCmd Struct Reference . . . . .	671
11.174.1 Detailed Description . . . . .	671
11.174.2 Field Documentation . . . . .	671
11.175 CFE_ES_TaskInfo Struct Reference . . . . .	672
11.175.1 Detailed Description . . . . .	672
11.175.2 Field Documentation . . . . .	672
11.176 CFE_ES_WriteERLogCmd Struct Reference . . . . .	673
11.176.1 Detailed Description . . . . .	673
11.176.2 Field Documentation . . . . .	674
11.177 CFE_ES_WriteSysLogCmd Struct Reference . . . . .	674
11.177.1 Detailed Description . . . . .	674
11.177.2 Field Documentation . . . . .	674
11.178 CFE_EVS_AddEventFilterCmd Struct Reference . . . . .	674
11.178.1 Detailed Description . . . . .	675
11.178.2 Field Documentation . . . . .	675
11.179 CFE_EVS_AppDataCmd_Payload Struct Reference . . . . .	675
11.179.1 Detailed Description . . . . .	675
11.179.2 Field Documentation . . . . .	675
11.180 CFE_EVS_AppNameBitMaskCmd_Payload Struct Reference . . . . .	676
11.180.1 Detailed Description . . . . .	676

---

11.180.2 Field Documentation . . . . .	676
11.181 CFE_EVS_AppNameCmd_Payload Struct Reference . . . . .	676
11.181.1 Detailed Description . . . . .	677
11.181.2 Field Documentation . . . . .	677
11.182 CFE_EVS_AppNameEventIDCmd_Payload Struct Reference . . . . .	677
11.182.1 Detailed Description . . . . .	677
11.182.2 Field Documentation . . . . .	677
11.183 CFE_EVS_AppNameEventIDMaskCmd_Payload Struct Reference . . . . .	677
11.183.1 Detailed Description . . . . .	678
11.183.2 Field Documentation . . . . .	678
11.184 CFE_EVS_AppTlmData Struct Reference . . . . .	678
11.184.1 Detailed Description . . . . .	679
11.184.2 Field Documentation . . . . .	679
11.185 CFE_EVS_BinFilter Struct Reference . . . . .	679
11.185.1 Detailed Description . . . . .	680
11.185.2 Field Documentation . . . . .	680
11.186 CFE_EVS_BitMaskCmd_Payload Struct Reference . . . . .	680
11.186.1 Detailed Description . . . . .	680
11.186.2 Field Documentation . . . . .	680
11.187 CFE_EVS_ClearLogCmd Struct Reference . . . . .	681
11.187.1 Detailed Description . . . . .	681
11.187.2 Field Documentation . . . . .	681
11.188 CFE_EVS_DeleteEventFilterCmd Struct Reference . . . . .	681
11.188.1 Detailed Description . . . . .	681
11.188.2 Field Documentation . . . . .	681
11.189 CFE_EVS_DisableAppEventsCmd Struct Reference . . . . .	682
11.189.1 Detailed Description . . . . .	682
11.189.2 Field Documentation . . . . .	682
11.190 CFE_EVS_DisableAppEventTypeCmd Struct Reference . . . . .	682
11.190.1 Detailed Description . . . . .	682
11.190.2 Field Documentation . . . . .	682
11.191 CFE_EVS_DisableEventTypeCmd Struct Reference . . . . .	683
11.191.1 Detailed Description . . . . .	683
11.191.2 Field Documentation . . . . .	683
11.192 CFE_EVS_DisablePortsCmd Struct Reference . . . . .	683
11.192.1 Detailed Description . . . . .	683
11.192.2 Field Documentation . . . . .	684
11.193 CFE_EVS_EnableAppEventsCmd Struct Reference . . . . .	684
11.193.1 Detailed Description . . . . .	684

11.193.2 Field Documentation . . . . .	684
11.194 CFE_EVS_EnableAppEventTypeCmd Struct Reference . . . . .	684
11.194.1 Detailed Description . . . . .	685
11.194.2 Field Documentation . . . . .	685
11.195 CFE_EVS_EnableEventTypeCmd Struct Reference . . . . .	685
11.195.1 Detailed Description . . . . .	685
11.195.2 Field Documentation . . . . .	685
11.196 CFE_EVS_EnablePortsCmd Struct Reference . . . . .	686
11.196.1 Detailed Description . . . . .	686
11.196.2 Field Documentation . . . . .	686
11.197 CFE_EVS_HousekeepingTlm Struct Reference . . . . .	686
11.197.1 Detailed Description . . . . .	686
11.197.2 Field Documentation . . . . .	687
11.198 CFE_EVS_HousekeepingTlm_Payload Struct Reference . . . . .	687
11.198.1 Detailed Description . . . . .	688
11.198.2 Field Documentation . . . . .	688
11.199 CFE_EVS_LogFileCmd_Payload Struct Reference . . . . .	690
11.199.1 Detailed Description . . . . .	690
11.199.2 Field Documentation . . . . .	690
11.200 CFE_EVS_LongEventTlm Struct Reference . . . . .	690
11.200.1 Detailed Description . . . . .	691
11.200.2 Field Documentation . . . . .	691
11.201 CFE_EVS_LongEventTlm_Payload Struct Reference . . . . .	691
11.201.1 Detailed Description . . . . .	691
11.201.2 Field Documentation . . . . .	691
11.202 CFE_EVS_NoopCmd Struct Reference . . . . .	692
11.202.1 Detailed Description . . . . .	692
11.202.2 Field Documentation . . . . .	692
11.203 CFE_EVS_PacketID Struct Reference . . . . .	692
11.203.1 Detailed Description . . . . .	693
11.203.2 Field Documentation . . . . .	693
11.204 CFE_EVS_ResetAllFiltersCmd Struct Reference . . . . .	694
11.204.1 Detailed Description . . . . .	694
11.204.2 Field Documentation . . . . .	694
11.205 CFE_EVS_ResetAppCounterCmd Struct Reference . . . . .	694
11.205.1 Detailed Description . . . . .	694
11.205.2 Field Documentation . . . . .	695
11.206 CFE_EVS_ResetCountersCmd Struct Reference . . . . .	695
11.206.1 Detailed Description . . . . .	695

---

11.206.2 Field Documentation . . . . .	695
11.207 CFE_EVS_ResetFilterCmd Struct Reference . . . . .	695
11.207.1 Detailed Description . . . . .	695
11.207.2 Field Documentation . . . . .	696
11.208 CFE_EVS_SendHkCmd Struct Reference . . . . .	696
11.208.1 Detailed Description . . . . .	696
11.208.2 Field Documentation . . . . .	696
11.209 CFE_EVS_SetEventFormatCode_Payload Struct Reference . . . . .	696
11.209.1 Detailed Description . . . . .	697
11.209.2 Field Documentation . . . . .	697
11.210 CFE_EVS_SetEventFormatModeCmd Struct Reference . . . . .	697
11.210.1 Detailed Description . . . . .	697
11.210.2 Field Documentation . . . . .	697
11.211 CFE_EVS_SetFilterCmd Struct Reference . . . . .	698
11.211.1 Detailed Description . . . . .	698
11.211.2 Field Documentation . . . . .	698
11.212 CFE_EVS_SetLogMode_Payload Struct Reference . . . . .	698
11.212.1 Detailed Description . . . . .	698
11.212.2 Field Documentation . . . . .	698
11.213 CFE_EVS_SetLogModeCmd Struct Reference . . . . .	699
11.213.1 Detailed Description . . . . .	699
11.213.2 Field Documentation . . . . .	699
11.214 CFE_EVS_ShortEventTlm Struct Reference . . . . .	699
11.214.1 Detailed Description . . . . .	700
11.214.2 Field Documentation . . . . .	700
11.215 CFE_EVS_ShortEventTlm_Payload Struct Reference . . . . .	700
11.215.1 Detailed Description . . . . .	700
11.215.2 Field Documentation . . . . .	700
11.216 CFE_EVS_WriteAppDataFileCmd Struct Reference . . . . .	700
11.216.1 Detailed Description . . . . .	701
11.216.2 Field Documentation . . . . .	701
11.217 CFE_EVS_WriteLogFileCmd Struct Reference . . . . .	701
11.217.1 Detailed Description . . . . .	701
11.217.2 Field Documentation . . . . .	701
11.218 CFE_FS_FileWriteMetaData Struct Reference . . . . .	702
11.218.1 Detailed Description . . . . .	702
11.218.2 Field Documentation . . . . .	702
11.219 CFE_FS_Header Struct Reference . . . . .	703
11.219.1 Detailed Description . . . . .	703

11.219.2 Field Documentation . . . . .	703
11.220 CFE_SB_AllSubscriptionsTlm Struct Reference . . . . .	704
11.220.1 Detailed Description . . . . .	705
11.220.2 Field Documentation . . . . .	705
11.221 CFE_SB_AllSubscriptionsTlm_Payload Struct Reference . . . . .	705
11.221.1 Detailed Description . . . . .	705
11.221.2 Field Documentation . . . . .	705
11.222 CFE_SB_DisableRouteCmd Struct Reference . . . . .	706
11.222.1 Detailed Description . . . . .	706
11.222.2 Field Documentation . . . . .	706
11.223 CFE_SB_DisableSubReportingCmd Struct Reference . . . . .	706
11.223.1 Detailed Description . . . . .	707
11.223.2 Field Documentation . . . . .	707
11.224 CFE_SB_EnableRouteCmd Struct Reference . . . . .	707
11.224.1 Detailed Description . . . . .	707
11.224.2 Field Documentation . . . . .	707
11.225 CFE_SB_EnableSubReportingCmd Struct Reference . . . . .	707
11.225.1 Detailed Description . . . . .	708
11.225.2 Field Documentation . . . . .	708
11.226 CFE_SB_HousekeepingTlm Struct Reference . . . . .	708
11.226.1 Detailed Description . . . . .	708
11.226.2 Field Documentation . . . . .	708
11.227 CFE_SB_HousekeepingTlm_Payload Struct Reference . . . . .	708
11.227.1 Detailed Description . . . . .	709
11.227.2 Field Documentation . . . . .	709
11.228 CFE_SB_Msg Union Reference . . . . .	712
11.228.1 Detailed Description . . . . .	712
11.228.2 Field Documentation . . . . .	712
11.229 CFE_SB_MsgId_t Struct Reference . . . . .	713
11.229.1 Detailed Description . . . . .	713
11.229.2 Field Documentation . . . . .	713
11.230 CFE_SB_MsgMapFileEntry Struct Reference . . . . .	713
11.230.1 Detailed Description . . . . .	714
11.230.2 Field Documentation . . . . .	714
11.231 CFE_SB_NoopCmd Struct Reference . . . . .	714
11.231.1 Detailed Description . . . . .	714
11.231.2 Field Documentation . . . . .	714
11.232 CFE_SB_PipeDepthStats Struct Reference . . . . .	714
11.232.1 Detailed Description . . . . .	715

---

11.232.2 Field Documentation . . . . .	715
11.233 CFE_SB_PipeInfoEntry Struct Reference . . . . .	716
11.233.1 Detailed Description . . . . .	716
11.233.2 Field Documentation . . . . .	716
11.234 CFE_SB_Qos_t Struct Reference . . . . .	717
11.234.1 Detailed Description . . . . .	717
11.234.2 Field Documentation . . . . .	717
11.235 CFE_SB_ResetCountersCmd Struct Reference . . . . .	718
11.235.1 Detailed Description . . . . .	718
11.235.2 Field Documentation . . . . .	718
11.236 CFE_SB_RouteCmd_Payload Struct Reference . . . . .	718
11.236.1 Detailed Description . . . . .	718
11.236.2 Field Documentation . . . . .	719
11.237 CFE_SB_RoutingFileEntry Struct Reference . . . . .	719
11.237.1 Detailed Description . . . . .	719
11.237.2 Field Documentation . . . . .	719
11.238 CFE_SB_SendHkCmd Struct Reference . . . . .	720
11.238.1 Detailed Description . . . . .	720
11.238.2 Field Documentation . . . . .	720
11.239 CFE_SB_SendPrevSubsCmd Struct Reference . . . . .	721
11.239.1 Detailed Description . . . . .	721
11.239.2 Field Documentation . . . . .	721
11.240 CFE_SB_SendSbStatsCmd Struct Reference . . . . .	721
11.240.1 Detailed Description . . . . .	721
11.240.2 Field Documentation . . . . .	721
11.241 CFE_SB_SingleSubscriptionTlm Struct Reference . . . . .	721
11.241.1 Detailed Description . . . . .	722
11.241.2 Field Documentation . . . . .	722
11.242 CFE_SB_SingleSubscriptionTlm_Payload Struct Reference . . . . .	722
11.242.1 Detailed Description . . . . .	722
11.242.2 Field Documentation . . . . .	722
11.243 CFE_SB_StatsTlm Struct Reference . . . . .	723
11.243.1 Detailed Description . . . . .	723
11.243.2 Field Documentation . . . . .	723
11.244 CFE_SB_StatsTlm_Payload Struct Reference . . . . .	724
11.244.1 Detailed Description . . . . .	724
11.244.2 Field Documentation . . . . .	724
11.245 CFE_SB_SubEntries Struct Reference . . . . .	727
11.245.1 Detailed Description . . . . .	727

11.245.2 Field Documentation . . . . .	727
11.246 CFE_SB_WriteFileInfoCmd_Payload Struct Reference . . . . .	728
11.246.1 Detailed Description . . . . .	728
11.246.2 Field Documentation . . . . .	728
11.247 CFE_SB_WriteMapInfoCmd Struct Reference . . . . .	728
11.247.1 Detailed Description . . . . .	728
11.247.2 Field Documentation . . . . .	728
11.248 CFE_SB_WritePipeInfoCmd Struct Reference . . . . .	729
11.248.1 Detailed Description . . . . .	729
11.248.2 Field Documentation . . . . .	729
11.249 CFE_SB_WriteRoutingInfoCmd Struct Reference . . . . .	729
11.249.1 Detailed Description . . . . .	730
11.249.2 Field Documentation . . . . .	730
11.250 CFE_TBL_AbortLoadCmd Struct Reference . . . . .	730
11.250.1 Detailed Description . . . . .	730
11.250.2 Field Documentation . . . . .	730
11.251 CFE_TBL_AbortLoadCmd_Payload Struct Reference . . . . .	731
11.251.1 Detailed Description . . . . .	731
11.251.2 Field Documentation . . . . .	731
11.252 CFE_TBL_ActivateCmd Struct Reference . . . . .	731
11.252.1 Detailed Description . . . . .	731
11.252.2 Field Documentation . . . . .	731
11.253 CFE_TBL_ActivateCmd_Payload Struct Reference . . . . .	732
11.253.1 Detailed Description . . . . .	732
11.253.2 Field Documentation . . . . .	732
11.254 CFE_TBL_CombinedFileHdr Struct Reference . . . . .	732
11.254.1 Detailed Description . . . . .	732
11.254.2 Field Documentation . . . . .	732
11.255 CFE_TBL_DelCDSCmd_Payload Struct Reference . . . . .	733
11.255.1 Detailed Description . . . . .	733
11.255.2 Field Documentation . . . . .	733
11.256 CFE_TBL_DeleteCDSCmd Struct Reference . . . . .	733
11.256.1 Detailed Description . . . . .	733
11.256.2 Field Documentation . . . . .	734
11.257 CFE_TBL_DumpCmd Struct Reference . . . . .	734
11.257.1 Detailed Description . . . . .	734
11.257.2 Field Documentation . . . . .	734
11.258 CFE_TBL_DumpCmd_Payload Struct Reference . . . . .	734
11.258.1 Detailed Description . . . . .	735

---

11.258.2 Field Documentation . . . . .	735
11.259 CFE_TBL_DumpRegistryCmd Struct Reference . . . . .	735
11.259.1 Detailed Description . . . . .	736
11.259.2 Field Documentation . . . . .	736
11.260 CFE_TBL_DumpRegistryCmd_Payload Struct Reference . . . . .	736
11.260.1 Detailed Description . . . . .	736
11.260.2 Field Documentation . . . . .	736
11.261 CFE_TBL_File_Hdr Struct Reference . . . . .	736
11.261.1 Detailed Description . . . . .	737
11.261.2 Field Documentation . . . . .	737
11.262 CFE_TBL_FileDef Struct Reference . . . . .	737
11.262.1 Detailed Description . . . . .	738
11.262.2 Field Documentation . . . . .	738
11.263 CFE_TBL_HousekeepingTlm Struct Reference . . . . .	739
11.263.1 Detailed Description . . . . .	739
11.263.2 Field Documentation . . . . .	739
11.264 CFE_TBL_HousekeepingTlm_Payload Struct Reference . . . . .	739
11.264.1 Detailed Description . . . . .	740
11.264.2 Field Documentation . . . . .	741
11.265 CFE_TBL_Info Struct Reference . . . . .	743
11.265.1 Detailed Description . . . . .	744
11.265.2 Field Documentation . . . . .	744
11.266 CFE_TBL_LoadCmd Struct Reference . . . . .	745
11.266.1 Detailed Description . . . . .	746
11.266.2 Field Documentation . . . . .	746
11.267 CFE_TBL_LoadCmd_Payload Struct Reference . . . . .	746
11.267.1 Detailed Description . . . . .	746
11.267.2 Field Documentation . . . . .	746
11.268 CFE_TBL_NoopCmd Struct Reference . . . . .	746
11.268.1 Detailed Description . . . . .	747
11.268.2 Field Documentation . . . . .	747
11.269 CFE_TBL_NotifyCmd Struct Reference . . . . .	747
11.269.1 Detailed Description . . . . .	747
11.269.2 Field Documentation . . . . .	747
11.270 CFE_TBL_NotifyCmd_Payload Struct Reference . . . . .	747
11.270.1 Detailed Description . . . . .	748
11.270.2 Field Documentation . . . . .	748
11.271 CFE_TBL_ResetCountersCmd Struct Reference . . . . .	748
11.271.1 Detailed Description . . . . .	748

11.271.2 Field Documentation . . . . .	748
11.272 CFE_TBL_SendHkCmd Struct Reference . . . . .	748
11.272.1 Detailed Description . . . . .	749
11.272.2 Field Documentation . . . . .	749
11.273 CFE_TBL_SendRegistryCmd Struct Reference . . . . .	749
11.273.1 Detailed Description . . . . .	749
11.273.2 Field Documentation . . . . .	749
11.274 CFE_TBL_SendRegistryCmd_Payload Struct Reference . . . . .	750
11.274.1 Detailed Description . . . . .	750
11.274.2 Field Documentation . . . . .	750
11.275 CFE_TBL_TableRegistryTlm Struct Reference . . . . .	750
11.275.1 Detailed Description . . . . .	750
11.275.2 Field Documentation . . . . .	750
11.276 CFE_TBL_TblRegPacket_Payload Struct Reference . . . . .	751
11.276.1 Detailed Description . . . . .	751
11.276.2 Field Documentation . . . . .	752
11.277 CFE_TBL_ValidateCmd Struct Reference . . . . .	754
11.277.1 Detailed Description . . . . .	754
11.277.2 Field Documentation . . . . .	754
11.278 CFE_TBL_ValidateCmd_Payload Struct Reference . . . . .	754
11.278.1 Detailed Description . . . . .	755
11.278.2 Field Documentation . . . . .	755
11.279 CFE_TIME_AddAdjustCmd Struct Reference . . . . .	755
11.279.1 Detailed Description . . . . .	755
11.279.2 Field Documentation . . . . .	755
11.280 CFE_TIME_AddDelayCmd Struct Reference . . . . .	756
11.280.1 Detailed Description . . . . .	756
11.280.2 Field Documentation . . . . .	756
11.281 CFE_TIME_AddOneHzAdjustmentCmd Struct Reference . . . . .	756
11.281.1 Detailed Description . . . . .	757
11.281.2 Field Documentation . . . . .	757
11.282 CFE_TIME_DiagnosticTlm Struct Reference . . . . .	757
11.282.1 Detailed Description . . . . .	757
11.282.2 Field Documentation . . . . .	757
11.283 CFE_TIME_DiagnosticTlm_Payload Struct Reference . . . . .	757
11.283.1 Detailed Description . . . . .	759
11.283.2 Field Documentation . . . . .	759
11.284 CFE_TIME_FakeToneCmd Struct Reference . . . . .	766
11.284.1 Detailed Description . . . . .	766

---

11.284.2 Field Documentation . . . . .	766
11.285 CFE_TIME_HousekeepingTlm Struct Reference . . . . .	766
11.285.1 Detailed Description . . . . .	766
11.285.2 Field Documentation . . . . .	766
11.286 CFE_TIME_HousekeepingTlm_Payload Struct Reference . . . . .	767
11.286.1 Detailed Description . . . . .	767
11.286.2 Field Documentation . . . . .	767
11.287 CFE_TIME_LeapsCmd_Payload Struct Reference . . . . .	768
11.287.1 Detailed Description . . . . .	769
11.287.2 Field Documentation . . . . .	769
11.288 CFE_TIME_NoopCmd Struct Reference . . . . .	769
11.288.1 Detailed Description . . . . .	769
11.288.2 Field Documentation . . . . .	769
11.289 CFE_TIME_OneHzAdjustmentCmd_Payload Struct Reference . . . . .	769
11.289.1 Detailed Description . . . . .	770
11.289.2 Field Documentation . . . . .	770
11.290 CFE_TIME_OneHzCmd Struct Reference . . . . .	770
11.290.1 Detailed Description . . . . .	770
11.290.2 Field Documentation . . . . .	770
11.291 CFE_TIME_ResetCountersCmd Struct Reference . . . . .	770
11.291.1 Detailed Description . . . . .	770
11.291.2 Field Documentation . . . . .	771
11.292 CFE_TIME_SendDiagnosticCmd Struct Reference . . . . .	771
11.292.1 Detailed Description . . . . .	771
11.292.2 Field Documentation . . . . .	771
11.293 CFE_TIME_SendHkCmd Struct Reference . . . . .	771
11.293.1 Detailed Description . . . . .	771
11.293.2 Field Documentation . . . . .	771
11.294 CFE_TIME_SetLeapSecondsCmd Struct Reference . . . . .	772
11.294.1 Detailed Description . . . . .	772
11.294.2 Field Documentation . . . . .	772
11.295 CFE_TIME_SetMETCmd Struct Reference . . . . .	772
11.295.1 Detailed Description . . . . .	772
11.295.2 Field Documentation . . . . .	773
11.296 CFE_TIME_SetSignalCmd Struct Reference . . . . .	773
11.296.1 Detailed Description . . . . .	773
11.296.2 Field Documentation . . . . .	773
11.297 CFE_TIME_SetSourceCmd Struct Reference . . . . .	773
11.297.1 Detailed Description . . . . .	774

11.297.2 Field Documentation . . . . .	774
11.298 CFE_TIME_SetStateCmd Struct Reference . . . . .	774
11.298.1 Detailed Description . . . . .	774
11.298.2 Field Documentation . . . . .	774
11.299 CFE_TIME_SetSTCFCmd Struct Reference . . . . .	775
11.299.1 Detailed Description . . . . .	775
11.299.2 Field Documentation . . . . .	775
11.300 CFE_TIME_SetTimeCmd Struct Reference . . . . .	775
11.300.1 Detailed Description . . . . .	775
11.300.2 Field Documentation . . . . .	776
11.301 CFE_TIME_SignalCmd_Payload Struct Reference . . . . .	776
11.301.1 Detailed Description . . . . .	776
11.301.2 Field Documentation . . . . .	776
11.302 CFE_TIME_SourceCmd_Payload Struct Reference . . . . .	776
11.302.1 Detailed Description . . . . .	777
11.302.2 Field Documentation . . . . .	777
11.303 CFE_TIME_StateCmd_Payload Struct Reference . . . . .	777
11.303.1 Detailed Description . . . . .	777
11.303.2 Field Documentation . . . . .	777
11.304 CFE_TIME_SubAdjustCmd Struct Reference . . . . .	778
11.304.1 Detailed Description . . . . .	778
11.304.2 Field Documentation . . . . .	778
11.305 CFE_TIME_SubDelayCmd Struct Reference . . . . .	778
11.305.1 Detailed Description . . . . .	778
11.305.2 Field Documentation . . . . .	778
11.306 CFE_TIME_SubOneHzAdjustmentCmd Struct Reference . . . . .	779
11.306.1 Detailed Description . . . . .	779
11.306.2 Field Documentation . . . . .	779
11.307 CFE_TIME_SysTime Struct Reference . . . . .	779
11.307.1 Detailed Description . . . . .	780
11.307.2 Field Documentation . . . . .	780
11.308 CFE_TIME_TimeCmd_Payload Struct Reference . . . . .	780
11.308.1 Detailed Description . . . . .	780
11.308.2 Field Documentation . . . . .	780
11.309 CFE_TIME_ToneDataCmd Struct Reference . . . . .	781
11.309.1 Detailed Description . . . . .	781
11.309.2 Field Documentation . . . . .	781
11.310 CFE_TIME_ToneDataCmd_Payload Struct Reference . . . . .	781
11.310.1 Detailed Description . . . . .	781

---

11.310.2 Field Documentation . . . . .	782
11.311 CFE_TIME_ToneSignalCmd Struct Reference . . . . .	782
11.311.1 Detailed Description . . . . .	782
11.311.2 Field Documentation . . . . .	782
11.312 OS_bin_sem_prop_t Struct Reference . . . . .	782
11.312.1 Detailed Description . . . . .	783
11.312.2 Field Documentation . . . . .	783
11.313 OS_condvar_prop_t Struct Reference . . . . .	783
11.313.1 Detailed Description . . . . .	783
11.313.2 Field Documentation . . . . .	783
11.314 OS_count_sem_prop_t Struct Reference . . . . .	784
11.314.1 Detailed Description . . . . .	784
11.314.2 Field Documentation . . . . .	784
11.315 os_dirent_t Struct Reference . . . . .	784
11.315.1 Detailed Description . . . . .	784
11.315.2 Field Documentation . . . . .	785
11.316 OS_FdSet Struct Reference . . . . .	785
11.316.1 Detailed Description . . . . .	785
11.316.2 Field Documentation . . . . .	785
11.317 OS_file_prop_t Struct Reference . . . . .	785
11.317.1 Detailed Description . . . . .	786
11.317.2 Field Documentation . . . . .	786
11.318 os_fsinfo_t Struct Reference . . . . .	786
11.318.1 Detailed Description . . . . .	786
11.318.2 Field Documentation . . . . .	786
11.319 os_fstat_t Struct Reference . . . . .	787
11.319.1 Detailed Description . . . . .	787
11.319.2 Field Documentation . . . . .	787
11.320 OS_heap_prop_t Struct Reference . . . . .	788
11.320.1 Detailed Description . . . . .	788
11.320.2 Field Documentation . . . . .	788
11.321 OS_module_address_t Struct Reference . . . . .	788
11.321.1 Detailed Description . . . . .	789
11.321.2 Field Documentation . . . . .	789
11.322 OS_module_prop_t Struct Reference . . . . .	789
11.322.1 Detailed Description . . . . .	790
11.322.2 Field Documentation . . . . .	790
11.323 OS_mut_sem_prop_t Struct Reference . . . . .	790
11.323.1 Detailed Description . . . . .	790

11.323.2 Field Documentation . . . . .	790
11.324 OS_queue_prop_t Struct Reference . . . . .	791
11.324.1 Detailed Description . . . . .	791
11.324.2 Field Documentation . . . . .	791
11.325 OS_rwlock_prop_t Struct Reference . . . . .	791
11.325.1 Detailed Description . . . . .	791
11.325.2 Field Documentation . . . . .	792
11.326 OS_SockAddr_t Struct Reference . . . . .	792
11.326.1 Detailed Description . . . . .	792
11.326.2 Field Documentation . . . . .	792
11.327 OS_SockAddrData_t Union Reference . . . . .	792
11.327.1 Detailed Description . . . . .	793
11.327.2 Field Documentation . . . . .	793
11.328 OS_socket_optval Union Reference . . . . .	793
11.328.1 Detailed Description . . . . .	793
11.328.2 Field Documentation . . . . .	794
11.329 OS_socket_prop_t Struct Reference . . . . .	794
11.329.1 Detailed Description . . . . .	794
11.329.2 Field Documentation . . . . .	794
11.330 OS_static_symbol_record_t Struct Reference . . . . .	794
11.330.1 Detailed Description . . . . .	795
11.330.2 Field Documentation . . . . .	795
11.331 OS_statvfs_t Struct Reference . . . . .	795
11.331.1 Detailed Description . . . . .	795
11.331.2 Field Documentation . . . . .	795
11.332 OS_task_prop_t Struct Reference . . . . .	796
11.332.1 Detailed Description . . . . .	796
11.332.2 Field Documentation . . . . .	796
11.333 OS_time_t Struct Reference . . . . .	796
11.333.1 Detailed Description . . . . .	797
11.333.2 Field Documentation . . . . .	797
11.334 OS_timebase_prop_t Struct Reference . . . . .	797
11.334.1 Detailed Description . . . . .	797
11.334.2 Field Documentation . . . . .	797
11.335 OS_timer_prop_t Struct Reference . . . . .	798
11.335.1 Detailed Description . . . . .	798
11.335.2 Field Documentation . . . . .	798
<b>12 File Documentation</b>	<b>799</b>

---

12.1 apps/cf/config/default_cf_extern_typedefs.h File Reference . . . . .	799
12.1.1 Detailed Description . . . . .	799
12.1.2 Typedef Documentation . . . . .	799
12.1.3 Enumeration Type Documentation . . . . .	800
12.2 apps/cf/config/default_cf_fcncode_values.h File Reference . . . . .	801
12.2.1 Detailed Description . . . . .	801
12.2.2 Macro Definition Documentation . . . . .	801
12.2.3 Enumeration Type Documentation . . . . .	802
12.3 apps/cf/config/default_cf_interface_cfg_values.h File Reference . . . . .	802
12.3.1 Detailed Description . . . . .	802
12.3.2 Macro Definition Documentation . . . . .	802
12.4 apps/cf/config/default_cf_internal_cfg_values.h File Reference . . . . .	803
12.4.1 Detailed Description . . . . .	803
12.4.2 Macro Definition Documentation . . . . .	803
12.5 apps/cf/config/default_cf_mission_cfg.h File Reference . . . . .	803
12.5.1 Detailed Description . . . . .	803
12.5.2 Macro Definition Documentation . . . . .	803
12.6 apps/cf/config/default_cf_msg.h File Reference . . . . .	804
12.6.1 Detailed Description . . . . .	804
12.6.2 Macro Definition Documentation . . . . .	804
12.7 apps/cf/config/default_cf_msgdefs.h File Reference . . . . .	804
12.7.1 Detailed Description . . . . .	806
12.8 apps/cf/config/default_cf_msgid_values.h File Reference . . . . .	806
12.8.1 Detailed Description . . . . .	807
12.8.2 Macro Definition Documentation . . . . .	807
12.9 apps/cf/config/default_cf_msghids.h File Reference . . . . .	807
12.9.1 Detailed Description . . . . .	807
12.10 apps/cf/config/default_cf_msgstruct.h File Reference . . . . .	807
12.10.1 Detailed Description . . . . .	810
12.11 apps/cf/config/default_cf_platform_cfg.h File Reference . . . . .	810
12.11.1 Detailed Description . . . . .	810
12.11.2 Macro Definition Documentation . . . . .	810
12.12 apps/cf/config/default_cf_tbl.h File Reference . . . . .	811
12.12.1 Detailed Description . . . . .	812
12.13 apps/cf/config/default_cf_tbldefs.h File Reference . . . . .	812
12.13.1 Detailed Description . . . . .	812
12.13.2 Typedef Documentation . . . . .	812
12.14 apps/cf/config/default_cf_tblstruct.h File Reference . . . . .	812
12.14.1 Detailed Description . . . . .	813

12.14.2 Typedef Documentation . . . . .	813
12.15 apps/cf/config/default_cf_topicid_values.h File Reference . . . . .	813
12.15.1 Detailed Description . . . . .	813
12.15.2 Macro Definition Documentation . . . . .	813
12.16 apps/cf/config/eds_cf_extern_typedefs.h File Reference . . . . .	813
12.16.1 Detailed Description . . . . .	814
12.16.2 Macro Definition Documentation . . . . .	814
12.16.3 Typedef Documentation . . . . .	814
12.17 apps/cf/config/eds_cf_fcncode_values.h File Reference . . . . .	815
12.17.1 Detailed Description . . . . .	815
12.17.2 Macro Definition Documentation . . . . .	815
12.18 apps/cf/config/eds_cf_interface_cfg_values.h File Reference . . . . .	815
12.18.1 Detailed Description . . . . .	815
12.18.2 Macro Definition Documentation . . . . .	815
12.19 apps/cf/config/eds_cf_msgdefs.h File Reference . . . . .	816
12.19.1 Detailed Description . . . . .	816
12.20 apps/cf/config/eds_cf_msgstruct.h File Reference . . . . .	816
12.20.1 Detailed Description . . . . .	816
12.21 apps/cf/config/eds_cf_tbldefs.h File Reference . . . . .	816
12.21.1 Detailed Description . . . . .	816
12.22 apps/cf/config/eds_cf_tblstruct.h File Reference . . . . .	816
12.22.1 Detailed Description . . . . .	816
12.23 apps/cf/config/eds_cf_topicid_values.h File Reference . . . . .	816
12.23.1 Detailed Description . . . . .	816
12.23.2 Macro Definition Documentation . . . . .	816
12.24 apps/cf/docs/dox_src/cfs_cf.dox File Reference . . . . .	817
12.25 apps/cf/fsw/inc(cf_eventids.h File Reference . . . . .	817
12.25.1 Detailed Description . . . . .	823
12.26 apps/cf/fsw/inc(cf_fcncodes.h File Reference . . . . .	823
12.26.1 Detailed Description . . . . .	824
12.27 apps/cf/fsw/inc(cf_interface_cfg.h File Reference . . . . .	824
12.27.1 Detailed Description . . . . .	824
12.28 apps/cf/fsw/inc(cf_internal_cfg.h File Reference . . . . .	825
12.28.1 Detailed Description . . . . .	826
12.29 apps/cf/fsw/inc(cf_perfids.h File Reference . . . . .	826
12.29.1 Detailed Description . . . . .	826
12.30 apps/cf/fsw/inc(cf_topicids.h File Reference . . . . .	826
12.30.1 Detailed Description . . . . .	827
12.30.2 Macro Definition Documentation . . . . .	827

---

12.31 apps/cf/fsw/src(cf_app.c File Reference . . . . .	829
12.31.1 Detailed Description . . . . .	830
12.31.2 Function Documentation . . . . .	830
12.31.3 Variable Documentation . . . . .	835
12.32 apps/cf/fsw/src(cf_app.h File Reference . . . . .	835
12.32.1 Detailed Description . . . . .	836
12.32.2 Macro Definition Documentation . . . . .	837
12.32.3 Function Documentation . . . . .	838
12.32.4 Variable Documentation . . . . .	843
12.33 apps/cf/fsw/src(cf_assert.h File Reference . . . . .	843
12.33.1 Detailed Description . . . . .	844
12.33.2 Macro Definition Documentation . . . . .	844
12.34 apps/cf/fsw/src(cf_cfdp.c File Reference . . . . .	844
12.34.1 Detailed Description . . . . .	847
12.34.2 Function Documentation . . . . .	847
12.35 apps/cf/fsw/src(cf_cfdp.h File Reference . . . . .	897
12.35.1 Detailed Description . . . . .	900
12.35.2 Typedef Documentation . . . . .	900
12.35.3 Function Documentation . . . . .	900
12.36 apps/cf/fsw/src(cf_cfdp_dispatch.c File Reference . . . . .	947
12.36.1 Function Documentation . . . . .	948
12.37 apps/cf/fsw/src(cf_cfdp_dispatch.h File Reference . . . . .	949
12.37.1 Detailed Description . . . . .	950
12.37.2 Typedef Documentation . . . . .	950
12.37.3 Function Documentation . . . . .	951
12.38 apps/cf/fsw/src(cf_cfdp_pdu.h File Reference . . . . .	953
12.38.1 Detailed Description . . . . .	955
12.38.2 Macro Definition Documentation . . . . .	956
12.38.3 Typedef Documentation . . . . .	956
12.38.4 Enumeration Type Documentation . . . . .	957
12.39 apps/cf/fsw/src(cf_cfdp_r.c File Reference . . . . .	960
12.39.1 Detailed Description . . . . .	961
12.39.2 Function Documentation . . . . .	961
12.40 apps/cf/fsw/src(cf_cfdp_r.h File Reference . . . . .	980
12.40.1 Detailed Description . . . . .	981
12.40.2 Function Documentation . . . . .	981
12.41 apps/cf/fsw/src(cf_cfdp_s.c File Reference . . . . .	997
12.41.1 Detailed Description . . . . .	998
12.41.2 Function Documentation . . . . .	998

---

12.42 apps/cf/fsw/src/cf_cfdp_s.h File Reference . . . . .	1013
12.42.1 Detailed Description . . . . .	1014
12.42.2 Function Documentation . . . . .	1014
12.43 apps/cf/fsw/src/cf_cfdp_sbintf.c File Reference . . . . .	1027
12.43.1 Detailed Description . . . . .	1027
12.43.2 Function Documentation . . . . .	1028
12.44 apps/cf/fsw/src/cf_cfdp_sbintf.h File Reference . . . . .	1031
12.44.1 Detailed Description . . . . .	1032
12.44.2 Typedef Documentation . . . . .	1032
12.44.3 Function Documentation . . . . .	1032
12.45 apps/cf/fsw/src/cf_cfdp_types.h File Reference . . . . .	1035
12.45.1 Detailed Description . . . . .	1038
12.45.2 Macro Definition Documentation . . . . .	1038
12.45.3 Typedef Documentation . . . . .	1038
12.45.4 Enumeration Type Documentation . . . . .	1040
12.46 apps/cf/fsw/src/cf_chunk.c File Reference . . . . .	1042
12.46.1 Detailed Description . . . . .	1043
12.46.2 Function Documentation . . . . .	1043
12.47 apps/cf/fsw/src/cf_chunk.h File Reference . . . . .	1052
12.47.1 Detailed Description . . . . .	1053
12.47.2 Typedef Documentation . . . . .	1053
12.47.3 Function Documentation . . . . .	1054
12.48 apps/cf/fsw/src/cf_clist.c File Reference . . . . .	1063
12.48.1 Detailed Description . . . . .	1063
12.48.2 Function Documentation . . . . .	1063
12.49 apps/cf/fsw/src/cf_clist.h File Reference . . . . .	1068
12.49.1 Detailed Description . . . . .	1069
12.49.2 Macro Definition Documentation . . . . .	1069
12.49.3 Typedef Documentation . . . . .	1069
12.49.4 Enumeration Type Documentation . . . . .	1070
12.49.5 Function Documentation . . . . .	1070
12.50 apps/cf/fsw/src/cf_cmd.c File Reference . . . . .	1074
12.50.1 Detailed Description . . . . .	1076
12.50.2 Function Documentation . . . . .	1076
12.51 apps/cf/fsw/src/cf_cmd.h File Reference . . . . .	1105
12.51.1 Detailed Description . . . . .	1108
12.51.2 Typedef Documentation . . . . .	1108
12.51.3 Enumeration Type Documentation . . . . .	1108
12.51.4 Function Documentation . . . . .	1109

---

12.52 apps/cf/fsw/src(cf_codec.c File Reference . . . . .	1137
12.52.1 Detailed Description . . . . .	1140
12.52.2 Macro Definition Documentation . . . . .	1140
12.52.3 Typedef Documentation . . . . .	1140
12.52.4 Function Documentation . . . . .	1140
12.52.5 Variable Documentation . . . . .	1165
12.53 apps/cf/fsw/src(cf_codec.h File Reference . . . . .	1168
12.53.1 Detailed Description . . . . .	1171
12.53.2 Macro Definition Documentation . . . . .	1171
12.53.3 Typedef Documentation . . . . .	1173
12.53.4 Function Documentation . . . . .	1173
12.54 apps/cf/fsw/src(cf_crc.c File Reference . . . . .	1198
12.54.1 Detailed Description . . . . .	1198
12.54.2 Function Documentation . . . . .	1198
12.55 apps/cf/fsw/src(cf_crc.h File Reference . . . . .	1199
12.55.1 Detailed Description . . . . .	1200
12.55.2 Typedef Documentation . . . . .	1200
12.55.3 Function Documentation . . . . .	1200
12.56 apps/cf/fsw/src(cf_dispatch.c File Reference . . . . .	1201
12.56.1 Detailed Description . . . . .	1202
12.56.2 Function Documentation . . . . .	1202
12.57 apps/cf/fsw/src(cf_dispatch.h File Reference . . . . .	1204
12.57.1 Detailed Description . . . . .	1204
12.57.2 Function Documentation . . . . .	1204
12.58 apps/cf/fsw/src(cf_eds_dispatch.c File Reference . . . . .	1206
12.58.1 Function Documentation . . . . .	1207
12.58.2 Variable Documentation . . . . .	1208
12.59 apps/cf/fsw/src(cf_logical_pdu.h File Reference . . . . .	1209
12.59.1 Detailed Description . . . . .	1210
12.59.2 Macro Definition Documentation . . . . .	1210
12.59.3 Typedef Documentation . . . . .	1211
12.60 apps/cf/fsw/src(cf_timer.c File Reference . . . . .	1213
12.60.1 Detailed Description . . . . .	1213
12.60.2 Function Documentation . . . . .	1213
12.61 apps/cf/fsw/src(cf_timer.h File Reference . . . . .	1215
12.61.1 Detailed Description . . . . .	1216
12.61.2 Typedef Documentation . . . . .	1216
12.61.3 Function Documentation . . . . .	1216
12.62 apps/cf/fsw/src(cf_utils.c File Reference . . . . .	1218

---

12.62.1 Detailed Description . . . . .	1219
12.62.2 Function Documentation . . . . .	1219
12.63 apps/cf/fsw/src(cf_utils.h File Reference . . . . .	1237
12.63.1 Detailed Description . . . . .	1239
12.63.2 Typedef Documentation . . . . .	1239
12.63.3 Function Documentation . . . . .	1240
12.64 apps/cf/fsw/src(cf_verify.h File Reference . . . . .	1260
12.64.1 Detailed Description . . . . .	1260
12.65 apps/cf/fsw/src(cf_version.h File Reference . . . . .	1260
12.65.1 Detailed Description . . . . .	1261
12.66 apps/cf/fsw/tables(cf_def_config.c File Reference . . . . .	1261
12.66.1 Detailed Description . . . . .	1261
12.66.2 Variable Documentation . . . . .	1261
12.67 build/osal_public_api/inc/osconfig.h File Reference . . . . .	1261
12.67.1 Macro Definition Documentation . . . . .	1263
12.68 cfe/cmake/sample_defs/cfe_perfids.h File Reference . . . . .	1267
12.68.1 Detailed Description . . . . .	1268
12.68.2 Macro Definition Documentation . . . . .	1268
12.69 cfe/cmake/sample_defs/example_2x32bit_cfe_es_memaddress.h File Reference . . . . .	1269
12.69.1 Detailed Description . . . . .	1270
12.69.2 Macro Definition Documentation . . . . .	1270
12.69.3 Function Documentation . . . . .	1271
12.70 cfe/cmake/sample_defs/example_32bit_cfe_es_memaddress.h File Reference . . . . .	1271
12.70.1 Detailed Description . . . . .	1272
12.70.2 Macro Definition Documentation . . . . .	1272
12.70.3 Typedef Documentation . . . . .	1272
12.71 cfe/cmake/sample_defs/example_64bit_cfe_es_memaddress.h File Reference . . . . .	1273
12.71.1 Detailed Description . . . . .	1273
12.71.2 Macro Definition Documentation . . . . .	1274
12.71.3 Typedef Documentation . . . . .	1274
12.72 cfe/cmake/sample_defs/example_mission_cfg.h File Reference . . . . .	1274
12.72.1 Detailed Description . . . . .	1275
12.72.2 Macro Definition Documentation . . . . .	1276
12.73 cfe/cmake/sample_defs/example_platform_cfg.h File Reference . . . . .	1285
12.73.1 Detailed Description . . . . .	1289
12.73.2 Macro Definition Documentation . . . . .	1289
12.74 cfe/docs/src/cfe_api.dox File Reference . . . . .	1329
12.75 cfe/docs/src/cfe_es.dox File Reference . . . . .	1329
12.76 cfe/docs/src/cfe_evs.dox File Reference . . . . .	1329

---

12.77 cfe/docs/src/cfe_frontpage.dox File Reference . . . . .	1329
12.78 cfe/docs/src/cfe_glossary.dox File Reference . . . . .	1329
12.79 cfe/docs/src/cfe_sb.dox File Reference . . . . .	1329
12.80 cfe/docs/src/cfe_tbl.dox File Reference . . . . .	1329
12.81 cfe/docs/src/cfe_time.dox File Reference . . . . .	1329
12.82 cfe/docs/src/cfe_xref.dox File Reference . . . . .	1329
12.83 cfe/docs/src/cfs_versions.dox File Reference . . . . .	1329
12.84 cfe/modules/config/fsw/inc/cfe_config_external.h File Reference . . . . .	1329
12.84.1 Detailed Description . . . . .	1329
12.84.2 Function Documentation . . . . .	1329
12.85 cfe/modules/config/fsw/inc/cfe_config_init.h File Reference . . . . .	1329
12.85.1 Detailed Description . . . . .	1329
12.85.2 Function Documentation . . . . .	1330
12.86 cfe/modules/config/fsw/inc/cfe_config_lookup.h File Reference . . . . .	1330
12.86.1 Detailed Description . . . . .	1330
12.86.2 Function Documentation . . . . .	1330
12.87 cfe/modules/config/fsw/inc/cfe_config_nametable.h File Reference . . . . .	1330
12.87.1 Detailed Description . . . . .	1330
12.87.2 Typedef Documentation . . . . .	1331
12.87.3 Variable Documentation . . . . .	1331
12.88 cfe/modules/config/fsw/inc/cfe_config_set.h File Reference . . . . .	1331
12.88.1 Detailed Description . . . . .	1331
12.88.2 Function Documentation . . . . .	1331
12.89 cfe/modules/config/fsw/inc/cfe_config_table.h File Reference . . . . .	1332
12.89.1 Detailed Description . . . . .	1332
12.89.2 Typedef Documentation . . . . .	1332
12.89.3 Enumeration Type Documentation . . . . .	1332
12.90 cfe/modules/core_api/config/default_cfe_core_api_basemsgid_values.h File Reference . . . . .	1333
12.90.1 Detailed Description . . . . .	1333
12.90.2 Macro Definition Documentation . . . . .	1333
12.91 cfe/modules/core_api/config/default_cfe_core_api_interface_cfg_values.h File Reference . . . . .	1333
12.91.1 Detailed Description . . . . .	1333
12.91.2 Macro Definition Documentation . . . . .	1333
12.92 cfe/modules/core_api/config/default_cfe_core_api_msgid_mapping.h File Reference . . . . .	1334
12.92.1 Detailed Description . . . . .	1334
12.92.2 Macro Definition Documentation . . . . .	1334
12.93 cfe/modules/core_api/config/default_cfe_mission_cfg.h File Reference . . . . .	1335
12.93.1 Detailed Description . . . . .	1336
12.94 cfe/modules/core_api/config/default_cfe_msgids.h File Reference . . . . .	1336

12.94.1 Detailed Description . . . . .	1336
12.95 cfe/modules/core_api/fsw/inc/cfe.h File Reference . . . . .	1336
12.95.1 Detailed Description . . . . .	1336
12.96 cfe/modules/core_api/fsw/inc/cfe_config.h File Reference . . . . .	1336
12.96.1 Detailed Description . . . . .	1337
12.96.2 Function Documentation . . . . .	1337
12.97 cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h File Reference . . . . .	1340
12.97.1 Detailed Description . . . . .	1341
12.97.2 Macro Definition Documentation . . . . .	1341
12.97.3 Typedef Documentation . . . . .	1341
12.98 cfe/modules/core_api/fsw/inc/cfe_core_api_base_msgids.h File Reference . . . . .	1341
12.98.1 Detailed Description . . . . .	1341
12.99 cfe/modules/core_api/fsw/inc/cfe_core_api_interface_cfg.h File Reference . . . . .	1342
12.99.1 Detailed Description . . . . .	1342
12.99.2 Macro Definition Documentation . . . . .	1342
12.100 cfe/modules/core_api/fsw/inc/cfe_endian.h File Reference . . . . .	1344
12.100.1 Detailed Description . . . . .	1345
12.100.2 Macro Definition Documentation . . . . .	1345
12.101 cfe/modules/core_api/fsw/inc/cfe_error.h File Reference . . . . .	1345
12.101.1 Detailed Description . . . . .	1351
12.101.2 Macro Definition Documentation . . . . .	1351
12.101.3 Typedef Documentation . . . . .	1353
12.101.4 Function Documentation . . . . .	1353
12.102 cfe/modules/core_api/fsw/inc/cfe_es.h File Reference . . . . .	1353
12.102.1 Detailed Description . . . . .	1356
12.102.2 Macro Definition Documentation . . . . .	1356
12.103 cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h File Reference . . . . .	1357
12.103.1 Detailed Description . . . . .	1358
12.103.2 Macro Definition Documentation . . . . .	1358
12.103.3 Typedef Documentation . . . . .	1360
12.103.4 Enumeration Type Documentation . . . . .	1361
12.104 cfe/modules/core_api/fsw/inc/cfe_evs.h File Reference . . . . .	1362
12.104.1 Detailed Description . . . . .	1363
12.104.2 Macro Definition Documentation . . . . .	1363
12.105 cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h File Reference . . . . .	1363
12.105.1 Detailed Description . . . . .	1364
12.105.2 Macro Definition Documentation . . . . .	1364
12.105.3 Typedef Documentation . . . . .	1365
12.106 cfe/modules/core_api/fsw/inc/cfe_fs.h File Reference . . . . .	1366

---

12.106.1 Detailed Description . . . . .	1366
12.107 cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h File Reference . . . . .	1366
12.107.1 Detailed Description . . . . .	1367
12.107.2 Typedef Documentation . . . . .	1367
12.107.3 Enumeration Type Documentation . . . . .	1368
12.108 cfe/modules/core_api/fsw/inc/cfe_msg.h File Reference . . . . .	1369
12.108.1 Detailed Description . . . . .	1371
12.109 cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h File Reference . . . . .	1371
12.109.1 Detailed Description . . . . .	1372
12.109.2 Macro Definition Documentation . . . . .	1372
12.109.3 Typedef Documentation . . . . .	1373
12.109.4 Enumeration Type Documentation . . . . .	1374
12.110 cfe/modules/core_api/fsw/inc/cfe_resourceid.h File Reference . . . . .	1375
12.110.1 Detailed Description . . . . .	1376
12.110.2 Macro Definition Documentation . . . . .	1376
12.110.3 Typedef Documentation . . . . .	1377
12.110.4 Function Documentation . . . . .	1378
12.111 cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h File Reference . . . . .	1382
12.111.1 Detailed Description . . . . .	1382
12.111.2 Macro Definition Documentation . . . . .	1383
12.112 cfe/modules/core_api/fsw/inc/cfe_sb.h File Reference . . . . .	1383
12.112.1 Detailed Description . . . . .	1385
12.112.2 Macro Definition Documentation . . . . .	1385
12.113 cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h File Reference . . . . .	1386
12.113.1 Detailed Description . . . . .	1386
12.113.2 Macro Definition Documentation . . . . .	1387
12.113.3 Typedef Documentation . . . . .	1389
12.114 cfe/modules/core_api/fsw/inc/cfe_tbl.h File Reference . . . . .	1389
12.114.1 Detailed Description . . . . .	1390
12.114.2 Function Documentation . . . . .	1390
12.115 cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h File Reference . . . . .	1391
12.115.1 Detailed Description . . . . .	1392
12.115.2 Macro Definition Documentation . . . . .	1392
12.115.3 Typedef Documentation . . . . .	1393
12.115.4 Enumeration Type Documentation . . . . .	1394
12.116 cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h File Reference . . . . .	1394
12.116.1 Detailed Description . . . . .	1394
12.116.2 Macro Definition Documentation . . . . .	1395
12.116.3 Typedef Documentation . . . . .	1395

---

12.117 cfe/modules/core_api/fsw/inc/cfe_time.h File Reference . . . . .	1395
12.117.1 Detailed Description . . . . .	1397
12.117.2 Macro Definition Documentation . . . . .	1397
12.118 cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h File Reference . . . . .	1397
12.118.1 Detailed Description . . . . .	1398
12.118.2 Macro Definition Documentation . . . . .	1398
12.118.3 Typedef Documentation . . . . .	1398
12.118.4 Enumeration Type Documentation . . . . .	1398
12.119 cfe/modules/core_api/fsw/inc/cfe_version.h File Reference . . . . .	1399
12.119.1 Detailed Description . . . . .	1399
12.119.2 Macro Definition Documentation . . . . .	1400
12.120 cfe/modules/es/config/default_cfe_es_extern_typedefs.h File Reference . . . . .	1401
12.120.1 Detailed Description . . . . .	1403
12.120.2 Typedef Documentation . . . . .	1403
12.120.3 Enumeration Type Documentation . . . . .	1406
12.121 cfe/modules/es/config/default_cfe_es_fcncode_values.h File Reference . . . . .	1408
12.121.1 Detailed Description . . . . .	1409
12.121.2 Macro Definition Documentation . . . . .	1409
12.121.3 Enumeration Type Documentation . . . . .	1409
12.122 cfe/modules/es/config/default_cfe_es_interface_cfg_values.h File Reference . . . . .	1410
12.122.1 Detailed Description . . . . .	1410
12.122.2 Macro Definition Documentation . . . . .	1410
12.123 cfe/modules/es/config/default_cfe_es_internal_cfg_values.h File Reference . . . . .	1410
12.123.1 Detailed Description . . . . .	1410
12.123.2 Macro Definition Documentation . . . . .	1410
12.124 cfe/modules/es/config/default_cfe_es_memaddress.h File Reference . . . . .	1410
12.124.1 Detailed Description . . . . .	1411
12.124.2 Macro Definition Documentation . . . . .	1411
12.124.3 Typedef Documentation . . . . .	1412
12.125 cfe/modules/es/config/default_cfe_es_mission_cfg.h File Reference . . . . .	1412
12.125.1 Detailed Description . . . . .	1412
12.126 cfe/modules/es/config/default_cfe_es_msg.h File Reference . . . . .	1412
12.126.1 Detailed Description . . . . .	1412
12.127 cfe/modules/es/config/default_cfe_es_msghdefs.h File Reference . . . . .	1413
12.127.1 Detailed Description . . . . .	1414
12.127.2 Typedef Documentation . . . . .	1414
12.127.3 Enumeration Type Documentation . . . . .	1416
12.128 cfe/modules/es/config/default_cfe_es_msghid_values.h File Reference . . . . .	1416
12.128.1 Detailed Description . . . . .	1417

---

12.128.2 Macro Definition Documentation . . . . .	1417
12.129 cfe/modules/es/config/default_cfe_es_msgids.h File Reference . . . . .	1417
12.129.1 Detailed Description . . . . .	1417
12.129.2 Macro Definition Documentation . . . . .	1417
12.130 cfe/modules/es/config/default_cfe_es_msgstruct.h File Reference . . . . .	1418
12.130.1 Detailed Description . . . . .	1420
12.130.2 Typedef Documentation . . . . .	1420
12.131 cfe/modules/es/config/default_cfe_es_platform_cfg.h File Reference . . . . .	1422
12.131.1 Detailed Description . . . . .	1422
12.132 cfe/modules/es/config/default_cfe_es_topicid_values.h File Reference . . . . .	1422
12.132.1 Detailed Description . . . . .	1422
12.132.2 Macro Definition Documentation . . . . .	1422
12.133 cfe/modules/es/fsw/inc/cfe_es_eventids.h File Reference . . . . .	1422
12.133.1 Detailed Description . . . . .	1426
12.133.2 Macro Definition Documentation . . . . .	1426
12.134 cfe/modules/es/fsw/inc/cfe_es_fnpcodes.h File Reference . . . . .	1448
12.134.1 Detailed Description . . . . .	1448
12.134.2 Macro Definition Documentation . . . . .	1448
12.135 cfe/modules/es/fsw/inc/cfe_es_interface_cfg.h File Reference . . . . .	1469
12.135.1 Detailed Description . . . . .	1470
12.135.2 Macro Definition Documentation . . . . .	1470
12.136 cfe/modules/es/fsw/inc/cfe_es_internal_cfg.h File Reference . . . . .	1473
12.136.1 Detailed Description . . . . .	1478
12.136.2 Macro Definition Documentation . . . . .	1478
12.137 cfe/modules/es/fsw/inc/cfe_es_topicids.h File Reference . . . . .	1505
12.137.1 Detailed Description . . . . .	1505
12.137.2 Macro Definition Documentation . . . . .	1505
12.138 cfe/modules/evs/config/default_cfe_evs_extern_typedefs.h File Reference . . . . .	1506
12.138.1 Detailed Description . . . . .	1507
12.138.2 Typedef Documentation . . . . .	1507
12.138.3 Enumeration Type Documentation . . . . .	1508
12.139 cfe/modules/evs/config/default_cfe_evs_fnicode_values.h File Reference . . . . .	1509
12.139.1 Detailed Description . . . . .	1510
12.139.2 Macro Definition Documentation . . . . .	1510
12.139.3 Enumeration Type Documentation . . . . .	1510
12.140 cfe/modules/evs/config/default_cfe_evs_interface_cfg_values.h File Reference . . . . .	1511
12.140.1 Detailed Description . . . . .	1511
12.140.2 Macro Definition Documentation . . . . .	1511
12.141 cfe/modules/evs/config/default_cfe_evs_internal_cfg_values.h File Reference . . . . .	1511

12.141.1 Detailed Description . . . . .	1511
12.141.2 Macro Definition Documentation . . . . .	1511
12.142 cfe/modules/evs/config/default_cfe_evs_mission_cfg.h File Reference . . . . .	1511
12.142.1 Detailed Description . . . . .	1512
12.143 cfe/modules/evs/config/default_cfe_evs_msg.h File Reference . . . . .	1512
12.143.1 Detailed Description . . . . .	1512
12.144 cfe/modules/evs/config/default_cfe_evs_msgdefs.h File Reference . . . . .	1512
12.144.1 Detailed Description . . . . .	1513
12.144.2 Macro Definition Documentation . . . . .	1514
12.144.3 Typedef Documentation . . . . .	1514
12.145 cfe/modules/evs/config/default_cfe_evsmsgid_values.h File Reference . . . . .	1516
12.145.1 Detailed Description . . . . .	1516
12.145.2 Macro Definition Documentation . . . . .	1516
12.146 cfe/modules/evs/config/default_cfe_evs_msgids.h File Reference . . . . .	1516
12.146.1 Detailed Description . . . . .	1516
12.146.2 Macro Definition Documentation . . . . .	1516
12.147 cfe/modules/evs/config/default_cfe_evs_msgstruct.h File Reference . . . . .	1517
12.147.1 Detailed Description . . . . .	1518
12.147.2 Typedef Documentation . . . . .	1518
12.148 cfe/modules/evs/config/default_cfe_evs_platform_cfg.h File Reference . . . . .	1520
12.148.1 Detailed Description . . . . .	1520
12.149 cfe/modules/evs/config/default_cfe_evs_topicid_values.h File Reference . . . . .	1520
12.149.1 Detailed Description . . . . .	1521
12.149.2 Macro Definition Documentation . . . . .	1521
12.150 cfe/modules/evs/fsw/inc/cfe_evs_eventids.h File Reference . . . . .	1521
12.150.1 Detailed Description . . . . .	1522
12.150.2 Macro Definition Documentation . . . . .	1522
12.151 cfe/modules/evs/fsw/inc/cfe_evs_fncodes.h File Reference . . . . .	1533
12.151.1 Detailed Description . . . . .	1533
12.151.2 Macro Definition Documentation . . . . .	1533
12.152 cfe/modules/evs/fsw/inc/cfe_evs_interface_cfg.h File Reference . . . . .	1551
12.152.1 Detailed Description . . . . .	1552
12.152.2 Macro Definition Documentation . . . . .	1552
12.153 cfe/modules/evs/fsw/inc/cfe_evs_internal_cfg.h File Reference . . . . .	1552
12.153.1 Detailed Description . . . . .	1553
12.153.2 Macro Definition Documentation . . . . .	1553
12.154 cfe/modules/evs/fsw/inc/cfe_evs_topicids.h File Reference . . . . .	1558
12.154.1 Detailed Description . . . . .	1558
12.154.2 Macro Definition Documentation . . . . .	1558

---

12.155 cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h File Reference . . . . .	1560
12.155.1 Detailed Description . . . . .	1560
12.156 cfe/modules/fs/config/default_cfe_fs_filedef.h File Reference . . . . .	1560
12.156.1 Detailed Description . . . . .	1561
12.156.2 Typedef Documentation . . . . .	1561
12.156.3 Enumeration Type Documentation . . . . .	1561
12.157 cfe/modules/fs/config/default_cfe_fs_interface_cfg_values.h File Reference . . . . .	1562
12.157.1 Detailed Description . . . . .	1562
12.157.2 Macro Definition Documentation . . . . .	1562
12.158 cfe/modules/fs/config/default_cfe_fs_mission_cfg.h File Reference . . . . .	1562
12.158.1 Detailed Description . . . . .	1563
12.159 cfe/modules/fs/fsw/inc/cfe_fs_interface_cfg.h File Reference . . . . .	1563
12.159.1 Detailed Description . . . . .	1563
12.159.2 Macro Definition Documentation . . . . .	1563
12.160 cfe/modules/msg/fsw/inc/ccsds_hdr.h File Reference . . . . .	1564
12.160.1 Detailed Description . . . . .	1564
12.160.2 Typedef Documentation . . . . .	1564
12.161 cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h File Reference . . . . .	1564
12.161.1 Detailed Description . . . . .	1565
12.162 cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h File Reference . . . . .	1565
12.162.1 Detailed Description . . . . .	1566
12.162.2 Macro Definition Documentation . . . . .	1566
12.163 cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h File Reference . . . . .	1566
12.163.1 Detailed Description . . . . .	1567
12.163.2 Typedef Documentation . . . . .	1567
12.163.3 Enumeration Type Documentation . . . . .	1568
12.164 cfe/modules/sb/config/default_cfe_sb_fcncode_values.h File Reference . . . . .	1568
12.164.1 Detailed Description . . . . .	1568
12.164.2 Macro Definition Documentation . . . . .	1568
12.164.3 Enumeration Type Documentation . . . . .	1569
12.165 cfe/modules/sb/config/default_cfe_sb_interface_cfg_values.h File Reference . . . . .	1569
12.165.1 Detailed Description . . . . .	1569
12.165.2 Macro Definition Documentation . . . . .	1569
12.166 cfe/modules/sb/config/default_cfe_sb_internal_cfg_values.h File Reference . . . . .	1570
12.166.1 Detailed Description . . . . .	1570
12.166.2 Macro Definition Documentation . . . . .	1570
12.167 cfe/modules/sb/config/default_cfe_sb_mission_cfg.h File Reference . . . . .	1570
12.167.1 Detailed Description . . . . .	1570
12.168 cfe/modules/sb/config/default_cfe_sb_msg.h File Reference . . . . .	1570

---

12.168.1 Detailed Description . . . . .	1570
12.169 cfe/modules/sb/config/default_cfe_sb_msgdefs.h File Reference . . . . .	1571
12.169.1 Detailed Description . . . . .	1572
12.169.2 Typedef Documentation . . . . .	1572
12.170 cfe/modules/sb/config/default_cfe_sbmsgid_values.h File Reference . . . . .	1573
12.170.1 Detailed Description . . . . .	1573
12.170.2 Macro Definition Documentation . . . . .	1573
12.171 cfe/modules/sb/config/default_cfe_sb_msgids.h File Reference . . . . .	1574
12.171.1 Detailed Description . . . . .	1574
12.171.2 Macro Definition Documentation . . . . .	1574
12.172 cfe/modules/sb/config/default_cfe_sb_msgstruct.h File Reference . . . . .	1575
12.172.1 Detailed Description . . . . .	1576
12.172.2 Typedef Documentation . . . . .	1576
12.173 cfe/modules/sb/config/default_cfe_sb_platform_cfg.h File Reference . . . . .	1577
12.173.1 Detailed Description . . . . .	1577
12.174 cfe/modules/sb/config/default_cfe_sb_topicid_values.h File Reference . . . . .	1577
12.174.1 Detailed Description . . . . .	1577
12.174.2 Macro Definition Documentation . . . . .	1577
12.175 cfe/modules/sb/fsw/inc/cfe_sb_eventids.h File Reference . . . . .	1577
12.175.1 Detailed Description . . . . .	1580
12.175.2 Macro Definition Documentation . . . . .	1580
12.176 cfe/modules/sb/fsw/inc/cfe_sb_fcncodes.h File Reference . . . . .	1597
12.176.1 Detailed Description . . . . .	1597
12.176.2 Macro Definition Documentation . . . . .	1597
12.177 cfe/modules/sb/fsw/inc/cfe_sb_interface_cfg.h File Reference . . . . .	1606
12.177.1 Detailed Description . . . . .	1606
12.177.2 Macro Definition Documentation . . . . .	1607
12.178 cfe/modules/sb/fsw/inc/cfe_sb_internal_cfg.h File Reference . . . . .	1608
12.178.1 Detailed Description . . . . .	1610
12.178.2 Macro Definition Documentation . . . . .	1610
12.179 cfe/modules/sb/fsw/inc/cfe_sb_topicids.h File Reference . . . . .	1623
12.179.1 Detailed Description . . . . .	1623
12.179.2 Macro Definition Documentation . . . . .	1623
12.180 cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h File Reference . . . . .	1625
12.180.1 Detailed Description . . . . .	1626
12.180.2 Typedef Documentation . . . . .	1626
12.180.3 Enumeration Type Documentation . . . . .	1627
12.181 cfe/modules/tbl/config/default_cfe_tbl_fcncode_values.h File Reference . . . . .	1627
12.181.1 Detailed Description . . . . .	1627

---

12.181.2 Macro Definition Documentation . . . . .	1627
12.181.3 Enumeration Type Documentation . . . . .	1627
12.182 cfe/modules/tbl/config/default_cfe_tbl_interface_cfg_values.h File Reference . . . . .	1628
12.182.1 Detailed Description . . . . .	1628
12.182.2 Macro Definition Documentation . . . . .	1628
12.183 cfe/modules/tbl/config/default_cfe_tbl_internal_cfg_values.h File Reference . . . . .	1628
12.183.1 Detailed Description . . . . .	1628
12.183.2 Macro Definition Documentation . . . . .	1629
12.184 cfe/modules/tbl/config/default_cfe_tbl_mission_cfg.h File Reference . . . . .	1629
12.184.1 Detailed Description . . . . .	1629
12.184.2 Macro Definition Documentation . . . . .	1629
12.185 cfe/modules/tbl/config/default_cfe_tbl_msg.h File Reference . . . . .	1630
12.185.1 Detailed Description . . . . .	1630
12.186 cfe/modules/tbl/config/default_cfe_tbl_msghDefs.h File Reference . . . . .	1630
12.186.1 Detailed Description . . . . .	1631
12.186.2 Typedef Documentation . . . . .	1632
12.187 cfe/modules/tbl/config/default_cfe_tbl_msqid_values.h File Reference . . . . .	1633
12.187.1 Detailed Description . . . . .	1633
12.187.2 Macro Definition Documentation . . . . .	1633
12.188 cfe/modules/tbl/config/default_cfe_tbl_msqidS.h File Reference . . . . .	1633
12.188.1 Detailed Description . . . . .	1633
12.188.2 Macro Definition Documentation . . . . .	1634
12.189 cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h File Reference . . . . .	1634
12.189.1 Detailed Description . . . . .	1635
12.189.2 Typedef Documentation . . . . .	1635
12.190 cfe/modules/tbl/config/default_cfe_tbl_platform_cfg.h File Reference . . . . .	1636
12.190.1 Detailed Description . . . . .	1637
12.190.2 Macro Definition Documentation . . . . .	1638
12.191 cfe/modules/tbl/config/default_cfe_tbl_topoCID_values.h File Reference . . . . .	1645
12.191.1 Detailed Description . . . . .	1645
12.191.2 Macro Definition Documentation . . . . .	1645
12.192 cfe/modules/tbl/fsw/inc/cfe_tbl_eventids.h File Reference . . . . .	1645
12.192.1 Detailed Description . . . . .	1648
12.192.2 Macro Definition Documentation . . . . .	1648
12.193 cfe/modules/tbl/fsw/inc/cfe_tbl_fcncodes.h File Reference . . . . .	1665
12.193.1 Detailed Description . . . . .	1665
12.193.2 Macro Definition Documentation . . . . .	1666
12.194 cfe/modules/tbl/fsw/inc/cfe_tbl_interface_cfg.h File Reference . . . . .	1674
12.194.1 Detailed Description . . . . .	1675

---

12.194.2 Macro Definition Documentation . . . . .	1675
12.195 cfe/modules/tbl/fsw/inc/cfe_tbl_internal_cfg.h File Reference . . . . .	1676
12.195.1 Detailed Description . . . . .	1677
12.195.2 Macro Definition Documentation . . . . .	1677
12.196 cfe/modules/tbl/fsw/inc/cfe_tbl_topicids.h File Reference . . . . .	1685
12.196.1 Detailed Description . . . . .	1685
12.196.2 Macro Definition Documentation . . . . .	1685
12.197 cfe/modules/time/config/default_cfe_time_extern_typedefs.h File Reference . . . . .	1686
12.197.1 Detailed Description . . . . .	1687
12.197.2 Typedef Documentation . . . . .	1687
12.197.3 Enumeration Type Documentation . . . . .	1689
12.198 cfe/modules/time/config/default_cfe_time_fcncode_values.h File Reference . . . . .	1691
12.198.1 Detailed Description . . . . .	1691
12.198.2 Macro Definition Documentation . . . . .	1691
12.198.3 Enumeration Type Documentation . . . . .	1691
12.199 cfe/modules/time/config/default_cfe_time_interface_cfg_values.h File Reference . . . . .	1692
12.199.1 Detailed Description . . . . .	1692
12.199.2 Macro Definition Documentation . . . . .	1692
12.200 cfe/modules/time/config/default_cfe_time_internal_cfg_values.h File Reference . . . . .	1692
12.200.1 Detailed Description . . . . .	1693
12.200.2 Macro Definition Documentation . . . . .	1693
12.201 cfe/modules/time/config/default_cfe_time_mission_cfg.h File Reference . . . . .	1693
12.201.1 Detailed Description . . . . .	1693
12.202 cfe/modules/time/config/default_cfe_time_msg.h File Reference . . . . .	1693
12.202.1 Detailed Description . . . . .	1693
12.203 cfe/modules/time/config/default_cfe_time_msgdefs.h File Reference . . . . .	1694
12.203.1 Detailed Description . . . . .	1695
12.203.2 Typedef Documentation . . . . .	1695
12.204 cfe/modules/time/config/default_cfe_time_msqid_values.h File Reference . . . . .	1696
12.204.1 Detailed Description . . . . .	1696
12.204.2 Macro Definition Documentation . . . . .	1696
12.205 cfe/modules/time/config/default_cfe_time_msqid.h File Reference . . . . .	1697
12.205.1 Detailed Description . . . . .	1697
12.205.2 Macro Definition Documentation . . . . .	1697
12.206 cfe/modules/time/config/default_cfe_time_msgstruct.h File Reference . . . . .	1698
12.206.1 Detailed Description . . . . .	1699
12.206.2 Typedef Documentation . . . . .	1699
12.207 cfe/modules/time/config/default_cfe_time_platform_cfg.h File Reference . . . . .	1701
12.207.1 Detailed Description . . . . .	1701

---

12.208 cfe/modules/time/config/default_cfe_time_topid_values.h File Reference . . . . .	1701
12.208.1 Detailed Description . . . . .	1701
12.208.2 Macro Definition Documentation . . . . .	1701
12.209 cfe/modules/time/fsw/inc/cfe_time_eventids.h File Reference . . . . .	1702
12.209.1 Detailed Description . . . . .	1703
12.209.2 Macro Definition Documentation . . . . .	1703
12.210 cfe/modules/time/fsw/inc/cfe_time_fcncodes.h File Reference . . . . .	1712
12.210.1 Detailed Description . . . . .	1713
12.210.2 Macro Definition Documentation . . . . .	1713
12.211 cfe/modules/time/fsw/inc/cfe_time_interface_cfg.h File Reference . . . . .	1727
12.211.1 Detailed Description . . . . .	1729
12.211.2 Macro Definition Documentation . . . . .	1729
12.212 cfe/modules/time/fsw/inc/cfe_time_internal_cfg.h File Reference . . . . .	1735
12.212.1 Detailed Description . . . . .	1736
12.212.2 Macro Definition Documentation . . . . .	1736
12.213 cfe/modules/time/fsw/inc/cfe_time_topicids.h File Reference . . . . .	1744
12.213.1 Detailed Description . . . . .	1744
12.213.2 Macro Definition Documentation . . . . .	1744
12.214 osal/docs/srcosal_frontpage.dox File Reference . . . . .	1746
12.215 osal/docs/srcosal_fs.dox File Reference . . . . .	1746
12.216 osal/docs/srcosal_timer.dox File Reference . . . . .	1746
12.217 osal/src/os/inc/common_types.h File Reference . . . . .	1746
12.217.1 Detailed Description . . . . .	1747
12.217.2 Macro Definition Documentation . . . . .	1747
12.217.3 Typedef Documentation . . . . .	1748
12.217.4 Function Documentation . . . . .	1750
12.218 osal/src/os/inc/osapi-binsem.h File Reference . . . . .	1751
12.218.1 Detailed Description . . . . .	1752
12.219 osal/src/os/inc/osapi-bsp.h File Reference . . . . .	1752
12.219.1 Detailed Description . . . . .	1752
12.220 osal/src/os/inc/osapi-clock.h File Reference . . . . .	1752
12.220.1 Detailed Description . . . . .	1754
12.220.2 Macro Definition Documentation . . . . .	1754
12.220.3 Enumeration Type Documentation . . . . .	1755
12.221 osal/src/os/inc/osapi-common.h File Reference . . . . .	1755
12.221.1 Detailed Description . . . . .	1756
12.221.2 Typedef Documentation . . . . .	1756
12.221.3 Enumeration Type Documentation . . . . .	1756
12.222 osal/src/os/inc/osapi-condvar.h File Reference . . . . .	1757

---

12.222.1 Detailed Description . . . . .	1758
12.223 osal/src/os/inc/osapi-constants.h File Reference . . . . .	1758
12.223.1 Detailed Description . . . . .	1758
12.223.2 Macro Definition Documentation . . . . .	1758
12.224 osal/src/os/inc/osapi-countsem.h File Reference . . . . .	1759
12.224.1 Detailed Description . . . . .	1759
12.225 osal/src/os/inc/osapi-dir.h File Reference . . . . .	1759
12.225.1 Detailed Description . . . . .	1760
12.225.2 Macro Definition Documentation . . . . .	1760
12.226 osal/src/os/inc/osapi-error.h File Reference . . . . .	1760
12.226.1 Detailed Description . . . . .	1763
12.226.2 Macro Definition Documentation . . . . .	1763
12.226.3 Typedef Documentation . . . . .	1763
12.227 osal/src/os/inc/osapi-file.h File Reference . . . . .	1763
12.227.1 Detailed Description . . . . .	1765
12.227.2 Macro Definition Documentation . . . . .	1765
12.227.3 Enumeration Type Documentation . . . . .	1766
12.228 osal/src/os/inc/osapi-filesystem.h File Reference . . . . .	1767
12.228.1 Detailed Description . . . . .	1768
12.228.2 Macro Definition Documentation . . . . .	1768
12.229 osal/src/os/inc/osapi-heap.h File Reference . . . . .	1768
12.229.1 Detailed Description . . . . .	1768
12.230 osal/src/os/inc/osapi-idmap.h File Reference . . . . .	1768
12.230.1 Detailed Description . . . . .	1770
12.230.2 Macro Definition Documentation . . . . .	1770
12.231 osal/src/os/inc/osapi-macros.h File Reference . . . . .	1770
12.231.1 Detailed Description . . . . .	1770
12.231.2 Macro Definition Documentation . . . . .	1770
12.232 osal/src/os/inc/osapi-module.h File Reference . . . . .	1772
12.232.1 Detailed Description . . . . .	1773
12.232.2 Macro Definition Documentation . . . . .	1773
12.233 osal/src/os/inc/osapi-mutex.h File Reference . . . . .	1773
12.233.1 Detailed Description . . . . .	1774
12.234 osal/src/os/inc/osapi-network.h File Reference . . . . .	1774
12.234.1 Detailed Description . . . . .	1774
12.235 osal/src/os/inc/osapi printf.h File Reference . . . . .	1774
12.235.1 Detailed Description . . . . .	1774
12.236 osal/src/os/inc/osapi-queue.h File Reference . . . . .	1775
12.236.1 Detailed Description . . . . .	1775

---

12.237 osal/src/os/inc/osapi-rwlock.h File Reference . . . . .	1775
12.237.1 Detailed Description . . . . .	1776
12.238 osal/src/os/inc/osapi-select.h File Reference . . . . .	1776
12.238.1 Detailed Description . . . . .	1777
12.238.2 Enumeration Type Documentation . . . . .	1777
12.239 osal/src/os/inc/osapi-shell.h File Reference . . . . .	1777
12.239.1 Detailed Description . . . . .	1777
12.240 osal/src/os/inc/osapi-sockets.h File Reference . . . . .	1777
12.240.1 Detailed Description . . . . .	1779
12.240.2 Macro Definition Documentation . . . . .	1779
12.240.3 Typedef Documentation . . . . .	1779
12.240.4 Enumeration Type Documentation . . . . .	1780
12.241 osal/src/os/inc/osapi-task.h File Reference . . . . .	1781
12.241.1 Detailed Description . . . . .	1782
12.241.2 Macro Definition Documentation . . . . .	1782
12.241.3 Typedef Documentation . . . . .	1782
12.241.4 Function Documentation . . . . .	1783
12.242 osal/src/os/inc/osapi-timebase.h File Reference . . . . .	1783
12.242.1 Detailed Description . . . . .	1784
12.242.2 Typedef Documentation . . . . .	1784
12.243 osal/src/os/inc/osapi-timer.h File Reference . . . . .	1784
12.243.1 Detailed Description . . . . .	1784
12.243.2 Typedef Documentation . . . . .	1785
12.244 osal/src/os/inc/osapi-version.h File Reference . . . . .	1785
12.244.1 Detailed Description . . . . .	1786
12.244.2 Macro Definition Documentation . . . . .	1786
12.244.3 Function Documentation . . . . .	1787
12.245 osal/src/os/inc/osapi.h File Reference . . . . .	1788
12.245.1 Detailed Description . . . . .	1789
12.246 psp/fsw/inc/cfe_psp.h File Reference . . . . .	1789
12.246.1 Macro Definition Documentation . . . . .	1790
12.246.2 Function Documentation . . . . .	1790
12.247 psp/fsw/inc/cfe_psp_cache_api.h File Reference . . . . .	1790
12.247.1 Function Documentation . . . . .	1790
12.248 psp/fsw/inc/cfe_psp_cds_api.h File Reference . . . . .	1792
12.248.1 Function Documentation . . . . .	1792
12.249 psp/fsw/inc/cfe_psp_eepromaccess_api.h File Reference . . . . .	1793
12.249.1 Function Documentation . . . . .	1793
12.250 psp/fsw/inc/cfe_psp_endian.h File Reference . . . . .	1796

12.250.1 Detailed Description . . . . .	1796
12.250.2 Function Documentation . . . . .	1796
12.251 psp/fsw/inc/cfe_psp_error.h File Reference . . . . .	1798
12.251.1 Detailed Description . . . . .	1798
12.251.2 Macro Definition Documentation . . . . .	1798
12.251.3 Typedef Documentation . . . . .	1800
12.251.4 Function Documentation . . . . .	1800
12.252 psp/fsw/inc/cfe_psp_exception_api.h File Reference . . . . .	1801
12.252.1 Function Documentation . . . . .	1801
12.253 psp/fsw/inc/cfe_psp_id_api.h File Reference . . . . .	1803
12.253.1 Function Documentation . . . . .	1803
12.254 psp/fsw/inc/cfe_psp_memaccess_api.h File Reference . . . . .	1803
12.254.1 Function Documentation . . . . .	1804
12.255 psp/fsw/inc/cfe_psp_memrange_api.h File Reference . . . . .	1807
12.255.1 Macro Definition Documentation . . . . .	1808
12.255.2 Function Documentation . . . . .	1809
12.256 psp/fsw/inc/cfe_psp_port_api.h File Reference . . . . .	1813
12.256.1 Function Documentation . . . . .	1813
12.257 psp/fsw/inc/cfe_psp_ssr_api.h File Reference . . . . .	1815
12.257.1 Function Documentation . . . . .	1816
12.258 psp/fsw/inc/cfe_psp_timertick_api.h File Reference . . . . .	1816
12.258.1 Macro Definition Documentation . . . . .	1817
12.258.2 Function Documentation . . . . .	1817
12.259 psp/fsw/inc/cfe_psp_version_api.h File Reference . . . . .	1818
12.259.1 Function Documentation . . . . .	1819
12.260 psp/fsw/inc/cfe_psp_watchdog_api.h File Reference . . . . .	1819
12.260.1 Macro Definition Documentation . . . . .	1821
12.260.2 Function Documentation . . . . .	1823
<b>Index</b>	<b>1825</b>

## 1 CFDP (CF) Documentation

- [CFS CFDP Introduction](#)
- [CFS CFDP Overview](#)
- [CFS CFDP Design](#)
- [CFS CFDP Operation](#)
- [CFS CFDP Deployment Guide](#)

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

## 1.1 CFS CFDP Introduction

### Scope

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

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

### CFS CFDP Version

### Acronyms

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

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

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

Next: [CFS CFDP Overview](#)

## 1.2 CFS CFDP Overview

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

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

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

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

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

The CFS CFDP application is a CFDP implementation written specifically for CFS. The goal was bounded, small, time and space-efficient implementation of the CFDP features necessary for flight. CF implements Class 1 and 2 send and

receive, as well as logic to cancel, suspend, resume, abandon, freeze, etc. Messages utilize "zero-copy" software bus buffers. CF configuration is mainly through the configuration table.

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

Prev: [CFS CFDP Introduction](#)

Next: [CFS CFDP Design](#)

### 1.3 CFS CFDP Design

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

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

#### Key differences from CF v2.X

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

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

Prev: [CFS CFDP Overview](#)

Next: [CFS CFDP Operation](#)

### 1.4 CFS CFDP Operation

#### Initialization

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

## Outgoing Messages

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

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

## Incoming Messages

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

## Temporary file and directory use

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

## Engine

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

## Starting a Transaction

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

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

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

## Transaction Class

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

## Queue Entries

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

## Queues

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

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

## Incoming File Transactions

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

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

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

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

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

## Outgoing File Transactions

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

## Output Channels

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

## Queueing Files for Output

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

## Priority

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

## Preserve Setting

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

## Throttling Semaphore

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

## Polling Directories

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

### Failed Transmit and directory use

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

### Efficiency

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

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

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

### Memory Use

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

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

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

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

For outgoing file transactions CF requests a buffer from SB sized for CF\_MAX\_PDU\_SIZE, the CCSDS header size, and CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES. Outgoing packets are then filled up to this maximum value for transmission. Note the maximum outgoing size can also be limited by the table or set parameter command for outgoing\_file\_chunk\_size.

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

Prev: [CFS CFDP Design](#)

Next: [CFS CFDP Deployment Guide](#)

## 1.5 CFS CFDP Deployment Guide

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

### Configuration

CF uses two sets of configuration parameters: compile-time configurable parameters in the cf\_platform\_cfg.h file and run-time configurable parameters in the [cf\\_def\\_config.c](#) file. Most parameters are included in the [cf\\_def\\_config.c](#) file for maximum flexibility.

cf\_platform\_cfg.h uses macro definitions for configurable options, while [cf\\_def\\_config.c](#) uses a table. [cf\\_def\\_config.c](#) parameters can be accessed using the get/set functions [CF\\_GetParamCmd\(\)](#) and [CF\\_SetParamCmd\(\)](#).

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

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

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

### Integration

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

**Scheduler** CF as a whole expects the CF\_WAKEUP\_MID message as described earlier at a fixed rate. If the number of wakeups per second is changed in SCH, then the ticks\_per\_second configuration parameter in the CF configuration table must also be updated.

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

**Integration with TO** TO's pipe needs to be able to receive packets of [CF\\_MAX\\_PDU\\_SIZE](#)

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

## 1.6 CFS CFDP Configuration

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

[CFS CFDP Mission Configuration](#)

[CFS CFDP Platform Configuration](#)

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

[CFS CFDP Table Definitions](#)

Prev: [CFS CFDP Deployment Guide](#)

Next: [CFS CFDP Commands](#)

## 1.7 CFS CFDP Table Definitions

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

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

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

Prev: [CFS CFDP Configuration](#)

Next: [CFS CFDP Constraints](#)

## 1.8 CFS CFDP Commands

[CFS CFDP Command Descriptions](#)

[CFS CFDP Command Message IDs](#)

[CFS CFDP Command Structures](#)

[CFS CFDP Command Codes](#)

Prev: [CFS CFDP Table Definitions](#)

Next: [CFS CFDP Telemetry](#)

### 1.8.1 CFS CFDP Command Descriptions

#### Noop Command

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

#### Reset Counters Command

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

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

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

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

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

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

## Transmit File Command

The CF Transmit File command is sent to CF using message ID [CF\\_CMD\\_MID](#) with command code [CF\\_TX\\_FILE\\_CC](#).

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

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

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

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

The first parameter, `cfdp_class`, identifies whether the file will be transferred using [CF\\_CFDP\\_CLASS\\_1](#) or [CF\\_CFDP\\_CLASS\\_2](#). For Class 1 transfers, CF will send the data once and expect no feedback from the peer. Class 2 transfers are more reliable and attempt to fill in data that may have been dropped on the first attempt.

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

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

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

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

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

## Playback Directory Command

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

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

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

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

The first parameter, `cfdp_class`, identifies whether the files will be transferred using [CF\\_CFDP\\_CLASS\\_1](#) or [CF\\_CFDP\\_CLASS\\_2](#). For Class 1 transfers, CF will send the data once and expect no feedback from the peer. Class 2 transfers are more reliable and attempt to fill in data that may have been dropped on the first attempt.

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

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

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

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

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

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

### Freeze Command

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

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

### Thaw Command

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

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

### Suspend Command

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

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

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

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

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

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

## Resume Command

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

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

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

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

## Cancel Command

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

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

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

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

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

### Abandon Command

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

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

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

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

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

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

### Set MIB Parameter Command

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

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

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

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

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

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

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

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

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

### Get MIB Parameter Command

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

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

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

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

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

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

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

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

### Write Queue Information Command

The CF Write Queue Information command is sent to CF using message ID [CF\\_CMD\\_MID](#) with command code [CF\\_WRITE\\_QUEUE\\_CC](#).

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

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

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

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

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

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

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

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

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

### Enable Dequeue Command

The CF Enable Dequeue command is sent to CF using message ID [CF\\_CMD\\_MID](#) with command code [CF\\_ENABLE\\_DEQUEUE\\_CC](#).

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

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

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

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

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

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

### Disable Dequeue Command

The CF Enable Dequeue command is sent to CF using message ID [CF\\_CMD\\_MID](#) with command code [CF\\_DISABLE\\_DEQUEUE\\_CC](#).

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

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

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

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

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

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

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

### Enable Directory Polling Command

The CF Enable Directory Polling command is sent to CF using message ID [CF\\_CMD\\_MID](#) with command code [CF\\_ENABLE\\_DIR\\_POLLING\\_CC](#).

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

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

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

```

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

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

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

### Disable Directory Polling Command

The CF Enable Directory Polling command is sent to CF using message ID [CF\\_CMD\\_MID](#) with command code [CF\\_DISABLE\\_DIR\\_POLLING\\_CC](#).

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

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

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

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

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

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

### Purge Queue Command

The CF Write Queue Information command is sent to CF using message ID [CF\\_CMD\\_MID](#) with command code [CF\\_PURGE\\_QUEUE\\_CC](#).

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

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

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

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

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

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

### Enable Engine Command

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

### Enable Disable Command

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

Prev: [CFS CFDP Configuration](#)

Next: [CFS CFDP Telemetry](#)

## 1.9 CFS CFDP Telemetry

[CFS CFDP Telemetry Descriptions](#)

[CFS CFDP Telemetry Message IDs](#)

[CFS CFDP Telemetry](#)

Prev: [CFS CFDP Commands](#)

Next: [CFS CFDP Events](#)

### 1.9.1 CFS CFDP Telemetry Descriptions

#### CF Housekeeping Telemetry Packet

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

#### CF End of Transaction Packet

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

Prev: [CFS CFDP Telemetry](#)

Next: [CFS CFDP Table Definitions](#)

## 1.10 CFS CFDP Events

[CFS CFDP Event IDs](#)

Prev: [CFS CFDP Telemetry](#)

Next: [CFS CFDP Constraints](#)

## 1.11 CFS CFDP Constraints

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

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

1. an Entity ID length other than the size defined in cf\_platform\_cfg.h
2. a transaction ID length other than the size defined in cf\_platform\_cfg.h
3. a PDU header size greater than CF\_CFDP\_MAX\_HEADER\_SIZE bytes

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

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

All files in polling directories must be closed.

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

Spaces are not allowed in filenames.

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

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

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

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

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

- Unable to create a Software Bus Pipe

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

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

Prev: [CFS CFDP Events](#)

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

## 1.12 CFS CFDP Frequently Asked Questions

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

Prev: [CFS CFDP Constraints](#)

Next: [CFS CFDP Lessons Learned](#)

## 1.13 CFS CFDP Lessons Learned

These are the lessons learned setting up CF application:

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

## 2 Core Flight Executive Documentation

- General Information and Concepts
  - Background
  - Applicable Documents
  - Version Numbers
  - Dependencies
  - Acronyms
  - Glossary of Terms
- Executive Services (ES)
  - cFE Executive Services Overview
  - cFE Executive Services Commands
  - cFE Executive Services Telemetry
  - ES Event Message Reference
  - cFE Executive Services Configuration Parameters
- Events Services (EVS)
  - cFE Event Services Overview
  - cFE Event Services Commands
  - cFE Event Services Telemetry
  - EVS Event Message Reference
  - cFE Event Services Configuration Parameters
- Software Bus Services (SB)
  - cFE Software Bus Overview
  - cFE Software Bus Commands
  - cFE Software Bus Telemetry
  - SB Event Message Reference
  - cFE Software Bus Configuration Parameters
- Table Services (TBL)
  - cFE Table Services Overview
  - cFE Table Services Commands
  - cFE Table Services Telemetry
  - TBL Event Message Reference
  - cFE Table Services Configuration Parameters
- Time Services (TIME)
  - cFE Time Services Overview
  - cFE Time Services Commands
  - cFE Time Services Telemetry
  - TIME Event Message Reference
  - cFE Time Services Configuration Parameters

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

## 2.1 Background

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

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

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

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

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

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

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

### 2.1.1 Core Flight Executive (cFE) Goals

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

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

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

## 2.2 Applicable Documents

Document Title	Link
cFE System (L4) Requirements Document	<a href="#">cfe/docs/'cfe requirements.docx'</a>
cFE Functional (L5) Requirements Document	<a href="#">cfe/docs/cFE_FunctionalRequirements.csv</a>
cFE Application Developers Guide	<a href="#">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 [cfe\\_version.h](#) header file as part of the API definition; these macros must expand to simple integer values, so that they can be used in simple if directives by the macro preprocessor.

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

### 2.3.3 Identifying Development Builds

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

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

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

See [cfe\\_version.h](#) for the standard layout and definition of version information. The apps and repositories follow the same pattern by replacing the CFE\_ prefix with the appropriate name; for example, osal uses OS\_, psp uses CFE\_↔ PSP\_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\_↔ SRC\_VERSION" (Codename: CFSCONSTELLATION), Last Official Release: MAJOR.MINOR.REVISION"
  - Example: "cFE DEVELOPMENT BUILD v6.8.0-rc1+dev123 (Codename: Bootes), Last Official Release: cfe v6.7.0"

Suggested pattern for official releases:

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

## 2.4 Dependencies

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

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

## 2.5 Acronyms

Acronym	Description
AC	Attitude Control
ACE	Attitude Control Electronics
ACS	Attitude Control System
API	Application Programming Interface
APID	CCSDS Application ID
App	Application
CCSDS	Consultative Committee for Space Data Systems
CDH, C&DH	Command and Data Handling
cFE	core Flight Executive
cFS	core Flight System
CM	Configuration Management
CMD	Command
CPU	Central Processing Unit
EDAC	Error Detection and Correction
EEPROM	Electrically Erasable Programmable Read-Only Memory
ES	Executive Services
EVS	Event Services
FC	Function Code
FDC	Failure Detection and Correction
FSW	Flight Software
HW, H/W	Hardware
ICD	Interface Control Document

Acronym	Description
MET	Mission Elapsed Time
MID	Message ID
OS	Operating System
OSAL	Operating System Abstraction Layer
PID	Pipeline ID
PKT	Packet
PSP	Platform Support Package
RAM	Random-Access Memory
SB	Software Bus
SDO	Solar Dynamics Observatory
ST5	Space Technology Five
STCF	Spacecraft Time Correlation Factor
SW, S/W	Software
TAI	International Atomic Time
TBD	To Be Determined
TBL	Table Services
TID	Task ID
TIME	Time Services
TLM	Telemetry
UTC	Coordinated Universal Time

## 2.6 cFE Executive Services Overview

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

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

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

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

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

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

### 2.6.1 Terminology

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

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

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

#### Application

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

#### cFE Application

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

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

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

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

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

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

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

The fields in a single entry include:

Object Type	CFE_APP for an Application, or CFE_LIB for a library.
Path/Filename	This is a cFE Virtual filename, not a vxWorks device pathname
Entry Point	This is the name of the "main" function for App.
CFE Name	The cFE name for the APP or Library
Priority	This is the Priority of the App, not used for a Library
Stack Size	This is the Stack size for the App, not used for a Library
Load Address	This is the Optional Load Address for the App or Library. It is currently not implemented so it should always be 0x0.
Exception Action	<p>This is the Action the cFE should take if the Application has an exception.</p> <ul style="list-style-type: none"> <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 file-name. If the file is found, ES begins to startup the applications that are listed in the file. If ES does not find the file, it attempts to open the [CFE\\_PLATFORM\\_ES\\_NONVOL\\_STARTUP\\_FILE](#).

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

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

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

## 2.6.2 Software Reset

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

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

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

### 2.6.3 Reset Types and Subtypes

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

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

### 2.6.4 Exception and Reset (ER) Log

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

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

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

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

### 2.6.5 Application and Child Task Management

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

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

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

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

### 2.6.6 Starting an Application

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

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

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

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

### 2.6.7 Stopping an Application

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

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

The format of the Stop Application command, is defined in the structure [CFE\\_ES\\_StopAppCmd\\_t](#). The only parameter in the command is an application name.

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

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

### 2.6.8 Restarting an Application

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

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

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

### 2.6.9 Reloading an Application

The [CFE\\_ES\\_RELOAD\\_APP\\_CC](#) command is used to reload an application using a new file name. This command performs the same actions as [CFE\\_ES\\_RESTART\\_APP\\_CC](#) only using the new file.

### 2.6.10 Listing Current Applications

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

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

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

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

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

### 2.6.11 Listing Current Tasks

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

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

### 2.6.12 Loading Common Libraries

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

### 2.6.13 Basic File System

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

### 2.6.14 Performance Data Collection

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

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

Additional information follows:

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

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

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

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

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

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

## 2.6.15 Critical Data Store

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

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

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

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

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

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

## 2.6.16 Memory Pool

Refer to the cFE Application Developers Guide for additional information.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

- **Memory Pool Handle** - the handle, as provided by the operator in the [Telemeter Memory Pool Statistics Command](#). This repeating of the handle in telemetry ensures the operator knows which Memory Pool Statistics are being viewed

- **Pool Size** - The total size of the memory pool (in bytes)
- **Number Blocks Requested** - The total number of memory blocks requested for allocation
- **Number of Errors** - The total number of errors encountered when a block was released
- **Number of Free Bytes** - The total number of bytes in the Memory Pool that have never been allocated to a Memory Block
- **Block Statistics** - For each specified size of memory block (of which there are [CFE\\_MISSION\\_ES\\_POOL\\_MAX\\_BUCKETS](#)), the following statistics are kept
  - **Block Size** - The size, in bytes, of all blocks of this type
  - **Number of Blocks Allocated** - The number of this sized block which are currently allocated and in use
  - **Number of Blocks Free** - The number of this size block which have been in use previously but are no longer being used

### 2.6.17 System Log

The System Log is an array of bytes that contains back-to-back printf type messages from applications. The cFE internal applications use this log when errors are encountered during initialization before the Event Manager is fully initialized. To view the information the [CFE\\_ES\\_WRITE\\_SYS\\_LOG\\_CC](#) command must be sent. This command will write the log to a binary file. The path and filename may be specified in the command. If the filename command field contains an empty string, the configuration parameter [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_SYSLOG\\_FILE](#) is used to specify the path and filename. Use the ground system to get the file and display the contents. The [CFE\\_ES\\_CLEAR\\_SYS\\_LOG\\_CC](#) is used to clear the System log.

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

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

### 2.6.18 Version Identification

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

### 2.6.19 Frequently Asked Questions about Executive Services

None submitted

## 2.7 cFE Executive Services Commands

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

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

### Global `CFE_ES_CLEAR_ER_LOG_CC`

Clears the contents of the Exception and Reset Log

### Global `CFE_ES_CLEAR_SYS_LOG_CC`

Clear Executive Services System Log

### Global `CFE_ES_DELETE_CDS_CC`

Delete Critical Data Store

### Global `CFE_ES_DUMP_CDS_REGISTRY_CC`

Dump Critical Data Store Registry to a File

### Global `CFE_ES_NOOP_CC`

Executive Services No-Op

### Global `CFE_ES_OVER_WRITE_SYS_LOG_CC`

Set Executive Services System Log Mode to Discard/Overwrite

### Global `CFE_ES_QUERY_ALL_CC`

Writes all Executive Services Information on all loaded modules to a File

### Global `CFE_ES_QUERY_ALL_TASKS_CC`

Writes a list of All Executive Services Tasks to a File

### Global `CFE_ES_QUERY_ONE_CC`

Request Executive Services Information on a specified module

### Global `CFE_ES_RELOAD_APP_CC`

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

### Global `CFE_ES_RESET_COUNTERS_CC`

Executive Services Reset Counters

### Global `CFE_ES_RESET_PR_COUNT_CC`

Resets the Processor Reset Counter to Zero

### Global `CFE_ES_RESTART_APP_CC`

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

### Global `CFE_ES_RESTART_CC`

Executive Services Processor / Power-On Reset

### Global `CFE_ES_SEND_MEM_POOL_STATS_CC`

Telemeter Memory Pool Statistics

### Global `CFE_ES_SET_MAX_PR_COUNT_CC`

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

**Global CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC**

Set Performance Analyzer's Filter Masks

**Global CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC**

Set Performance Analyzer's Trigger Masks

**Global CFE\_ES\_START\_APP\_CC**

Load and Start an Application

**Global CFE\_ES\_START\_PERF\_DATA\_CC**

Start Performance Analyzer

**Global CFE\_ES\_STOP\_APP\_CC**

Stop and Unload Application

**Global CFE\_ES\_STOP\_PERF\_DATA\_CC**

Stop Performance Analyzer and write data file

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

Writes Exception and Reset Log to a File

**Global CFE\_ES\_WRITE\_SYS\_LOG\_CC**

Writes contents of Executive Services System Log to a File

## 2.8 cFE Executive Services Telemetry

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

**Global CFE\_ES\_HousekeepingTlm\_Payload\_t**

Executive Services Housekeeping Packet

**Global CFE\_ES\_HousekeepingTlm\_t**

Executive Services Housekeeping Packet

**Global CFE\_ES\_MemStatsTlm\_t**

Memory Pool Statistics Packet

**Global CFE\_ES\_OneAppTlm\_Payload\_t**

Single Application Information Packet

**Global CFE\_ES\_OneAppTlm\_t**

Single Application Information 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\_TRIMASK\_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\_POOL\_MAX\_BUCKETS**

Number of block sizes in SB memory pool structure

**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

Define TBL Task Priority

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

Define TBL Task Stack Size

Define TBL Task Stack Size

Define TBL Task Stack Size

## 2.10 cFE Event Services Overview

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

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

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

- [Event Message Format](#)
- [Local Event Log](#)
- [Event Message Control](#)

- Event Message Filtering
- EVS Registry
- EVS Counters
- Resetting EVS Counters
- Effects of a Processor Reset on EVS
- EVS squelching of misbehaving apps
- Frequently Asked Questions about Event Services

### 2.10.1 Event Message Format

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

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

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

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

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

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

### 2.10.2 Local Event Log

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

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

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

### 2.10.3 Event Message Control

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

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

1. DEBUG

2. INFORMATION

3. ERROR

4. CRITICAL

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

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

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

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

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

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

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

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

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

#### 2.10.4 Event Message Filtering

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

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

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

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

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

	packet x	packet X+1	packet X+2	packet X+3	packet X+4	...
<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.

	packet x	packet X+1	packet X+2	packet X+3	packet X+4	...
<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.

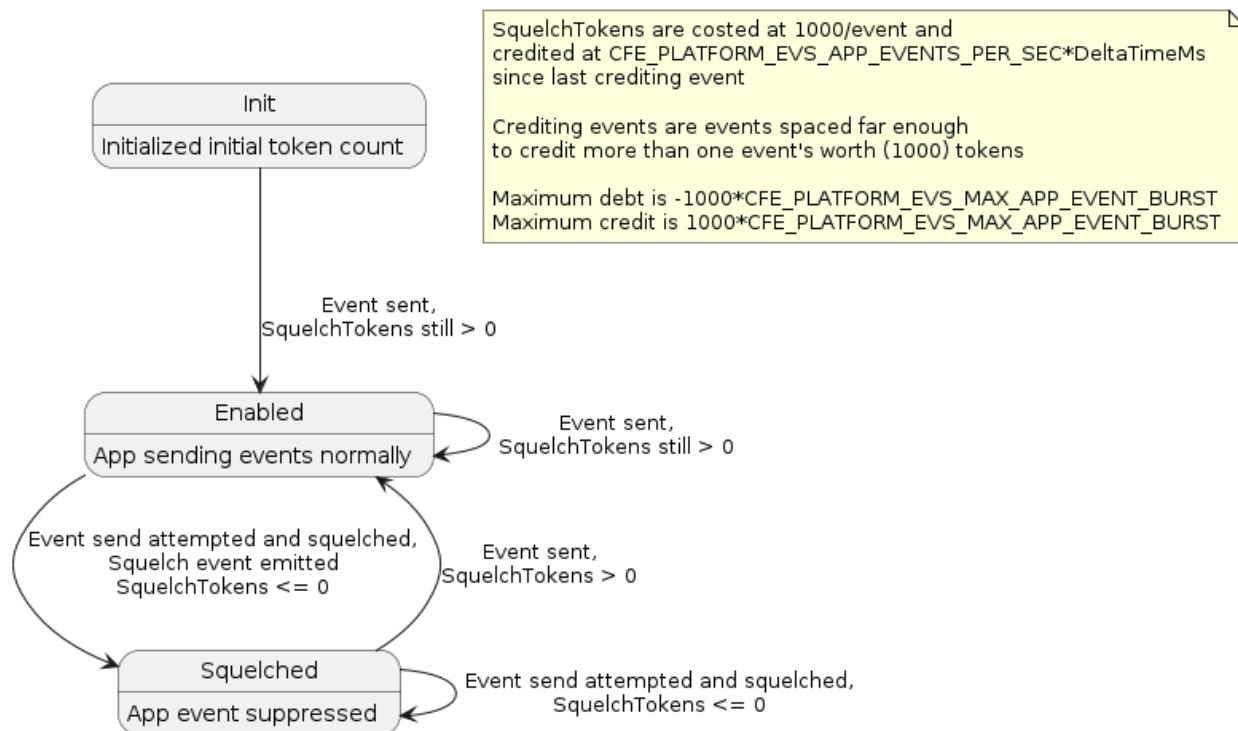


Figure 1 Figure EVS-1: EVS Squelching State Diagram

### 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 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_LongEventTlm_Payload_t`

Event Message Telemetry Packet (Long 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 PipelD) is given back to the caller after the API is executed. Each pipe can be read by only one task, but a task may read more than one pipe. Only the pipe owner is allowed to subscribe to messages on the pipe.

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

**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\_HousekeepingTlm\_Payload\_t**

Software Bus task housekeeping Packet

**Global CFE\_SB\_SingleSubscriptionTlm\_Payload\_t**

SB Subscription Report 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\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT**

Maximum Number of subscription entries per subscription report packet

**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_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

Maximum Length of Full Table Name in messages

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

Maximum Table 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

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

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

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

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

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

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

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

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

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

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

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

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

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 (GPS receiver, spacewire time tick, 1553 bus signal, etc). TIME may also be configured to simulate the tone for lab environments that do not have the necessary hardware to provide a tone signal. Note that MET sub-seconds will be zero at the instant of the tone.

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

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

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

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

## 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 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.

### 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 Normal Operation

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

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

**2.22.20.1 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.

**2.22.20.2 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](#)

**2.22.20.2.1 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](#)

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

#### See also

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

**2.22.20.2.3 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](#)

## 2.22.21 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.

## 2.22.22 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](#)

**2.22.22.0.1 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](#)

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

#### See also

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

**2.22.22.0.3 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](#)

### 2.22.23 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](#)

### 2.22.24 Adjusting Time

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

See also

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

### 2.22.25 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 Frequently Asked Questions about Time Services

None submitted

## 2.23 cFE Time Services Commands

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

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

#### Global [CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#)

Add Delta to Spacecraft Time Correlation Factor

**Global CFE\_TIME\_ADD\_DELAY\_CC**

Add Time to Tone Time Delay

**Global CFE\_TIME\_ADD\_ONE\_HZ\_ADJUSTMENT\_CC**

Add Delta to Spacecraft Time Correlation Factor each 1Hz

**Global CFE\_TIME\_NOOP\_CC**

Time No-Op

**Global CFE\_TIME\_RESET\_COUNTERS\_CC**

Time Reset Counters

**Global CFE\_TIME\_SEND\_DIAGNOSTIC\_CC**

Request TIME Diagnostic Telemetry

**Global CFE\_TIME\_SET\_LEAP\_SECONDS\_CC**

Set Leap Seconds

**Global CFE\_TIME\_SET\_MET\_CC**

Set Mission Elapsed Time

**Global CFE\_TIME\_SET\_SIGNAL\_CC**

Set Tone Signal Source

**Global CFE\_TIME\_SET\_SOURCE\_CC**

Set Time Source

**Global CFE\_TIME\_SET\_STATE\_CC**

Set Time State

**Global CFE\_TIME\_SET\_STCF\_CC**

Set Spacecraft Time Correlation Factor

**Global CFE\_TIME\_SET\_TIME\_CC**

Set Spacecraft Time

**Global CFE\_TIME\_SUB\_ADJUST\_CC**

Subtract Delta from Spacecraft Time Correlation Factor

**Global CFE\_TIME\_SUB\_DELAY\_CC**

Subtract Time from Tone Time Delay

**Global CFE\_TIME\_SUB\_ONE\_HZ\_ADJUSTMENT\_CC**

Subtract Delta from Spacecraft Time Correlation Factor each 1Hz

## 2.24 cFE Time Services Telemetry

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

**Global CFE\_TIME\_DiagnosticTlm\_Payload\_t**

Time Services Diagnostics Packet

**Global CFE\_TIME\_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

Local MET or Virtual MET Selection for Time Servers

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

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

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

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

Define the Local Clock Rollover Value in seconds and subseconds

Define the Local Clock Rollover Value in seconds and subseconds

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

Define TIME Task Priorities

Define TIME Task Priorities

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

Define TIME Task Stack Sizes

Define TIME Task Stack Sizes

## 2.26 cFE Event Message Cross Reference

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

---

## 2.27 cFE Command Mnemonic Cross Reference

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

---

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

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

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

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

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

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

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

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

**Global CFE\_ES\_NOOP\_CC**

\$sc\_\$cpu\_ES\_NOOP

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

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

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

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

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

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

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

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

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

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

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

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

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

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

**Global CFE\_ES\_START\_PERF\_DATA\_CC**  
\$sc\_\$cpu\_ES\_StartLAData

**Global CFE\_ES\_STOP\_APP\_CC**  
\$sc\_\$cpu\_ES\_StopApp

**Global CFE\_ES\_STOP\_PERF\_DATA\_CC**  
\$sc\_\$cpu\_ES\_StopLAData

**Global CFE\_ES\_WRITE\_ER\_LOG\_CC**  
\$sc\_\$cpu\_ES\_WriteERLog2File

**Global CFE\_ES\_WRITE\_SYS\_LOG\_CC**  
\$sc\_\$cpu\_ES\_WriteSysLog2File

**Global CFE\_EVS\_ADD\_EVENT\_FILTER\_CC**  
\$sc\_\$cpu\_EVS\_AddEvtFltr

**Global CFE\_EVS\_CLEAR\_LOG\_CC**  
\$sc\_\$cpu\_EVS\_ClrLog

**Global CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC**  
\$sc\_\$cpu\_EVS\_DelEvtFltr

**Global CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC**  
\$sc\_\$cpu\_EVS\_DisAppEvtType, \$sc\_\$cpu\_EVS\_DisAppEvtTypeMask

**Global CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC**  
\$sc\_\$cpu\_EVS\_DisAppEvGen

**Global CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC**  
\$sc\_\$cpu\_EVS\_DisEventType, \$sc\_\$cpu\_EVS\_DisEventTypeMask

**Global CFE\_EVS\_DISABLE\_PORTS\_CC**  
\$sc\_\$cpu\_EVS\_DisPort, \$sc\_\$cpu\_EVS\_DisPortMask

**Global CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC**  
\$sc\_\$cpu\_EVS\_EnaAppEvtType, \$sc\_\$cpu\_EVS\_EnaAppEvtTypeMask

**Global CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC**  
\$sc\_\$cpu\_EVS\_EnaAppEvGen

**Global CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC**  
\$sc\_\$cpu\_EVS\_EnaEventType, \$sc\_\$cpu\_EVS\_EnaEventTypeMask

**Global CFE\_EVS\_ENABLE\_PORTS\_CC**  
\$sc\_\$cpu\_EVS\_EnaPort, \$sc\_\$cpu\_EVS\_EnaPortMask

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

**Global CFE\_EVS\_RESET\_ALL\_FILTERS\_CC**  
\$sc\_\$cpu\_EVS\_RstAllFltrs

**Global CFE\_EVS\_RESET\_APP\_COUNTER\_CC**  
\$sc\_\$cpu\_EVS\_RstAppCtrs

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

**Global CFE\_EVS\_RESET\_FILTER\_CC**  
\$sc\_\$cpu\_EVS\_RstBinFltrCtr

**Global CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC**  
\$sc\_\$cpu\_EVS\_SetEvtFmt

**Global CFE\_EVS\_SET\_FILTER\_CC**  
\$sc\_\$cpu\_EVS\_SetBinFltrMask

**Global CFE\_EVS\_SET\_LOG\_MODE\_CC**  
\$sc\_\$cpu\_EVS\_SetLogMode

**Global CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC**  
\$sc\_\$cpu\_EVS\_WriteAppData2File

**Global CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC**  
\$sc\_\$cpu\_EVS\_WriteLog2File

**Global CFE\_SB\_DISABLE\_ROUTE\_CC**  
\$sc\_\$cpu\_SB\_DisRoute

**Global CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC**  
\$sc\_\$cpu\_SB\_DisSubRptg

**Global CFE\_SB\_ENABLE\_ROUTE\_CC**  
\$sc\_\$cpu\_SB\_EnaRoute

**Global CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC**  
\$sc\_\$cpu\_SB\_EnaSubRptg

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

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

**Global CFE\_SB\_SEND\_PREV\_SUBS\_CC**  
\$sc\_\$cpu\_SB\_SendPrevSubs

**Global CFE\_SB\_SEND\_SB\_STATS\_CC**  
\$sc\_\$cpu\_SB\_DumpStats

**Global CFE\_SB\_WRITE\_MAP\_INFO\_CC**  
\$sc\_\$cpu\_SB\_WriteMap2File

**Global CFE\_SB\_WRITE\_PIPE\_INFO\_CC**  
\$sc\_\$cpu\_SB\_WritePipe2File

**Global CFE\_SB\_WRITE\_ROUTING\_INFO\_CC**  
\$sc\_\$cpu\_SB\_WriteRouting2File

**Global CFE\_TBL\_ABORT\_LOAD\_CC**  
\$sc\_\$cpu\_TBL\_LOADABORT

**Global CFE\_TBL\_ACTIVATE\_CC**  
\$sc\_\$cpu\_TBL\_ACTIVATE

**Global CFE\_TBL\_DELETE\_CDS\_CC**  
\$sc\_\$cpu\_TBL\_DeleteCDS

**Global CFE\_TBL\_DUMP\_CC**  
\$sc\_\$cpu\_TBL\_DUMP

**Global CFE\_TBL\_DUMP\_REGISTRY\_CC**  
\$sc\_\$cpu\_TBL\_WriteReg2File

**Global CFE\_TBL\_LOAD\_CC**  
\$sc\_\$cpu\_TBL\_Load

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

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

**Global CFE\_TBL\_SEND\_REGISTRY\_CC**  
\$sc\_\$cpu\_TBL\_TLMReg

**Global CFE\_TBL\_VALIDATE\_CC**  
\$sc\_\$cpu\_TBL\_VALIDATE

**Global CFE\_TIME\_ADD\_ADJUST\_CC**  
\$sc\_\$cpu\_TIME\_AddSTCFAdj

**Global CFE\_TIME\_ADD\_DELAY\_CC**  
\$sc\_\$cpu\_TIME\_AddClockLat

**Global CFE\_TIME\_ADD\_ONE\_HZ\_ADJUSTMENT\_CC**  
\$sc\_\$cpu\_TIME\_Add1HzSTCF

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

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

**Global CFE\_TIME\_SEND\_DIAGNOSTIC\_CC**  
\$sc\_\$cpu\_TIME\_RequestDiag

**Global CFE\_TIME\_SET\_LEAP\_SECONDS\_CC**  
\$sc\_\$cpu\_TIME\_SetClockLeap

**Global CFE\_TIME\_SET\_MET\_CC**  
\$sc\_\$cpu\_TIME\_SetClockMET

**Global CFE\_TIME\_SET\_SIGNAL\_CC**  
\$sc\_\$cpu\_TIME\_SetSignal

**Global CFE\_TIME\_SET\_SOURCE\_CC**  
\$sc\_\$cpu\_TIME\_SetSource

**Global CFE\_TIME\_SET\_STATE\_CC**  
\$sc\_\$cpu\_TIME\_SetState

**Global CFE\_TIME\_SET\_STCF\_CC**  
\$sc\_\$cpu\_TIME\_SetClockSTCF

**Global CFE\_TIME\_SET\_TIME\_CC**  
\$sc\_\$cpu\_TIME\_SetClock

**Global CFE\_TIME\_SUB\_Adjust\_CC**  
\$sc\_\$cpu\_TIME\_SubSTCFAj

**Global CFE\_TIME\_SUB\_DELAY\_CC**  
\$sc\_\$cpu\_TIME\_SubClockLat

**Global CFE\_TIME\_SUB\_ONE\_HZ\_ADJUSTMENT\_CC**  
\$sc\_\$cpu\_TIME\_Sub1HzSTCF

## 2.28 cFE Telemetry Mnemonic Cross Reference

The following cross reference maps the cFE telemetry packet members to their associated ground system telemetry mnemonics.

**Global CFE\_ES\_AppInfo::AddressesAreValid**  
\$sc\_\$cpu\_ES\_AddrsValid

**Global CFE\_ES\_AppInfo::BSSAddress**  
\$sc\_\$cpu\_ES\_BSSAddress

**Global CFE\_ES\_AppInfo::BSSSize**  
\$sc\_\$cpu\_ES\_BSSSize

**Global CFE\_ES\_AppInfo::CodeAddress**  
\$sc\_\$cpu\_ES\_CodeAddress

**Global CFE\_ES\_AppInfo::CodeSize**  
\$sc\_\$cpu\_ES\_CodeSize

**Global CFE\_ES\_AppInfo::DataAddress**  
\$sc\_\$cpu\_ES\_DataAddress

**Global CFE\_ES\_AppInfo::DataSize**  
\$sc\_\$cpu\_ES\_DataSize

**Global CFE\_ES\_AppInfo::EntryPoint [CFE\_MISSION\_MAX\_API\_LEN]**  
\$sc\_\$cpu\_ES\_AppEntryPt[OS\_MAX\_API\_NAME]

**Global CFE\_ES\_AppInfo::ExceptionAction**  
\$sc\_\$cpu\_ES\_ExceptnActn

**Global CFE\_ES\_AppInfo::ExecutionCounter**  
\$sc\_\$cpu\_ES\_ExecutionCtr

**Global CFE\_ES\_AppInfo::FileName [CFE\_MISSION\_MAX\_PATH\_LEN]**  
\$sc\_\$cpu\_ES\_AppFilename[OS\_MAX\_PATH\_LEN]

**Global CFE\_ES\_AppInfo::MainTaskId**  
\$sc\_\$cpu\_ES\_MainTaskId

**Global CFE\_ES\_AppInfo::MainTaskName [CFE\_MISSION\_MAX\_API\_LEN]**  
\$sc\_\$cpu\_ES\_MainTaskName[OS\_MAX\_API\_NAME]

**Global CFE\_ES\_AppInfo::Name [CFE\_MISSION\_MAX\_API\_LEN]**  
\$sc\_\$cpu\_ES\_AppName[OS\_MAX\_API\_NAME]

**Global CFE\_ES\_AppInfo::NumOfChildTasks**  
\$sc\_\$cpu\_ES\_ChildTasks

**Global CFE\_ES\_AppInfo::Priority**  
\$sc\_\$cpu\_ES\_Priority

**Global CFE\_ES\_AppInfo::ResourceID**  
\$sc\_\$cpu\_ES\_AppID

**Global CFE\_ES\_AppInfo::StackSize**  
\$sc\_\$cpu\_ES\_StackSize

**Global CFE\_ES\_AppInfo::StartAddress**  
\$sc\_\$cpu\_ES\_StartAddr

**Global CFE\_ES\_AppInfo::Type**  
\$sc\_\$cpu\_ES\_AppType

**Global CFE\_ES\_HousekeepingTlm\_Payload::BootSource**  
\$sc\_\$cpu\_ES\_BootSource

**Global CFE\_ES\_HousekeepingTlm\_Payload::CFECoreChecksum**  
\$sc\_\$cpu\_ES\_CKSUM

**Global CFE\_ES\_HousekeepingTlm\_Payload::CFEMajorVersion**  
\$sc\_\$cpu\_ES\_CFEMAJORVER

**Global CFE\_ES\_HousekeepingTlm\_Payload::CFEMinorVersion**  
\$sc\_\$cpu\_ES\_CFEminorVER

**Global CFE\_ES\_HousekeepingTlm\_Payload::CFEMissionRevision**  
\$sc\_\$cpu\_ES\_CFEMISSIONREV

**Global CFE\_ES\_HousekeepingTlm\_Payload::CFERevision**  
\$sc\_\$cpu\_ES\_CFEREVISION

**Global CFE\_ES\_HousekeepingTlm\_Payload::CommandCounter**  
\$sc\_\$cpu\_ES\_CMDPC

**Global CFE\_ES\_HousekeepingTlm\_Payload::CommandErrorCounter**  
\$sc\_\$cpu\_ES\_CMDEC

**Global CFE\_ES\_HousekeepingTlm\_Payload::ERLogEntries**  
\$sc\_\$cpu\_ES\_ERLOGENTRIES

**Global CFE\_ES\_HousekeepingTlm\_Payload::ERLogIndex**  
\$sc\_\$cpu\_ES\_ERLOGINDEX

**Global CFE\_ES\_HousekeepingTlm\_Payload::HeapBlocksFree**  
\$sc\_\$cpu\_ES\_HeapBlocksFree

**Global CFE\_ES\_HousekeepingTlm\_Payload::HeapBytesFree**  
\$sc\_\$cpu\_ES\_HeapBytesFree

**Global CFE\_ES\_HousekeepingTlm\_Payload::HeapMaxBlockSize**  
\$sc\_\$cpu\_ES\_HeapMaxBlkSize

**Global CFE\_ES\_HousekeepingTlm\_Payload::MaxProcessorResets**  
\$sc\_\$cpu\_ES\_MaxProcResets

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALMajorVersion**  
\$sc\_\$cpu\_ES\_OSMAJORVER

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALMinorVersion**  
\$sc\_\$cpu\_ES\_OSMINORVER

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALMissionRevision**  
\$sc\_\$cpu\_ES\_OSMISSIONREV

**Global CFE\_ES\_HousekeepingTlm\_Payload::OSALRevision**  
\$sc\_\$cpu\_ES\_OSREVISION

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfDataCount**  
\$sc\_\$cpu\_ES\_PerfDataCnt

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfDataEnd**  
\$sc\_\$cpu\_ES\_PerfDataEnd

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfDataStart**  
\$sc\_\$cpu\_ES\_PerfDataStart

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfDataToWrite**  
\$sc\_\$cpu\_ES\_PerfData2Write

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfFilterMask [CFE\_MISSION\_ES\_PERF\_MAX\_IDS/32]**  
\$sc\_\$cpu\_ES\_PerfFltrMask[MaskCnt]

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfMode**  
\$sc\_\$cpu\_ES\_PerfMode

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfState**  
\$sc\_\$cpu\_ES\_PerfState

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfTriggerCount**  
\$sc\_\$cpu\_ES\_PerfTrigCnt

**Global CFE\_ES\_HousekeepingTlm\_Payload::PerfTriggerMask [CFE\_MISSION\_ES\_PERF\_MAX\_IDS/32]**  
\$sc\_\$cpu\_ES\_PerfTrigMask[MaskCnt]

```
Global CFE_ES_HousekeepingTlm_Payload::ProcessorResets
$sc_$cpu_ES_ProcResetCnt

Global CFE_ES_HousekeepingTlm_Payload::PSPMajorVersion
$sc_$cpu_ES_PSPMAJORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMinorVersion
$sc_$cpu_ES_PSPMINORVER

Global CFE_ES_HousekeepingTlm_Payload::PSPMissionRevision
$sc_$cpu_ES_PSPMISSIONREV

Global CFE_ES_HousekeepingTlm_Payload::PSPRevision
$sc_$cpu_ES_PSPREVISION

Global CFE_ES_HousekeepingTlm_Payload::RegisteredCoreApps
$sc_$cpu_ES_RegCoreApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredExternalApps
$sc_$cpu_ES_RegExtApps

Global CFE_ES_HousekeepingTlm_Payload::RegisteredLibs
$sc_$cpu_ES_RegLibs

Global CFE_ES_HousekeepingTlm_Payload::RegisteredTasks
$sc_$cpu_ES_RegTasks

Global CFE_ES_HousekeepingTlm_Payload::ResetSubtype
$sc_$cpu_ES_ResetSubtype

Global CFE_ES_HousekeepingTlm_Payload::ResetType
$sc_$cpu_ES_ResetType

Global CFE_ES_HousekeepingTlm_Payload::SysLogBytesUsed
$sc_$cpu_ES_SYSLOGBYTEUSED

Global CFE_ES_HousekeepingTlm_Payload::SysLogEntries
$sc_$cpu_ES_SYSLOGENTRIES

Global CFE_ES_HousekeepingTlm_Payload::SysLogMode
$sc_$cpu_ES_SYSLOGMODE

Global CFE_ES_HousekeepingTlm_Payload::SysLogSize
$sc_$cpu_ES_SYSLOGSIZE

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

Global CFE_ES_MemPoolStats::CheckErrCtr
$sc_$cpu_ES_BlkErrCTR

Global CFE_ES_MemPoolStats::NumBlocksRequested
$sc_$cpu_ES_BlksREQ

Global CFE_ES_MemPoolStats::NumFreeBytes
$sc_$cpu_ES_FreeBytes

Global CFE_ES_MemPoolStats::PoolSize
$sc_$cpu_ES_PoolSize

Global CFE_ES_PoolStatsTlm_Payload::PoolHandle
$sc_$cpu_ES_PoolHandle
```

**Global CFE\_EVS\_AppTlmData::AppEnableStatus**

\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS].APPENASTAT

**Global CFE\_EVS\_AppTlmData::AppID**

\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS].APPID

**Global CFE\_EVS\_AppTlmData::AppMessageSentCounter**

\$sc\_\$cpu\_EVS\_APP[CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS].APPMSGSENTC

**Global CFE\_EVS\_AppTlmData::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\_LOGOVERLOWC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::MessageFormatMode**

\$sc\_\$cpu\_EVS\_MSGFMTMODE

**Global CFE\_EVS\_HousekeepingTlm\_Payload::MessageSendCounter**

\$sc\_\$cpu\_EVS\_MSGSENTC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::MessageTruncCounter**

\$sc\_\$cpu\_EVS\_MSGTRUNC

**Global CFE\_EVS\_HousekeepingTlm\_Payload::OutputPort**

\$sc\_\$cpu\_EVS\_OUTPUTPORT

**Global CFE\_EVS\_HousekeepingTlm\_Payload::Spare1**

\$sc\_\$cpu\_EVS\_HK\_SPARE1

**Global CFE\_EVS\_HousekeepingTlm\_Payload::Spare2**

\$sc\_\$cpu\_EVS\_HK\_SPARE2

**Global CFE\_EVS\_HousekeepingTlm\_Payload::Spare3**

\$sc\_\$cpu\_EVS\_HK\_SPARE3

**Global CFE\_EVS\_HousekeepingTlm\_Payload::UnregisteredAppCounter**

\$sc\_\$cpu\_EVS\_UNREGAPPC

**Global CFE\_EVS\_LongEventTlm\_Payload::Message [CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH]**

\$sc\_\$cpu\_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\_SMPSBBIU

**Global CFE\_SB\_StatsTlm\_Payload::PeakSubscriptionsInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPSIU

**Global CFE\_SB\_StatsTlm\_Payload::PipeDepthStats [CFE\_MISSION\_SB\_MAX\_PIPES]**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES]

**Global CFE\_SB\_StatsTlm\_Payload::PipesInUse**  
\$sc\_\$cpu\_SB\_Stat.SB\_SMPIU

---

```

Global CFE_SB_StatsTlm_Payload::SBBuffersInUse
$sc_$cpu_SB_Stat.SB_SMSBBIU

Global CFE_SB_StatsTlm_Payload::SubscriptionsInUse
$sc_$cpu_SB_Stat.SB_SMSIU

Global CFE_TBL_HousekeepingTlm_Payload::ActiveBuffer
$sc_$cpu_TBL_LastValBuf

Global CFE_TBL_HousekeepingTlm_Payload::ByteAlignPad1
$sc_$cpu_TBL_BytAlignPad1

Global CFE_TBL_HousekeepingTlm_Payload::CommandCounter
$sc_$cpu_TBL_CMDPC

Global CFE_TBL_HousekeepingTlm_Payload::CommandErrorCounter
$sc_$cpu_TBL_CMDEC

Global CFE_TBL_HousekeepingTlm_Payload::FailedValCounter
$sc_$cpu_TBL_ValFailedCtr

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

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

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

Global CFE_TBL_HousekeepingTlm_Payload::LastUpdatedTable [CFE_MISSION_TBL_MAX_FULL_NAME_← LEN]
$sc_$cpu_TBL_LastUpdTblName[CFE_TB_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload::LastUpdateTime
$sc_$cpu_TBL_LastUpdTime, $sc_$cpu_TBL_SECONDS, $sc_$cpu_TBL_SUBSECONDS

Global CFE_TBL_HousekeepingTlm_Payload::LastValCrc
$sc_$cpu_TBL_LastValCRC

Global CFE_TBL_HousekeepingTlm_Payload::LastValStatus
$sc_$cpu_TBL_LastValS

Global CFE_TBL_HousekeepingTlm_Payload::LastValTableName [CFE_MISSION_TBL_MAX_FULL_NAME_← LEN]
$sc_$cpu_TBL_LastValTblName[CFE_TB_MAX_FULL_NAME_LEN]

Global CFE_TBL_HousekeepingTlm_Payload::MemPoolHandle
$sc_$cpu_TBL_MemPoolHandle

Global CFE_TBL_HousekeepingTlm_Payload::NumFreeSharedBufs
$sc_$cpu_TBL_NumFreeShrBuf

Global CFE_TBL_HousekeepingTlm_Payload::NumLoadPending
$sc_$cpu_TBL_NumUpdatesPend

Global CFE_TBL_HousekeepingTlm_Payload::NumTables
$sc_$cpu_TBL_NumTables

Global CFE_TBL_HousekeepingTlm_Payload::NumValRequests
$sc_$cpu_TBL_ValReqCtr

```

**Global CFE\_TBL\_HousekeepingTlm\_Payload::SuccessValCounter**  
\$sc\_\$cpu\_TBL\_ValSuccessCtr

**Global CFE\_TBL\_HousekeepingTlm\_Payload::ValidationCounter**  
\$sc\_\$cpu\_TBL\_ValCompltdCtr

**Global CFE\_TBL\_TblRegPacket\_Payload::ActiveBufferAddr**  
\$sc\_\$cpu\_TBL\_ActBufAdd

**Global CFE\_TBL\_TblRegPacket\_Payload::ByteAlign4**  
\$sc\_\$cpu\_TBL\_Spare4

**Global CFE\_TBL\_TblRegPacket\_Payload::Crc**  
\$sc\_\$cpu\_TBL\_CRC

**Global CFE\_TBL\_TblRegPacket\_Payload::Critical**  
\$sc\_\$cpu\_TBL\_Spare3

**Global CFE\_TBL\_TblRegPacket\_Payload::DoubleBuffered**  
\$sc\_\$cpu\_TBL\_DblBuffered

**Global CFE\_TBL\_TblRegPacket\_Payload::DumpOnly**  
\$sc\_\$cpu\_TBL\_DumpOnly

**Global CFE\_TBL\_TblRegPacket\_Payload::FileTime**  
\$sc\_\$cpu\_TBL\_FILETIME

**Global CFE\_TBL\_TblRegPacket\_Payload::InactiveBufferAddr**  
\$sc\_\$cpu\_TBL\_IActBufAdd

**Global CFE\_TBL\_TblRegPacket\_Payload::LastFileLoaded [CFE\_MISSION\_MAX\_PATH\_LEN]**  
\$sc\_\$cpu\_TBL\_LastFileUpd[OS\_MAX\_PATH\_LEN]

**Global CFE\_TBL\_TblRegPacket\_Payload::LoadPending**  
\$sc\_\$cpu\_TBL\_UpdatePndng

**Global CFE\_TBL\_TblRegPacket\_Payload::Name [CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN]**  
\$sc\_\$cpu\_TBL\_Name[CFE\_TB\_MAX\_FULL\_NAME\_LEN]

**Global CFE\_TBL\_TblRegPacket\_Payload::OwnerAppName [CFE\_MISSION\_MAX\_API\_LEN]**  
\$sc\_\$cpu\_TBL\_OwnerApp[OS\_MAX\_API\_NAME]

**Global CFE\_TBL\_TblRegPacket\_Payload::Size**  
\$sc\_\$cpu\_TBL\_SIZE

**Global CFE\_TBL\_TblRegPacket\_Payload::TableLoadedOnce**  
\$sc\_\$cpu\_TBL\_LoadedOnce

**Global CFE\_TBL\_TblRegPacket\_Payload::TimeOfLastUpdate**  
\$sc\_\$cpu\_TBL\_TimeLastUpd, \$sc\_\$cpu\_TBL\_TLUSECONDS, \$sc\_\$cpu\_TBL\_TLU SUBSECONDS

**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\_DTVlaidS, \$sc\_\$cpu\_TIME\_DTVlaidSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::AtToneLeapSeconds**  
\$sc\_\$cpu\_TIME\_DLepS

**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\_DFlagAdjd,  
\$sc\_\$cpu\_TIME\_DFlag1Hzd, \$sc\_\$cpu\_TIME\_DFlagClat, \$sc\_\$cpu\_TIME\_DFlagSorC, \$sc\_\$cpu\_TIME\_DFlag←  
NIU

**Global CFE\_TIME\_DiagnosticTlm\_Payload::CurrentLatch**  
\$sc\_\$cpu\_TIME\_DLlocalS, \$sc\_\$cpu\_TIME\_DLlocalSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::CurrentMET**  
\$sc\_\$cpu\_TIME\_DMETS, \$sc\_\$cpu\_TIME\_DMETSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::CurrentTAI**  
\$sc\_\$cpu\_TIME\_DTAIS, \$sc\_\$cpu\_TIME\_DTAISS

**Global CFE\_TIME\_DiagnosticTlm\_Payload::CurrentUTC**  
\$sc\_\$cpu\_TIME\_DUTCS, \$sc\_\$cpu\_TIME\_DUTCSS

**Global CFE\_TIME\_DiagnosticTlm\_Payload::DataStoreStatus**  
\$sc\_\$cpu\_TIME\_DataStStat

**Global CFE\_TIME\_DiagnosticTlm\_Payload::DelayDirection**  
\$sc\_\$cpu\_TIME\_DLlatentDir

**Global CFE\_TIME\_DiagnosticTlm\_Payload::Forced2Fly**  
\$sc\_\$cpu\_TIME\_DCMD2Fly

**Global CFE\_TIME\_DiagnosticTlm\_Payload::LocalIntCounter**  
\$sc\_\$cpu\_TIME\_D1HzSRCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::LocalTaskCounter**  
\$sc\_\$cpu\_TIME\_D1HzTaskCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::MaxElapsed**  
\$sc\_\$cpu\_TIME\_DMaxWindow

**Global CFE\_TIME\_DiagnosticTlm\_Payload::MaxLocalClock**  
\$sc\_\$cpu\_TIME\_DWrapS, \$sc\_\$cpu\_TIME\_DWrapSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::MinElapsed**  
\$sc\_\$cpu\_TIME\_DMinWindow

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneHzAdjust**  
\$sc\_\$cpu\_TIME\_D1HzAdjS, \$sc\_\$cpu\_TIME\_D1HzAdjSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneHzDirection**  
\$sc\_\$cpu\_TIME\_D1HzAdjDir

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneTimeAdjust**  
\$sc\_\$cpu\_TIME\_DAdjustS, \$sc\_\$cpu\_TIME\_DAdjustSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::OneTimeDirection**  
\$sc\_\$cpu\_TIME\_DAdjustDir

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ServerFlyState**  
\$sc\_\$cpu\_TIME\_DSrvFly

**Global CFE\_TIME\_DiagnosticTlm\_Payload::TimeSinceTone**  
\$sc\_\$cpu\_TIME\_DElapsedS, \$sc\_\$cpu\_TIME\_DElapsedSs

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ToneDataCounter**  
\$sc\_\$cpu\_TIME\_DTatTCNT

**Global CFE\_TIME\_DiagnosticTlm\_Payload::ToneDataLatch**  
\$sc\_\$cpu\_TIME\_DTDs, \$sc\_\$cpu\_TIME\_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::AdjustmentFactor**  
\$sc\_\$cpu\_TIME\_ADJ

**Global CFE\_TIME\_HousekeepingTlm\_Payload::ClockStateAPI**

\$sc\_\$cpu\_TIME\_DAPISate

**Global CFE\_TIME\_HousekeepingTlm\_Payload::ClockStateFlags**

\$sc\_\$cpu\_TIME\_StateFlg, \$sc\_\$cpu\_TIME\_FlagSet, \$sc\_\$cpu\_TIME\_FlagFly, \$sc\_\$cpu\_TIME\_FlagSrc, \$sc\_\$cpu\_TIME\_FlagPri, \$sc\_\$cpu\_TIME\_FlagSfly, \$sc\_\$cpu\_TIME\_FlagCfly, \$sc\_\$cpu\_TIME\_FlagAdjd, \$sc\_\$cpu\_TIME\_Flag1Hzd, \$sc\_\$cpu\_TIME\_FlagClat, \$sc\_\$cpu\_TIME\_FlagSorC, \$sc\_\$cpu\_TIME\_FlagNIU

**Global CFE\_TIME\_HousekeepingTlm\_Payload::CommandCounter**

\$sc\_\$cpu\_TIME\_CMDPC

**Global CFE\_TIME\_HousekeepingTlm\_Payload::CommandErrorCounter**

\$sc\_\$cpu\_TIME\_CMDEC

**Global CFE\_TIME\_HousekeepingTlm\_Payload::LeapSeconds**

\$sc\_\$cpu\_TIME\_LeapSecs

**Global CFE\_TIME\_HousekeepingTlm\_Payload::MET**

\$sc\_\$cpu\_TIME\_MET

**Global CFE\_TIME\_HousekeepingTlm\_Payload::STCF**

\$sc\_\$cpu\_TIME\_STCF

### 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.
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> .

Term	Definition
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 that maps to an available 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\\_RegisterSynchCallback](#) - Registers a callback function that is called whenever time synchronization occurs.
  - [CFE\\_TIME\\_UnregisterSynchCallback](#) - Unregisters a callback function that is called whenever time synchronization occurs.
- cFE Miscellaneous Time APIs
  - [CFE\\_TIME\\_Print](#) - Print a time value as a string.
  - [CFE\\_TIME\\_Local1HzISR](#) - This function is called via a timer callback set up at initialization of the TIME service.

## 5 Osal API Documentation

- General Information and Concepts
  - [OSAL Introduction](#)
- Core
  - [OSAL Return Code Defines](#)
  - [OSAL Object Type Defines](#)
  - APIs
    - \* [OSAL Core Operation APIs](#)
    - \* [OSAL Object ID Utility APIs](#)
    - \* [OSAL Task APIs](#)
    - \* [OSAL Message Queue APIs](#)
    - \* [OSAL Heap APIs](#)

- \* OSAL Error Info APIs
- \* OSAL Select APIs
- \* OSAL Printf APIs
- \* OSAL BSP low level access APIs
- \* OSAL Real Time Clock APIs
- \* OSAL Shell APIs
- Common Reference
- Return Code Reference
- Id Map Reference
- Clock Reference
- Task Reference
- Message Queue Reference
- Heap Reference
- Select Reference
- Printf Reference
- BSP Reference
- Shell Reference
- File System
  - File System Overview
  - File Descriptors In Osal
  - OSAL File Access Option Defines
  - OSAL Reference Point For Seek Offset Defines
  - APIs
    - \* OSAL Standard File APIs
    - \* OSAL Directory APIs
    - \* OSAL File System Level APIs
  - File System Reference
  - File Reference
  - Directory Reference
- Object File Loader
  - APIs
    - \* OSAL Dynamic Loader and Symbol APIs
  - File Loader Reference
- Network
  - APIs
    - \* OSAL Network ID APIs
    - \* OSAL Socket Address APIs
    - \* OSAL Socket Management APIs
  - Network Reference
  - Socket Reference
- Timer

- Timer Overview
- APIs
  - \* OSAL Time Base APIs
  - \* OSAL Timer APIs
- Timer Reference
- Time Base Reference
- Semaphore and Mutex
  - OSAL Semaphore State Defines
  - APIs
    - \* OSAL Binary Semaphore APIs
    - \* OSAL Counting Semaphore APIs
    - \* OSAL Mutex APIs
  - Binary Semaphore Reference
  - Counting Semaphore Reference
  - Mutex Reference

## 5.1 OSAL Introduction

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

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

### Note

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

## 5.2 File System Overview

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

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

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

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

- Device Name: This is the name of the virtual device that the Application uses. Common names are “ramdisk1”, “flash1”, or “volatile1” etc. But the name can be any unique string.

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

### 5.3 File Descriptors In Osal

The OSAL uses abstracted file descriptors. This means that the file descriptors passed back from the OS\_open and OS\_creat calls will only work with other OSAL OS\_\* calls. The reasoning for this is as follows:

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

All of the possible file system calls are not implemented. "Special" files requiring OS specific control/operations are by nature not portable. Abstraction in this case is not possible, so the raw OS calls should be used (including open/close/etc). Mixing with OSAL calls is not supported for such cases. [OS\\_TranslatePath](#) is available to support using open directly by an app and maintain abstraction on the file system.

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

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

### 5.4 Timer Overview

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

## 6 cFE Mission Configuration Parameters

### Global [CFE\\_MISSION\\_ES\\_CMD\\_TOPICID](#)

CFE Portable Message Numbers for Commands

**Global CFE\_MISSION\_ES\_HK\_TLM\_TOPICID**

cFE Portable Message Numbers for Telemetry

**Global CFE\_MISSION\_EVS\_CMD\_TOPICID**

cFE Portable Message Numbers for Commands

**Global CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID**

cFE Portable Message Numbers for Telemetry

**Global CFE\_MISSION\_MAX\_API\_LEN**

cFE Maximum length for API names within data exchange structures

cFE Maximum length for API names within data exchange structures

**Global CFE\_MISSION\_MAX\_FILE\_LEN**

cFE Maximum length for filenames within data exchange structures

cFE Maximum length for filenames within data exchange structures

**Global CFE\_MISSION\_MAX\_NUM\_FILES**

cFE Maximum number of files in a message/data exchange

cFE Maximum number of files in a message/data exchange

**Global CFE\_MISSION\_MAX\_PATH\_LEN**

cFE Maximum length for pathnames within data exchange structures

cFE Maximum length for pathnames within data exchange structures

**Global CFE\_MISSION\_SB\_CMD\_TOPICID**

cFE Portable Message Numbers for Commands

**Global CFE\_MISSION\_SB\_HK\_TLM\_TOPICID**

cFE Portable Message Numbers for Telemetry

**Global CFE\_MISSION\_TBL\_CMD\_TOPICID**

cFE Portable Message Numbers for Commands

**Global CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID**

cFE Portable Message Numbers for Telemetry

**Global CFE\_MISSION\_TIME\_CMD\_TOPICID**

cFE Portable Message Numbers for Commands

**Global CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID**

cFE Portable Message Numbers for Global Messages

**Global CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID**

cFE Portable Message Numbers for Telemetry

## 7 Module Index

### 7.1 Modules

Here is a list of all modules:

<b>CFS CFDP Event IDs</b>	<b>152</b>
<b>CFS CFDP Command Codes</b>	<b>193</b>
<b>CFS CFDP Platform Configuration</b>	<b>209</b>

<b>CFS CFDP Mission Configuration</b>	<b>217</b>
<b>CFS CFDP Version</b>	<b>219</b>
<b>CFS CFDP Telemetry</b>	<b>220</b>
<b>CFS CFDP Command Structures</b>	<b>221</b>
<b>CFS CFDP Command Message IDs</b>	<b>228</b>
<b>CFS CFDP Telemetry Message IDs</b>	<b>228</b>
<b>CFS CFDP Data Interface Message IDs</b>	<b>229</b>
<b>cFE Return Code Defines</b>	<b>229</b>
<b>cFE Resource ID APIs</b>	<b>252</b>
<b>cFE Entry/Exit APIs</b>	<b>254</b>
<b>cFE Application Control APIs</b>	<b>256</b>
<b>cFE Application Behavior APIs</b>	<b>258</b>
<b>cFE Information APIs</b>	<b>262</b>
<b>cFE Child Task APIs</b>	<b>270</b>
<b>cFE Miscellaneous APIs</b>	<b>274</b>
<b>cFE Critical Data Store APIs</b>	<b>276</b>
<b>cFE Memory Manager APIs</b>	<b>281</b>
<b>cFE Performance Monitor APIs</b>	<b>289</b>
<b>cFE Generic Counter APIs</b>	<b>291</b>
<b>cFE Registration APIs</b>	<b>296</b>
<b>cFE Send Event APIs</b>	<b>297</b>
<b>cFE Reset Event Filter APIs</b>	<b>301</b>
<b>cFE File Header Management APIs</b>	<b>303</b>
<b>cFE File Utility APIs</b>	<b>306</b>
<b>cFE Generic Message APIs</b>	<b>311</b>
<b>cFE Message Primary Header APIs</b>	<b>312</b>
<b>cFE Message Extended Header APIs</b>	<b>320</b>
<b>cFE Message Secondary Header APIs</b>	<b>326</b>
<b>cFE Message Id APIs</b>	<b>330</b>
<b>cFE Message Integrity APIs</b>	<b>332</b>

cFE Pipe Management APIs	334
cFE Message Subscription Control APIs	339
cFE Send/Receive Message APIs	343
cFE Zero Copy APIs	345
cFE Message Characteristics APIs	348
cFE Message ID APIs	352
cFE SB Pipe options	357
cFE Registration APIs	357
cFE Manage Table Content APIs	362
cFE Access Table Content APIs	368
cFE Get Table Information APIs	372
cFE Table Type Defines	375
cFE Get Current Time APIs	377
cFE Get Time Information APIs	380
cFE Time Arithmetic APIs	382
cFE Time Conversion APIs	384
cFE External Time Source APIs	386
cFE Miscellaneous Time APIs	390
cFE Resource ID base values	392
cFE Clock State Flag Defines	393
OSAL Semaphore State Defines	395
OSAL Binary Semaphore APIs	396
OSAL BSP low level access APIs	400
OSAL Real Time Clock APIs	401
OSAL Core Operation APIs	415
OSAL Condition Variable APIs	418
OSAL Counting Semaphore APIs	424
OSAL Directory APIs	428
OSAL Return Code Defines	432
OSAL Error Info APIs	439

OSAL File Access Option Defines	440
OSAL Reference Point For Seek Offset Defines	440
OSAL Standard File APIs	441
OSAL File System Level APIs	454
OSAL Heap APIs	462
OSAL Object Type Defines	462
OSAL Object ID Utility APIs	465
OSAL Dynamic Loader and Symbol APIs	470
OSAL Mutex APIs	474
OSAL Network ID APIs	477
OSAL Printf APIs	478
OSAL Message Queue APIs	479
OSAL RwLock APIs	483
OSAL Select APIs	487
OSAL Shell APIs	493
OSAL Socket Address APIs	494
OSAL Socket Management APIs	497
OSAL Task APIs	508
OSAL Time Base APIs	513
OSAL Timer APIs	517

## 8 Data Structure Index

### 8.1 Data Structures

Here are the data structures with brief descriptions:

<b>CCSDS_ExtendedHeader</b> CCSDS packet extended header	523
<b>CCSDS_PrimaryHeader</b> CCSDS packet primary header	523
<b>CF_AbandonCmd</b> Abandon command structure	524
<b>CF_AppData_t</b> The CF application global state structure	525

<b>CF_CancelCmd</b>		
Cancel command structure		526
<b>CF_CFDP_FileDirectiveDispatchTable_t</b>		
A table of receive handler functions based on file directive code		527
<b>CF_CFDP_Inv</b>		
Structure representing CFDP LV Object format		527
<b>CF_CFDP_PduAck</b>		
Structure representing CFDP Acknowledge PDU		528
<b>CF_CFDP_PduEof</b>		
Structure representing CFDP End of file PDU		528
<b>CF_CFDP_PduFileDataContent</b>		
PDU file data content typedef for limit checking outgoing_file_chunk_size table value and set parameter command		529
<b>CF_CFDP_PduFileDataHeader</b>		
PDU file data header		530
<b>CF_CFDP_PduFileDirectiveHeader</b>		
Structure representing CFDP File Directive Header		530
<b>CF_CFDP_PduFin</b>		
Structure representing CFDP Finished PDU		530
<b>CF_CFDP_PduHeader</b>		
Structure representing base CFDP PDU header		531
<b>CF_CFDP_PduMd</b>		
Structure representing CFDP Metadata PDU		532
<b>CF_CFDP_PduNak</b>		
Structure representing CFDP Non-Acknowledge PDU		533
<b>CF_CFDP_R_SubstateDispatchTable_t</b>		
A dispatch table for receive file transactions, receive side		533
<b>CF_CFDP_S_SubstateRecvDispatchTable_t</b>		
A dispatch table for send file transactions, receive side		534
<b>CF_CFDP_S_SubstateSendDispatchTable_t</b>		
A dispatch table for send file transactions, transmit side		534
<b>CF_CFDP_SegmentRequest</b>		
Structure representing CFDP Segment Request		535
<b>CF_CFDP_Tick_args</b>		
Structure for use with the <b>CF_CFDP_DoTick()</b> functions		535
<b>CF_CFDP_tlv</b>		
Structure representing CFDP TLV Object format		536
<b>CF_CFDP_TxnRecvDispatchTable_t</b>		
A table of receive handler functions based on transaction state		537

<a href="#"><b>CF_CFDP_TxnSendDispatchTable_t</b></a>	A table of transmit handler functions based on transaction state	537
<a href="#"><b>CF_CFDP_uint16_t</b></a>	Encoded 16-bit value in the CFDP PDU	538
<a href="#"><b>CF_CFDP_uint32_t</b></a>	Encoded 32-bit value in the CFDP PDU	538
<a href="#"><b>CF_CFDP_uint64_t</b></a>	Encoded 64-bit value in the CFDP PDU	539
<a href="#"><b>CF_CFDP_uint8_t</b></a>	Encoded 8-bit value in the CFDP PDU	539
<a href="#"><b>CF_ChанAction_BoolArg</b></a>	An object to use with channel-scope actions requiring only a boolean argument	540
<a href="#"><b>CF_ChанAction_BoolMsgArg</b></a>	An object to use with channel-scope actions that require the message value	540
<a href="#"><b>CF_ChанAction_MsgArg</b></a>	An object to use with channel-scope actions that require the message value	541
<a href="#"><b>CF_ChанAction_SuspResArg</b></a>	An object to use with channel-scope actions for suspend/resume	541
<a href="#"><b>CF_Channel</b></a>	Channel state object	542
<a href="#"><b>CF_ChannelConfig</b></a>	Configuration entry for CFDP channel	543
<a href="#"><b>CF_Chunk</b></a>	Pairs an offset with a size to identify a specific piece of a file	546
<a href="#"><b>CF_ChunkList</b></a>	A list of CF_Chunk_t pairs	547
<a href="#"><b>CF_ChunkWrapper</b></a>	Wrapper around a CF_ChunkList_t object	547
<a href="#"><b>CF_CListNode</b></a>	Node link structure	548
<a href="#"><b>CF_Codec_BitField</b></a>		549
<a href="#"><b>CF_CodecState</b></a>	Tracks the current state of an encode or decode operation	549
<a href="#"><b>CF_ConfigTable</b></a>	Top-level CFDP configuration structure	550
<a href="#"><b>CF_Crc</b></a>	CRC state object	552

<b>CF_DecoderState</b>	Current state of a decode operation	552
<b>CF_DisableDequeueCmd</b>	DisableDequeue command structure	553
<b>CF_DisableDirPollingCmd</b>	DisableDirPolling command structure	554
<b>CF_DisableEngineCmd</b>	DisableEngine command structure	554
<b>CF_EnableDequeueCmd</b>	EnableDequeue command structure	555
<b>CF_EnableDirPollingCmd</b>	EnableDirPolling command structure	555
<b>CF_EnableEngineCmd</b>	EnableEngine command structure	556
<b>CF_EncoderState</b>	Current state of an encode operation	556
<b>CF_Engine</b>	An engine represents a pairing to a local EID	557
<b>CF_EotPacket</b>	End of transaction packet	559
<b>CF_EotPacket_Payload</b>	End of transaction packet	559
<b>CF_Flags_Common</b>	Data that applies to all types of transactions	561
<b>CF_Flags_Rx</b>	Flags that apply to receive transactions	563
<b>CF_Flags_Tx</b>	Flags that apply to send transactions	564
<b>CF_FreezeCmd</b>	Freeze command structure	566
<b>CF_GapComputeArgs_t</b>	Argument for Gap Compute function	566
<b>CF_GetParam_Payload</b>	Get parameter command structure	567
<b>CF_GetParamCmd</b>	Get parameter command structure	568
<b>CF_History</b>	CF History entry	568

<b>CF_HkChannel_Data</b>	Housekeeping channel data	570
<b>CF_HkCmdCounters</b>	Housekeeping command counters	572
<b>CF_HkCounters</b>	Housekeeping counters	573
<b>CF_HkFault</b>	Housekeeping fault counters	573
<b>CF_HkPacket</b>	Housekeeping packet	576
<b>CF_HkPacket_Payload</b>	Housekeeping packet	577
<b>CF_HkRecv</b>	Housekeeping received counters	578
<b>CF_HkSent</b>	Housekeeping sent counters	579
<b>CF_Input</b>	CF engine input state	580
<b>CF_Logical_IntHeader</b>	A union of all possible internal header types in a PDU	581
<b>CF_Logical_Lv</b>	Structure representing logical LV Object format	582
<b>CF_Logical_PduAck</b>	Structure representing CFDP Acknowledge PDU	583
<b>CF_Logical_PduBuffer</b>	Encapsulates the entire PDU information	584
<b>CF_Logical_PduEof</b>	Structure representing logical End of file PDU	585
<b>CF_Logical_PduFileDataHeader</b>		586
<b>CF_Logical_PduFileDirectiveHeader</b>	Structure representing logical File Directive header	587
<b>CF_Logical_PduFin</b>	Structure representing logical Finished PDU	588
<b>CF_Logical_PduHeader</b>	Structure representing base CFDP PDU header	589
<b>CF_Logical_PduMd</b>	Structure representing CFDP Metadata PDU	592

<b>CF_Logical_PduNak</b>	Structure representing logical Non-Acknowledge PDU	593
<b>CF_Logical_SegmentList</b>		594
<b>CF_Logical_SegmentRequest</b>	Structure representing logical Segment Request data	594
<b>CF_Logical_Tlv</b>	Structure representing logical TLV Object format	595
<b>CF_Logical_TlvData</b>	Union of various data items that may occur in a TLV item	596
<b>CF_Logical_TlvList</b>		597
<b>CF_NoopCmd</b>	Noop command structure	597
<b>CF_Output</b>	CF engine output state	598
<b>CF_PduCmdMsg</b>	PDU command encapsulation structure	598
<b>CF_PduTimMsg</b>	PDU send encapsulation structure	599
<b>CF_Playback</b>	CF Playback entry	600
<b>CF_PlaybackDirCmd</b>	Playback directory command structure	602
<b>CF_Poll</b>	CF Poll entry	602
<b>CF_PollDir</b>	Configuration entry for directory polling	603
<b>CF_PurgeQueueCmd</b>	PurgeQueue command structure	604
<b>CF_ResetCountersCmd</b>	Reset command structure	605
<b>CF_ResumeCmd</b>	Resume command structure	606
<b>CF_SendHkCmd</b>	Send Housekeeping Command	606
<b>CF_SetParam_Payload</b>	Set parameter command structure	607
<b>CF_SetParamCmd</b>	Set parameter command structure	608

<b>CF_StateData</b>	Summary of all possible transaction state information (tx and rx)	608
<b>CF_StateFlags</b>	Summary of all possible transaction flags (tx and rx)	610
<b>CF_SuspendCmd</b>	Suspend command structure	611
<b>CF_ThawCmd</b>	Thaw command structure	612
<b>CF_Timer</b>	Basic CF timer object	612
<b>CF_Transaction</b>	Transaction state object	613
<b>CF_Transaction_Payload</b>	Transaction command structure	617
<b>CF_Traverse_PriorityArg</b>	Argument structure for use with <a href="#">CF_CList_Traverse_R()</a>	618
<b>CF_Traverse_TransSeqArg</b>	Argument structure for use with <a href="#">CList_Traverse()</a>	618
<b>CF_Traverse_WriteHistoryFileArg</b>	Argument structure for use with <a href="#">CF_Traverse_WriteHistoryQueueEntryToFile()</a>	619
<b>CF_Traverse_WriteTxnFileArg</b>	Argument structure for use with <a href="#">CF_Traverse_WriteTxnQueueEntryToFile()</a>	620
<b>CF_TraverseAll_Arg</b>	Argument structure for use with <a href="#">CF_TraverseAllTransactions()</a>	621
<b>CF_TxFile_Payload</b>	Transmit file command structure	622
<b>CF_TxFileCmd</b>	Transmit file command structure	623
<b>CF_TxnFilenames</b>	Cache of source and destination filename	624
<b>CF_UnionArgs_Payload</b>	Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32	625
<b>CF_WakeupCmd</b>	Wake Up Command	625
<b>CF_WriteQueue_Payload</b>	Write Queue command structure	626
<b>CF_WriteQueueCmd</b>	Write Queue command structure	627

<b>CFE_Config_ArrayValue</b>		
Wrapper type for array configuration		628
<b>CFE_Config_IdNameEntry</b>		628
<b>CFE_Config_ValueBuffer</b>		629
<b>CFE_Config_ValueEntry</b>		629
<b>CFE_ES_AppInfo</b>		
Application Information		630
<b>CFE_ES_AppNameCmd_Payload</b>		
Generic application name command payload		634
<b>CFE_ES_AppReloadCmd_Payload</b>		
Reload Application Command Payload		634
<b>CFE_ES_BlockStats</b>		
Block statistics		635
<b>CFE_ES_CDSRegDumpRec</b>		
CDS Register Dump Record		636
<b>CFE_ES_ClearERLogCmd</b>		637
<b>CFE_ES_ClearSysLogCmd</b>		637
<b>CFE_ES_DeleteCDSCmd</b>		
Delete Critical Data Store Command		638
<b>CFE_ES_DeleteCDSCmd_Payload</b>		
Delete Critical Data Store Command Payload		638
<b>CFE_ES_DumpCDSRegistryCmd</b>		
Dump CDS Registry Command		639
<b>CFE_ES_DumpCDSRegistryCmd_Payload</b>		
Dump CDS Registry Command Payload		639
<b>CFE_ES_FileNameCmd</b>		
Generic file name command		640
<b>CFE_ES_FileNameCmd_Payload</b>		
Generic file name command payload		641
<b>CFE_ES_HousekeepingTIm</b>		641
<b>CFE_ES_HousekeepingTIm_Payload</b>		642
<b>CFE_ES_MemAddress_t</b>		
Type used for memory addresses in command and telemetry messages		650
<b>CFE_ES_MemOffset_t</b>		
Type used for memory sizes and offsets in commands and telemetry		650
<b>CFE_ES_MemPoolStats</b>		
Memory Pool Statistics		651

<a href="#">CFE_ES_MemStatsTlm</a>	652
<a href="#">CFE_ES_NoopCmd</a>	653
<a href="#">CFE_ES_OneAppTlm</a>	653
<a href="#">CFE_ES_OneAppTlm_Payload</a>	654
<a href="#">CFE_ES_OverWriteSysLogCmd</a>	
Overwrite/Discard System Log Configuration Command Payload	654
<a href="#">CFE_ES_OverWriteSysLogCmd_Payload</a>	
Overwrite/Discard System Log Configuration Command Payload	655
<a href="#">CFE_ES_PoolAlign</a>	
Pool Alignment	656
<a href="#">CFE_ES_PoolStatsTlm_Payload</a>	656
<a href="#">CFE_ES_QueryAllCmd</a>	657
<a href="#">CFE_ES_QueryAllTasksCmd</a>	658
<a href="#">CFE_ES_QueryOneCmd</a>	658
<a href="#">CFE_ES_ReloadAppCmd</a>	
Reload Application Command	659
<a href="#">CFE_ES_ResetCountersCmd</a>	659
<a href="#">CFE_ES_ResetPRCountCmd</a>	660
<a href="#">CFE_ES_RestartAppCmd</a>	660
<a href="#">CFE_ES_RestartCmd</a>	
Restart cFE Command	661
<a href="#">CFE_ES_RestartCmd_Payload</a>	
Restart cFE Command Payload	661
<a href="#">CFE_ES_SendHkCmd</a>	662
<a href="#">CFE_ES_SendMemPoolStatsCmd</a>	
Send Memory Pool Statistics Command	662
<a href="#">CFE_ES_SendMemPoolStatsCmd_Payload</a>	
Send Memory Pool Statistics Command Payload	663
<a href="#">CFE_ES_SetMaxPRCountCmd</a>	
Set Maximum Processor Reset Count Command	664
<a href="#">CFE_ES_SetMaxPRCountCmd_Payload</a>	
Set Maximum Processor Reset Count Command Payload	664
<a href="#">CFE_ES_SetPerfFilterMaskCmd</a>	
Set Performance Analyzer Filter Mask Command	665

<b>CFE_ES_SetPerfFilterMaskCmd_Payload</b>	Set Performance Analyzer Filter Mask Command Payload	665
<b>CFE_ES_SetPerfTriggerMaskCmd</b>	Set Performance Analyzer Trigger Mask Command	666
<b>CFE_ES_SetPerfTrigMaskCmd_Payload</b>	Set Performance Analyzer Trigger Mask Command Payload	667
<b>CFE_ES_StartApp</b>	Start Application Command	667
<b>CFE_ES_StartAppCmd_Payload</b>	Start Application Command Payload	668
<b>CFE_ES_StartPerfCmd_Payload</b>	Start Performance Analyzer Command Payload	669
<b>CFE_ES_StartPerfDataCmd</b>	Start Performance Analyzer Command	669
<b>CFE_ES_StopAppCmd</b>		670
<b>CFE_ES_StopPerfCmd_Payload</b>	Stop Performance Analyzer Command Payload	671
<b>CFE_ES_StopPerfDataCmd</b>	Stop Performance Analyzer Command	671
<b>CFE_ES_TaskInfo</b>	Task Information	672
<b>CFE_ES_WriteERLogCmd</b>		673
<b>CFE_ES_WriteSysLogCmd</b>		674
<b>CFE_EVS_AddEventFilterCmd</b>		674
<b>CFE_EVS_AppDataCmd_Payload</b>	Write Event Services Application Information to File Command Payload	675
<b>CFE_EVS_AppNameBitMaskCmd_Payload</b>	Generic App Name and Bitmask Command Payload	676
<b>CFE_EVS_AppNameCmd_Payload</b>	Generic App Name Command Payload	676
<b>CFE_EVS_AppNameEventIDCmd_Payload</b>	Generic App Name and Event ID Command Payload	677
<b>CFE_EVS_AppNameEventIDMaskCmd_Payload</b>	Generic App Name, Event ID, Mask Command Payload	677
<b>CFE_EVS_AppTlmData</b>		678
<b>CFE_EVS_BinFilter</b>	Event message filter definition structure	679

<b>CFE_EVS_BitMaskCmd_Payload</b>	
Generic Bitmask Command Payload	680
<b>CFE_EVS_ClearLogCmd</b>	
	681
<b>CFE_EVS_DeleteEventFilterCmd</b>	
	681
<b>CFE_EVS_DisableAppEventsCmd</b>	
	682
<b>CFE_EVS_DisableAppEventTypeCmd</b>	
	682
<b>CFE_EVS_DisableEventTypeCmd</b>	
	683
<b>CFE_EVS_DisablePortsCmd</b>	
	683
<b>CFE_EVS_EnableAppEventsCmd</b>	
	684
<b>CFE_EVS_EnableAppEventTypeCmd</b>	
	684
<b>CFE_EVS_EnableEventTypeCmd</b>	
	685
<b>CFE_EVS_EnablePortsCmd</b>	
	686
<b>CFE_EVS_HousekeepingTIm</b>	
	686
<b>CFE_EVS_HousekeepingTIm_Payload</b>	
	687
<b>CFE_EVS_LogFileCmd_Payload</b>	
Write Event Log to File Command Payload	690
<b>CFE_EVS_LongEventTIm</b>	
	690
<b>CFE_EVS_LongEventTIm_Payload</b>	
	691
<b>CFE_EVS_NoopCmd</b>	
	692
<b>CFE_EVS_PacketID</b>	
	692
<b>CFE_EVS_ResetAllFiltersCmd</b>	
	694
<b>CFE_EVS_ResetAppCounterCmd</b>	
	694
<b>CFE_EVS_ResetCountersCmd</b>	
	695
<b>CFE_EVS_ResetFilterCmd</b>	
	695
<b>CFE_EVS_SendHkCmd</b>	
	696
<b>CFE_EVS_SetEventFormatCode_Payload</b>	
Set Event Format Mode Command Payload	696
<b>CFE_EVS_SetEventFormatModeCmd</b>	
Set Event Format Mode Command	697
<b>CFE_EVS_SetFilterCmd</b>	
	698
<b>CFE_EVS_SetLogMode_Payload</b>	
Set Log Mode Command Payload	698

<a href="#">CFE_EVS_SetLogModeCmd</a>	Set Log Mode Command	699
<a href="#">CFE_EVS_ShortEventTlm</a>		699
<a href="#">CFE_EVS_ShortEventTlm_Payload</a>		700
<a href="#">CFE_EVS_WriteAppDataFileCmd</a>	Write Event Services Application Information to File Command	700
<a href="#">CFE_EVS_WriteLogFileCmd</a>	Write Event Log to File Command	701
<a href="#">CFE_FS_FileWriteMetaData</a>	External Metadata/State object associated with background file writes	702
<a href="#">CFE_FS_Header</a>	Standard cFE File header structure definition	703
<a href="#">CFE_SB_AllSubscriptionsTlm</a>		704
<a href="#">CFE_SB_AllSubscriptionsTlm_Payload</a>		705
<a href="#">CFE_SB_DisableRouteCmd</a>		706
<a href="#">CFE_SB_DisableSubReportingCmd</a>		706
<a href="#">CFE_SB_EnableRouteCmd</a>		707
<a href="#">CFE_SB_EnableSubReportingCmd</a>		707
<a href="#">CFE_SB_HousekeepingTlm</a>		708
<a href="#">CFE_SB_HousekeepingTlm_Payload</a>		708
<a href="#">CFE_SB_Msg</a>	Software Bus generic message	712
<a href="#">CFE_SB_MsgId_t</a>	CFE_SB_MsgId_t type definition	713
<a href="#">CFE_SB_MsgMapFileEntry</a>	SB Map File Entry	713
<a href="#">CFE_SB_NoopCmd</a>		714
<a href="#">CFE_SB_PipeDepthStats</a>	SB Pipe Depth Statistics	714
<a href="#">CFE_SB_PipeInfoEntry</a>	SB Pipe Information File Entry	716
<a href="#">CFE_SB_Qos_t</a>	Quality Of Service Type Definition	717
<a href="#">CFE_SB_ResetCountersCmd</a>		718

<b>CFE_SB_RouteCmd_Payload</b>	Enable/Disable Route Command Payload	718
<b>CFE_SB_RoutingFileEntry</b>	SB Routing File Entry	719
<b>CFE_SB_SendHkCmd</b>		720
<b>CFE_SB_SendPrevSubsCmd</b>		721
<b>CFE_SB_SendSbStatsCmd</b>		721
<b>CFE_SB_SingleSubscriptionTlm</b>		721
<b>CFE_SB_SingleSubscriptionTlm_Payload</b>		722
<b>CFE_SB_StatsTlm</b>		723
<b>CFE_SB_StatsTlm_Payload</b>		724
<b>CFE_SB_SubEntries</b>	SB Previous Subscriptions Entry	727
<b>CFE_SB_WriteFileInfoCmd_Payload</b>	Write File Info Command Payload	728
<b>CFE_SB_WriteMapInfoCmd</b>		728
<b>CFE_SB_WritePipeInfoCmd</b>		729
<b>CFE_SB_WriteRoutingInfoCmd</b>		729
<b>CFE_TBL_AbortLoadCmd</b>	Abort Load Command	730
<b>CFE_TBL_AbortLoadCmd_Payload</b>	Abort Load Command Payload	731
<b>CFE_TBL_ActivateCmd</b>	Activate Table Command	731
<b>CFE_TBL_ActivateCmd_Payload</b>	Activate Table Command Payload	732
<b>CFE_TBL_CombinedFileHdr</b>	Complete header for CFE table files	732
<b>CFE_TBL_DeleteCDSCmd_Payload</b>	Delete Critical Table CDS Command Payload	733
<b>CFE_TBL_DeleteCDSCmd</b>	Delete Critical Table CDS Command	733
<b>CFE_TBL_DumpCmd</b>		734
<b>CFE_TBL_DumpCmd_Payload</b>	Dump Table Command Payload	734

<b>CFE_TBL_DumpRegistryCmd</b>	
Dump Registry Command	735
<b>CFE_TBL_DumpRegistryCmd_Payload</b>	
Dump Registry Command Payload	736
<b>CFE_TBL_File_Hdr</b>	
The definition of the header fields that are included in CFE Table Data files	736
<b>CFE_TBL_FileDef</b>	
Table File summary object	737
<b>CFE_TBL_HousekeepingTlm</b>	
<b>CFE_TBL_HousekeepingTlm_Payload</b>	739
<b>CFE_TBL_Info</b>	
Table Info	743
<b>CFE_TBL_LoadCmd</b>	
Load Table Command	745
<b>CFE_TBL_LoadCmd_Payload</b>	
Load Table Command Payload	746
<b>CFE_TBL_NoopCmd</b>	746
<b>CFE_TBL_NotifyCmd</b>	747
<b>CFE_TBL_NotifyCmd_Payload</b>	
Table Management Notification Command Payload	747
<b>CFE_TBL_ResetCountersCmd</b>	748
<b>CFE_TBL_SendHkCmd</b>	748
<b>CFE_TBL_SendRegistryCmd</b>	
Send Table Registry Command	749
<b>CFE_TBL_SendRegistryCmd_Payload</b>	
Send Table Registry Command Payload	750
<b>CFE_TBL_TableRegistryTlm</b>	750
<b>CFE_TBL_TblRegPacket_Payload</b>	751
<b>CFE_TBL_ValidateCmd</b>	
Validate Table Command	754
<b>CFE_TBL_ValidateCmd_Payload</b>	
Validate Table Command Payload	754
<b>CFE_TIME_AddAdjustCmd</b>	755
<b>CFE_TIME_AddDelayCmd</b>	756
<b>CFE_TIME_AddOneHzAdjustmentCmd</b>	756

<b>CFE_TIME_DiagnosticTlm</b>	757
<b>CFE_TIME_DiagnosticTlm_Payload</b>	757
<b>CFE_TIME_FakeToneCmd</b>	766
<b>CFE_TIME_HousekeepingTlm</b>	766
<b>CFE_TIME_HousekeepingTlm_Payload</b>	767
<b>CFE_TIME_LeapsCmd_Payload</b>	
Set leap seconds command payload	768
<b>CFE_TIME_NoopCmd</b>	769
<b>CFE_TIME_OneHzAdjustmentCmd_Payload</b>	
Generic seconds, subseconds command payload	769
<b>CFE_TIME_OneHzCmd</b>	770
<b>CFE_TIME_ResetCountersCmd</b>	770
<b>CFE_TIME_SendDiagnosticCmd</b>	771
<b>CFE_TIME_SendHkCmd</b>	771
<b>CFE_TIME_SetLeapSecondsCmd</b>	
Set leap seconds command	772
<b>CFE_TIME_SetMETCmd</b>	772
<b>CFE_TIME_SetSignalCmd</b>	
Set tone signal source command	773
<b>CFE_TIME_SetSourceCmd</b>	
Set time data source command	773
<b>CFE_TIME_SetStateCmd</b>	
Set clock state command	774
<b>CFE_TIME_SetSTCFCmd</b>	775
<b>CFE_TIME_SetTimeCmd</b>	775
<b>CFE_TIME_SignalCmd_Payload</b>	
Set tone signal source command payload	776
<b>CFE_TIME_SourceCmd_Payload</b>	
Set time data source command payload	776
<b>CFE_TIME_StateCmd_Payload</b>	
Set clock state command payload	777
<b>CFE_TIME_SubAdjustCmd</b>	778
<b>CFE_TIME_SubDelayCmd</b>	778
<b>CFE_TIME_SubOneHzAdjustmentCmd</b>	779

<b>CFE_TIME_SysTime</b>	Data structure used to hold system time values	779
<b>CFE_TIME_TimeCmd_Payload</b>	Generic seconds, microseconds command payload	780
<b>CFE_TIME_ToneDataCmd</b>	Time at tone data command	781
<b>CFE_TIME_ToneDataCmd_Payload</b>	Time at tone data command payload	781
<b>CFE_TIME_ToneSignalCmd</b>		782
<b>OS_bin_sem_prop_t</b>	OSAL binary semaphore properties	782
<b>OS_condvar_prop_t</b>	OSAL condition variable properties	783
<b>OS_count_sem_prop_t</b>	OSAL counting semaphore properties	784
<b>os_dirent_t</b>	Directory entry	784
<b>OS_FdSet</b>	An abstract structure capable of holding several OSAL IDs	785
<b>OS_file_prop_t</b>	OSAL file properties	785
<b>os_fsinfo_t</b>	OSAL file system info	786
<b>os_fstat_t</b>	File system status	787
<b>OS_heap_prop_t</b>	OSAL heap properties	788
<b>OS_module_address_t</b>	OSAL module address properties	788
<b>OS_module_prop_t</b>	OSAL module properties	789
<b>OS_mut_sem_prop_t</b>	OSAL mutex properties	790
<b>OS_queue_prop_t</b>	OSAL queue properties	791
<b>OS_rwlock_prop_t</b>	OSAL rwlock properties	791

<a href="#"><b>OS_SockAddr_t</b></a> Encapsulates a generic network address	792
<a href="#"><b>OS_SockAddrData_t</b></a> Storage buffer for generic network address	792
<a href="#"><b>OS_socket_optval</b></a> Socket option value	793
<a href="#"><b>OS_socket_prop_t</b></a> Encapsulates socket properties	794
<a href="#"><b>OS_static_symbol_record_t</b></a> Associates a single symbol name with a memory address	794
<a href="#"><b>OS_statvfs_t</b></a>	795
<a href="#"><b>OS_task_prop_t</b></a> OSAL task properties	796
<a href="#"><b>OS_time_t</b></a> OSAL time interval structure	796
<a href="#"><b>OS_timebase_prop_t</b></a> Time base properties	797
<a href="#"><b>OS_timer_prop_t</b></a> Timer properties	798

## 9 File Index

### 9.1 File List

Here is a list of all files with brief descriptions:

<a href="#">apps/cf/config/default_cf_extern_typedefs.h</a>	799
<a href="#">apps/cf/config/default_cf_fcncode_values.h</a>	801
<a href="#">apps/cf/config/default_cf_interface_cfg_values.h</a>	802
<a href="#">apps/cf/config/default_cf_internal_cfg_values.h</a>	803
<a href="#">apps/cf/config/default_cf_mission_cfg.h</a>	803
<a href="#">apps/cf/config/default_cf_msg.h</a>	804
<a href="#">apps/cf/config/default_cf_msgdefs.h</a>	804
<a href="#">apps/cf/config/default_cf_msqid_values.h</a>	806
<a href="#">apps/cf/config/default_cf_msgids.h</a>	807
<a href="#">apps/cf/config/default_cf_msgstruct.h</a>	807
<a href="#">apps/cf/config/default_cf_platform_cfg.h</a>	810

<a href="#">apps/cf/config/default_cf_tbl.h</a>	811
<a href="#">apps/cf/config/default_cf_tbldefs.h</a>	812
<a href="#">apps/cf/config/default_cf_tblstruct.h</a>	812
<a href="#">apps/cf/config/default_cf_topicid_values.h</a>	813
<a href="#">apps/cf/config/eds_cf_extern_typedefs.h</a>	813
<a href="#">apps/cf/config/eds_cf_fcncode_values.h</a>	815
<a href="#">apps/cf/config/eds_cf_interface_cfg_values.h</a>	815
<a href="#">apps/cf/config/eds_cf_msgdefs.h</a>	816
<a href="#">apps/cf/config/eds_cf_msgstruct.h</a>	816
<a href="#">apps/cf/config/eds_cf_tbldefs.h</a>	816
<a href="#">apps/cf/config/eds_cf_tblstruct.h</a>	816
<a href="#">apps/cf/config/eds_cf_topicid_values.h</a>	816
<a href="#">apps/cf/fsw/inc/cf_eventids.h</a>	817
<a href="#">apps/cf/fsw/inc/cf_fcncodes.h</a>	823
<a href="#">apps/cf/fsw/inc/cf_interface_cfg.h</a>	824
<a href="#">apps/cf/fsw/inc/cf_internal_cfg.h</a>	825
<a href="#">apps/cf/fsw/inc/cf_perfids.h</a>	826
<a href="#">apps/cf/fsw/inc/cf_topicids.h</a>	826
<a href="#">apps/cf/fsw/src/cf_app.c</a>	829
<a href="#">apps/cf/fsw/src/cf_app.h</a>	835
<a href="#">apps/cf/fsw/src/cf_assert.h</a>	843
<a href="#">apps/cf/fsw/src/cf_cfdp.c</a>	844
<a href="#">apps/cf/fsw/src/cf_cfdp.h</a>	897
<a href="#">apps/cf/fsw/src/cf_cfdp_dispatch.c</a>	947
<a href="#">apps/cf/fsw/src/cf_cfdp_dispatch.h</a>	949
<a href="#">apps/cf/fsw/src/cf_cfdp_pdu.h</a>	953
<a href="#">apps/cf/fsw/src/cf_cfdp_r.c</a>	960
<a href="#">apps/cf/fsw/src/cf_cfdp_r.h</a>	980
<a href="#">apps/cf/fsw/src/cf_cfdp_s.c</a>	997
<a href="#">apps/cf/fsw/src/cf_cfdp_s.h</a>	1013

apps/cf/fsw/src/ <a href="#">cf_cfdp_sbintf.c</a>	1027
apps/cf/fsw/src/ <a href="#">cf_cfdp_sbintf.h</a>	1031
apps/cf/fsw/src/ <a href="#">cf_cfdp_types.h</a>	1035
apps/cf/fsw/src/ <a href="#">cf_chunk.c</a>	1042
apps/cf/fsw/src/ <a href="#">cf_chunk.h</a>	1052
apps/cf/fsw/src/ <a href="#">cf_clist.c</a>	1063
apps/cf/fsw/src/ <a href="#">cf_clist.h</a>	1068
apps/cf/fsw/src/ <a href="#">cf_cmd.c</a>	1074
apps/cf/fsw/src/ <a href="#">cf_cmd.h</a>	1105
apps/cf/fsw/src/ <a href="#">cf_codec.c</a>	1137
apps/cf/fsw/src/ <a href="#">cf_codec.h</a>	1168
apps/cf/fsw/src/ <a href="#">cf_crc.c</a>	1198
apps/cf/fsw/src/ <a href="#">cf_crc.h</a>	1199
apps/cf/fsw/src/ <a href="#">cf_dispatch.c</a>	1201
apps/cf/fsw/src/ <a href="#">cf_dispatch.h</a>	1204
apps/cf/fsw/src/ <a href="#">cf_eds_dispatch.c</a>	1206
apps/cf/fsw/src/ <a href="#">cf_logical_pdu.h</a>	1209
apps/cf/fsw/src/ <a href="#">cf_timer.c</a>	1213
apps/cf/fsw/src/ <a href="#">cf_timer.h</a>	1215
apps/cf/fsw/src/ <a href="#">cf_utils.c</a>	1218
apps/cf/fsw/src/ <a href="#">cf_utils.h</a>	1237
apps/cf/fsw/src/ <a href="#">cf_verify.h</a>	1260
apps/cf/fsw/src/ <a href="#">cf_version.h</a>	1260
apps/cf/fsw/tables/ <a href="#">cf_def_config.c</a>	1261
build/osal_public_api/inc/ <a href="#">osconfig.h</a>	1261
cfe/cmake/sample_defs/ <a href="#">cfe_perfids.h</a>	1267
cfe/cmake/sample_defs/ <a href="#">example_2x32bit_cfe_es_memaddress.h</a>	1269
cfe/cmake/sample_defs/ <a href="#">example_32bit_cfe_es_memaddress.h</a>	1271
cfe/cmake/sample_defs/ <a href="#">example_64bit_cfe_es_memaddress.h</a>	1273
cfe/cmake/sample_defs/ <a href="#">example_mission_cfg.h</a>	1274

<a href="#">cfe/cmake/sample_defs/example_platform_cfg.h</a>	1285
<a href="#">cfe/modules/config/fsw/inc/cfe_config_external.h</a>	1329
<a href="#">cfe/modules/config/fsw/inc/cfe_config_init.h</a>	1329
<a href="#">cfe/modules/config/fsw/inc/cfe_config_lookup.h</a>	1330
<a href="#">cfe/modules/config/fsw/inc/cfe_config_nametable.h</a>	1330
<a href="#">cfe/modules/config/fsw/inc/cfe_config_set.h</a>	1331
<a href="#">cfe/modules/config/fsw/inc/cfe_config_table.h</a>	1332
<a href="#">cfe/modules/core_api/config/default_cfe_core_api_base_msgid_values.h</a>	1333
<a href="#">cfe/modules/core_api/config/default_cfe_core_api_interface_cfg_values.h</a>	1333
<a href="#">cfe/modules/core_api/config/default_cfe_core_api_msgid_mapping.h</a>	1334
<a href="#">cfe/modules/core_api/config/default_cfe_mission_cfg.h</a>	1335
<a href="#">cfe/modules/core_api/config/default_cfe_msgids.h</a>	1336
<a href="#">cfe/modules/core_api/fsw/inc/cfe.h</a>	1336
<a href="#">cfe/modules/core_api/fsw/inc/cfe_config.h</a>	1336
<a href="#">cfe/modules/core_api/fsw/inc/cfe_config_api_typedefs.h</a>	1340
<a href="#">cfe/modules/core_api/fsw/inc/cfe_core_api_base_msgids.h</a>	1341
<a href="#">cfe/modules/core_api/fsw/inc/cfe_core_api_interface_cfg.h</a>	1342
<a href="#">cfe/modules/core_api/fsw/inc/cfe_endian.h</a>	1344
<a href="#">cfe/modules/core_api/fsw/inc/cfe_error.h</a>	1345
<a href="#">cfe/modules/core_api/fsw/inc/cfe_es.h</a>	1353
<a href="#">cfe/modules/core_api/fsw/inc/cfe_es_api_typedefs.h</a>	1357
<a href="#">cfe/modules/core_api/fsw/inc/cfe_evs.h</a>	1362
<a href="#">cfe/modules/core_api/fsw/inc/cfe_evs_api_typedefs.h</a>	1363
<a href="#">cfe/modules/core_api/fsw/inc/cfe_fs.h</a>	1366
<a href="#">cfe/modules/core_api/fsw/inc/cfe_fs_api_typedefs.h</a>	1366
<a href="#">cfe/modules/core_api/fsw/inc/cfe_msg.h</a>	1369
<a href="#">cfe/modules/core_api/fsw/inc/cfe_msg_api_typedefs.h</a>	1371
<a href="#">cfe/modules/core_api/fsw/inc/cfe_resourceid.h</a>	1375
<a href="#">cfe/modules/core_api/fsw/inc/cfe_resourceid_api_typedefs.h</a>	1382
<a href="#">cfe/modules/core_api/fsw/inc/cfe_sb.h</a>	1383

cfe/modules/core_api/fsw/inc/cfe_sb_api_typedefs.h	1386
cfe/modules/core_api/fsw/inc/cfe_tbl.h	1389
cfe/modules/core_api/fsw/inc/cfe_tbl_api_typedefs.h	1391
cfe/modules/core_api/fsw/inc/cfe_tbl_filedef.h	1394
cfe/modules/core_api/fsw/inc/cfe_time.h	1395
cfe/modules/core_api/fsw/inc/cfe_time_api_typedefs.h	1397
cfe/modules/core_api/fsw/inc/cfe_version.h	1399
cfe/modules/es/config/default_cfe_es_extern_typedefs.h	1401
cfe/modules/es/config/default_cfe_es_fcncode_values.h	1408
cfe/modules/es/config/default_cfe_es_interface_cfg_values.h	1410
cfe/modules/es/config/default_cfe_es_internal_cfg_values.h	1410
cfe/modules/es/config/default_cfe_es_memaddress.h	1410
cfe/modules/es/config/default_cfe_es_mission_cfg.h	1412
cfe/modules/es/config/default_cfe_es_msg.h	1412
cfe/modules/es/config/default_cfe_es_msgdefs.h	1413
cfe/modules/es/config/default_cfe_es_msgid_values.h	1416
cfe/modules/es/config/default_cfe_es_msgids.h	1417
cfe/modules/es/config/default_cfe_es_msgstruct.h	1418
cfe/modules/es/config/default_cfe_es_platform_cfg.h	1422
cfe/modules/es/config/default_cfe_es_topicid_values.h	1422
cfe/modules/es/fsw/inc/cfe_es_eventids.h	1422
cfe/modules/es/fsw/inc/cfe_es_fcncodes.h	1448
cfe/modules/es/fsw/inc/cfe_es_interface_cfg.h	1469
cfe/modules/es/fsw/inc/cfe_es_internal_cfg.h	1473
cfe/modules/es/fsw/inc/cfe_es_topicids.h	1505
cfe/modules/evs/config/default_cfe_evs_extern_typedefs.h	1506
cfe/modules/evs/config/default_cfe_evs_fcncode_values.h	1509
cfe/modules/evs/config/default_cfe_evs_interface_cfg_values.h	1511
cfe/modules/evs/config/default_cfe_evs_internal_cfg_values.h	1511
cfe/modules/evs/config/default_cfe_evs_mission_cfg.h	1511

cfe/modules/evs/config/default_cfe_evs_msg.h	1512
cfe/modules/evs/config/default_cfe_evs_msgdefs.h	1512
cfe/modules/evs/config/default_cfe_evsmsgid_values.h	1516
cfe/modules/evs/config/default_cfe_evs_msgids.h	1516
cfe/modules/evs/config/default_cfe_evs_msgstruct.h	1517
cfe/modules/evs/config/default_cfe_evs_platform_cfg.h	1520
cfe/modules/evs/config/default_cfe_evs_topicid_values.h	1520
cfe/modules/evs/fsw/inc/cfe_evs_eventids.h	1521
cfe/modules/evs/fsw/inc/cfe_evs_fcncodes.h	1533
cfe/modules/evs/fsw/inc/cfe_evs_interface_cfg.h	1551
cfe/modules/evs/fsw/inc/cfe_evs_internal_cfg.h	1552
cfe/modules/evs/fsw/inc/cfe_evs_topicids.h	1558
cfe/modules/fs/config/default_cfe_fs_extern_typedefs.h	1560
cfe/modules/fs/config/default_cfe_fs_filedef.h	1560
cfe/modules/fs/config/default_cfe_fs_interface_cfg_values.h	1562
cfe/modules/fs/config/default_cfe_fs_mission_cfg.h	1562
cfe/modules/fs/fsw/inc/cfe_fs_interface_cfg.h	1563
cfe/modules/msg/fsw/inc/ccsds_hdr.h	1564
cfe/modules/resourceid/fsw/inc/cfe_core_resourceid_basevalues.h	1564
cfe/modules/resourceid/fsw/inc/cfe_resourceid_basevalue.h	1565
cfe/modules/sb/config/default_cfe_sb_extern_typedefs.h	1566
cfe/modules/sb/config/default_cfe_sb_fcncode_values.h	1568
cfe/modules/sb/config/default_cfe_sb_interface_cfg_values.h	1569
cfe/modules/sb/config/default_cfe_sb_internal_cfg_values.h	1570
cfe/modules/sb/config/default_cfe_sb_mission_cfg.h	1570
cfe/modules/sb/config/default_cfe_sb_msg.h	1570
cfe/modules/sb/config/default_cfe_sb_msgdefs.h	1571
cfe/modules/sb/config/default_cfe_sbmsgid_values.h	1573
cfe/modules/sb/config/default_cfe_sb_msgids.h	1574
cfe/modules/sb/config/default_cfe_sb_msgstruct.h	1575

<a href="#">cfe/modules/sb/config/default_cfe_sb_platform_cfg.h</a>	1577
<a href="#">cfe/modules/sb/config/default_cfe_sb_topicid_values.h</a>	1577
<a href="#">cfe/modules/sb/fsw/inc/cfe_sb_eventids.h</a>	1577
<a href="#">cfe/modules/sb/fsw/inc/cfe_sb_fcncodes.h</a>	1597
<a href="#">cfe/modules/sb/fsw/inc/cfe_sb_interface_cfg.h</a>	1606
<a href="#">cfe/modules/sb/fsw/inc/cfe_sb_internal_cfg.h</a>	1608
<a href="#">cfe/modules/sb/fsw/inc/cfe_sb_topicids.h</a>	1623
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_extern_typedefs.h</a>	1625
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_fcncode_values.h</a>	1627
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_interface_cfg_values.h</a>	1628
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_internal_cfg_values.h</a>	1628
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_mission_cfg.h</a>	1629
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_msg.h</a>	1630
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h</a>	1630
<a href="#">cfe/modules/tbl/config/default_cfe_tblmsgid_values.h</a>	1633
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_msgids.h</a>	1633
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h</a>	1634
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_platform_cfg.h</a>	1636
<a href="#">cfe/modules/tbl/config/default_cfe_tbl_topicid_values.h</a>	1645
<a href="#">cfe/modules/tbl/fsw/inc/cfe_tbl_eventids.h</a>	1645
<a href="#">cfe/modules/tbl/fsw/inc/cfe_tbl_fcncodes.h</a>	1665
<a href="#">cfe/modules/tbl/fsw/inc/cfe_tbl_interface_cfg.h</a>	1674
<a href="#">cfe/modules/tbl/fsw/inc/cfe_tbl_internal_cfg.h</a>	1676
<a href="#">cfe/modules/tbl/fsw/inc/cfe_tbl_topicids.h</a>	1685
<a href="#">cfe/modules/time/config/default_cfe_time_extern_typedefs.h</a>	1686
<a href="#">cfe/modules/time/config/default_cfe_time_fcncode_values.h</a>	1691
<a href="#">cfe/modules/time/config/default_cfe_time_interface_cfg_values.h</a>	1692
<a href="#">cfe/modules/time/config/default_cfe_time_internal_cfg_values.h</a>	1692
<a href="#">cfe/modules/time/config/default_cfe_time_mission_cfg.h</a>	1693
<a href="#">cfe/modules/time/config/default_cfe_time_msg.h</a>	1693

<a href="#">cfe/modules/time/config/default_cfe_time_msgdefs.h</a>	1694
<a href="#">cfe/modules/time/config/default_cfe_timemsgid_values.h</a>	1696
<a href="#">cfe/modules/time/config/default_cfe_time_msgids.h</a>	1697
<a href="#">cfe/modules/time/config/default_cfe_time_msgstruct.h</a>	1698
<a href="#">cfe/modules/time/config/default_cfe_time_platform_cfg.h</a>	1701
<a href="#">cfe/modules/time/config/default_cfe_time_topicid_values.h</a>	1701
<a href="#">cfe/modules/time/fsw/inc/cfe_time_eventids.h</a>	1702
<a href="#">cfe/modules/time/fsw/inc/cfe_time_fcncodes.h</a>	1712
<a href="#">cfe/modules/time/fsw/inc/cfe_time_interface_cfg.h</a>	1727
<a href="#">cfe/modules/time/fsw/inc/cfe_time_internal_cfg.h</a>	1735
<a href="#">cfe/modules/time/fsw/inc/cfe_time_topicids.h</a>	1744
<a href="#">osal/src/os/inc/common_types.h</a>	1746
<a href="#">osal/src/os/inc/osapi-binsem.h</a>	1751
<a href="#">osal/src/os/inc/osapi-bsp.h</a>	1752
<a href="#">osal/src/os/inc/osapi-clock.h</a>	1752
<a href="#">osal/src/os/inc/osapi-common.h</a>	1755
<a href="#">osal/src/os/inc/osapi-condvar.h</a>	1757
<a href="#">osal/src/os/inc/osapi-constants.h</a>	1758
<a href="#">osal/src/os/inc/osapi-countsem.h</a>	1759
<a href="#">osal/src/os/inc/osapi-dir.h</a>	1759
<a href="#">osal/src/os/inc/osapi-error.h</a>	1760
<a href="#">osal/src/os/inc/osapi-file.h</a>	1763
<a href="#">osal/src/os/inc/osapi-fs.h</a>	1767
<a href="#">osal/src/os/inc/osapi-heap.h</a>	1768
<a href="#">osal/src/os/inc/osapi-idmap.h</a>	1768
<a href="#">osal/src/os/inc/osapi-macros.h</a>	1770
<a href="#">osal/src/os/inc/osapi-module.h</a>	1772
<a href="#">osal/src/os/inc/osapi-mutex.h</a>	1773
<a href="#">osal/src/os/inc/osapi-network.h</a>	1774
<a href="#">osal/src/os/inc/osapi-printf.h</a>	1774

<a href="#">osal/src/os/inc/osapi-queue.h</a>	1775
<a href="#">osal/src/os/inc/osapi-rwlock.h</a>	1775
<a href="#">osal/src/os/inc/osapi-select.h</a>	1776
<a href="#">osal/src/os/inc/osapi-shell.h</a>	1777
<a href="#">osal/src/os/inc/osapi-sockets.h</a>	1777
<a href="#">osal/src/os/inc/osapi-task.h</a>	1781
<a href="#">osal/src/os/inc/osapi-timebase.h</a>	1783
<a href="#">osal/src/os/inc/osapi-timer.h</a>	1784
<a href="#">osal/src/os/inc/osapi-version.h</a>	1785
<a href="#">osal/src/os/inc/osapi.h</a>	1788
<a href="#">psp/fsw/inc/cfe_psp.h</a>	1789
<a href="#">psp/fsw/inc/cfe_psp_cache_api.h</a>	1790
<a href="#">psp/fsw/inc/cfe_psp_cds_api.h</a>	1792
<a href="#">psp/fsw/inc/cfe_psp_eepromaccess_api.h</a>	1793
<a href="#">psp/fsw/inc/cfe_psp_endian.h</a>	1796
<a href="#">psp/fsw/inc/cfe_psp_error.h</a> CFE PSP Error header	1798
<a href="#">psp/fsw/inc/cfe_psp_exception_api.h</a>	1801
<a href="#">psp/fsw/inc/cfe_psp_id_api.h</a>	1803
<a href="#">psp/fsw/inc/cfe_psp_memaccess_api.h</a>	1803
<a href="#">psp/fsw/inc/cfe_psp_memrange_api.h</a>	1807
<a href="#">psp/fsw/inc/cfe_psp_port_api.h</a>	1813
<a href="#">psp/fsw/inc/cfe_psp_ss्र_api.h</a>	1815
<a href="#">psp/fsw/inc/cfe_psp_timertick_api.h</a>	1816
<a href="#">psp/fsw/inc/cfe_psp_version_api.h</a>	1818
<a href="#">psp/fsw/inc/cfe_psp_watchdog_api.h</a>	1819

## 10 Module Documentation

### 10.1 CFS CFDP Event IDs

#### Macros

- #define CF\_INIT\_INF\_EID 20

- `#define CF_INIT_TBL_CHECK_REL_ERR_EID 21`  
*CF Initialization Event ID.*
- `#define CF_INIT_TBL_CHECK_MAN_ERR_EID 22`  
*CF Check Table Manage Failed Event ID.*
- `#define CF_INIT_TBL_CHECK_GA_ERR_EID 23`  
*CF Check Table Get Address Failed Event ID.*
- `#define CF_INIT_TBL_REG_ERR_EID 24`  
*CF Table Registration At Initialization Failed Event ID.*
- `#define CF_INIT_TBL_LOAD_ERR_EID 25`  
*CF Table Load At Initialization Failed Event ID.*
- `#define CF_INIT_TBL_MANAGE_ERR_EID 26`  
*CF Table Manage At Initialization Failed Event ID.*
- `#define CF_INIT_TBL_GETADDR_ERR_EID 27`  
*CF Table Get Address At Initialization Failed Event ID.*
- `#define CF_MID_ERR_EID 28`  
*CF Message ID Invalid Event ID.*
- `#define CF_INIT_MSG_RECV_ERR_EID 29`  
*CF SB Receive Buffer Failed Event ID.*
- `#define CF_INIT_SEM_ERR_EID 30`  
*CF Channel Semaphore Initialization Failed Event ID.*
- `#define CF_CR_CHANNEL_PIPE_ERR_EID 31`  
*CF Channel Create Pipe Failed Event ID.*
- `#define CF_INIT_SUB_ERR_EID 32`  
*CF Channel Message Subscription Failed Event ID.*
- `#define CF_INIT_TPS_ERR_EID 33`  
*CF Ticks Per Second Config Table Validation Failed Event ID.*
- `#define CF_INIT_CRC_ALIGN_ERR_EID 34`  
*CF CRC Bytes Per Wakeup Config Table Validation Failed Event ID.*
- `#define CF_INIT_OUTGOING_SIZE_ERR_EID 35`  
*CF Outgoing Chunk Size Config Table Validation Failed Event ID.*
- `#define CF_CR_PIPE_ERR_EID 36`  
*CF Create SB Command Pipe at Initialization Failed Event ID.*
- `#define CF_PDU_MD_RECVD_INF_EID 40`  
*CF Metadata PDU Received Event ID.*
- `#define CF_PDU_SHORT_HEADER_ERR_EID 41`  
*CF PDU Header Too Short Event ID.*
- `#define CF_PDU_MD_SHORT_ERR_EID 43`  
*CF Metadata PDU Too Short Event ID.*
- `#define CF_PDU_INVALID_SRC_LEN_ERR_EID 44`  
*CF Metadata PDU Source Filename Length Invalid Event ID.*
- `#define CF_PDU_INVALID_DST_LEN_ERR_EID 45`  
*CF Metadata PDU Destination Filename Length Invalid Event ID.*
- `#define CF_PDU_FD_SHORT_ERR_EID 46`  
*CF File Data PDU Too Short Event ID.*
- `#define CF_PDU_EOF_SHORT_ERR_EID 47`  
*CF End-Of-File PDU Too Short Event ID.*

- #define CF\_PDU\_ACK\_SHORT\_ERR\_EID 48  
*CF Acknowledgment PDU Too Short Event ID.*
- #define CF\_PDU\_FIN\_SHORT\_ERR\_EID 49  
*CF Finished PDU Too Short Event ID.*
- #define CF\_PDU\_NAK\_SHORT\_ERR\_EID 50  
*CF Negative Acknowledgment PDU Too Short Event ID.*
- #define CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID 54  
*CF File Data PDU Unsupported Option Event ID.*
- #define CF\_PDU\_LARGE\_FILE\_ERR\_EID 55  
*CF PDU Header Large File Flag Set Event ID.*
- #define CF\_PDU\_TRUNCATION\_ERR\_EID 56  
*CF PDU Header Field Truncation.*
- #define CF\_RESET\_FREED\_XACT\_DBG\_EID 59  
*Attempt to reset a transaction that has already been freed.*
- #define CF\_CFDP\_RX\_DROPPED\_ERR\_EID 60  
*CF PDU Received Without Existing Transaction, Dropped Due To Max RX Reached Event ID.*
- #define CF\_CFDP\_INVALID\_DST\_ERR\_EID 61  
*CF PDU Received With Invalid Destination Entity ID Event ID.*
- #define CF\_CFDP\_IDLE\_MD\_ERR\_EID 62  
*CF Invalid Metadata PDU Received On Idle Transaction Event ID.*
- #define CF\_CFDP\_FD\_UNHANDLED\_ERR\_EID 63  
*CF Non-metadata File Directive PDU Received On Idle Transaction Event ID.*
- #define CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID 64  
*CF Transmission Request Rejected Due To Max Commanded TX Reached Event ID.*
- #define CF\_CFDP\_OPENDIR\_ERR\_EID 65  
*CF Playback/Polling Directory Open Failed Event ID.*
- #define CF\_CFDP\_DIR\_SLOT\_ERR\_EID 66  
*CF Playback Request Rejected Due to Max Playback Directories Reached Event ID.*
- #define CF\_CFDP\_NO\_MSG\_ERR\_EID 67  
*CF No Message Buffer Available Event ID.*
- #define CF\_CFDP\_CLOSE\_ERR\_EID 68  
*CF Close File Failed Event ID.*
- #define CF\_CFDP\_NO\_CHUNKLIST\_AVAIL\_EID 69  
*CF No chunklist available.*
- #define CF\_CFDP\_R\_REQUEST\_MD\_INF\_EID 70  
*CF Requesting RX Metadata Event ID.*
- #define CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID 71  
*CF Creating Temp File For RX Transaction.*
- #define CF\_CFDP\_R\_NAK\_LIMIT\_ERR\_EID 72  
*CF RX Transaction NAK Limit Reached Event ID.*
- #define CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_EID 73  
*CF RX Transaction ACK Limit Reached Event ID.*
- #define CF\_CFDP\_R\_CRC\_ERR\_EID 74  
*CF RX Transaction CRC Mismatch Event ID.*
- #define CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID 75  
*CF RX File Data PDU Seek Failed Event ID.*
- #define CF\_CFDP\_R\_SEEK\_CRC\_ERR\_EID 76  
*CF RX File Data PDU Seek Failed Event ID.*

- `#define CF_CFDP_R_WRITE_ERR_EID 77`  
*CF RX Class 2 CRC Seek Failed Event ID.*
- `#define CF_CFDP_R_SIZE_MISMATCH_ERR_EID 78`  
*CF RX File Data PDU Write Failed Event ID.*
- `#define CF_CFDP_R_PDU_EOF_ERR_EID 79`  
*CF RX End-Of-File PDU File Size Mismatch Event ID.*
- `#define CF_CFDP_R_CREAT_ERR_EID 80`  
*CF RX Transaction File Create Failed Event ID.*
- `#define CF_CFDP_R_PDU_FINACK_ERR_EID 81`  
*CF Class 2 RX Transaction Invalid FIN-ACK PDU Event ID.*
- `#define CF_CFDP_R_EOF_MD_SIZE_ERR_EID 82`  
*CF RX Class 2 Metadata PDU Size Mismatch Event ID.*
- `#define CF_CFDP_R_RENAME_ERR_EID 83`  
*CF RX Class 2 Metadata PDU File Rename Failed Event ID.*
- `#define CF_CFDP_R_FILE_RETAINED_EID (84)`  
*CF File retained.*
- `#define CF_CFDP_R_NOT_RETAINED_EID (85)`  
*CF RX File not retained.*
- `#define CF_CFDP_R_READ_ERR_EID (86)`  
*CF Class 2 CRC Read From File Failed Event ID.*
- `#define CF_CFDP_R_DC_INV_ERR_EID 87`  
*CF RX Invalid File Directive PDU Code Received Event ID.*
- `#define CF_CFDP_R_INACT_TIMER_ERR_EID 88`  
*CF RX Inactivity Timer Expired Event ID.*
- `#define CF_CFDP_S_START_SEND_INF_EID 90`  
*CF TX Initiated Event ID.*
- `#define CF_CFDP_S_SEEK_FD_ERR_EID 91`  
*CF TX File Data PDU Seek Failed Event ID.*
- `#define CF_CFDP_S_READ_ERR_EID 92`  
*CF TX File Data PDU Read Failed Event ID.*
- `#define CF_CFDP_S_SEND_FD_ERR_EID 93`  
*CF TX File Data PDU Send Failed Event ID.*
- `#define CF_CFDP_S_ALREADY_OPEN_ERR_EID 94`  
*CF TX Metadata PDU File Already Open Event ID.*
- `#define CF_CFDP_S_OPEN_ERR_EID 95`  
*CF TX Metadata PDU File Open Failed Event ID.*
- `#define CF_CFDP_S_SEEK_END_ERR_EID 96`  
*CF TX Metadata PDU File Seek End Failed Event ID.*
- `#define CF_CFDP_S_SEEK_BEG_ERR_EID 97`  
*CF TX Metadata PDU File Seek Beginning Failed Event ID.*
- `#define CF_CFDP_S_SEND_MD_ERR_EID 98`  
*CF TX Metadata PDU Send Failed Event ID.*
- `#define CF_CFDP_S_INVALID_SR_ERR_EID 100`  
*CF TX Received NAK PDU Bad Segment Request Event ID.*
- `#define CF_CFDP_S_PDU_NAK_ERR_EID 101`  
*CF TX Received NAK PDU Invalid Event ID.*

- #define CF\_CFDP\_S\_PDU\_EOF\_ERR\_EID 102  
*CF TX Received EOF ACK PDU Invalid Event ID.*
- #define CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID 103  
*CF TX Received Early FIN PDU Event ID.*
- #define CF\_CFDP\_S\_DC\_INV\_ERR\_EID 104  
*CF Invalid TX File Directive PDU Code Event ID.*
- #define CF\_CFDP\_S\_NON\_FD\_PDU\_ERR\_EID 105  
*CF Received TX Non-File Directive PDU Event ID.*
- #define CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID 106  
*CF TX EOF PDU Send Limit Reached Event ID.*
- #define CF\_CFDP\_S\_INACT\_TIMER\_ERR\_EID 107  
*CF TX Inactivity Timer Expired Event ID.*
- #define CF\_CFDP\_S\_FILE\_MOVED\_EID (108)  
*CF TX File Moved.*
- #define CF\_CFDP\_S\_FILE\_REMOVED\_EID (109)  
*CF TX File Removed.*
- #define CF\_NOOP\_INF\_EID 110  
*CF NOOP Command Received Event ID.*
- #define CF\_RESET\_INF\_EID 111  
*CF Reset Counters Command Received Event ID.*
- #define CF\_CMD\_GETSET1\_INF\_EID 112  
*CF Set Parameter Command Received Event ID.*
- #define CF\_CMD\_GETSET2\_INF\_EID 113  
*CF Get Parameter Command Received Event ID.*
- #define CF\_CMD\_SUSPRES\_INF\_EID 114  
*CF Suspend/Resume Command Received Event ID.*
- #define CF\_CMD\_WQ\_INF\_EID 115  
*CF Write Queue Command Received Event ID.*
- #define CF\_CMD\_ENABLE\_ENGINE\_INF\_EID 116  
*CF Enable Engine Command Received Event ID.*
- #define CF\_CMD\_DISABLE\_ENGINE\_INF\_EID 117  
*CF Disable Engine Command Received Event ID.*
- #define CF\_CMD\_TX\_FILE\_INF\_EID 118  
*CF Transfer File Command Received Event ID.*
- #define CF\_CMD\_PLAYBACK\_DIR\_INF\_EID 119  
*CF Playback Directory Command Received Event ID.*
- #define CF\_CMD\_FREEZE\_INF\_EID 120  
*CF Freeze Command Received Event ID.*
- #define CF\_CMD\_THAW\_INF\_EID 121  
*CF Thaw Command Received Event ID.*
- #define CF\_CMD\_CANCEL\_INF\_EID 122  
*CF Cancel Command Received Event ID.*
- #define CF\_CMD\_ABANDON\_INF\_EID 123  
*CF Abandon Command Received Event ID.*
- #define CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID 124  
*CF Enable Dequeue Command Received Event ID.*
- #define CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID 125

- `#define CF_CMD_ENABLE_POLLDIR_INF_EID 126`  
*CF Disable Dequeue Command Received Event ID.*
- `#define CF_CMD_DISABLE_POLLDIR_INF_EID 127`  
*CF Enable Polldir Command Received Event ID.*
- `#define CF_CMD_PURGE_QUEUE_INF_EID 128`  
*CF Disable Polldir Command Received Event ID.*
- `#define CF_CMD_RESET_INVALID_ERR_EID 129`  
*CF Purge Queue Command Received Event ID.*
- `#define CF_CMD_CHAN_PARAM_ERR_EID 130`  
*CF Reset Counters Command Invalid Event ID.*
- `#define CF_CMD_TRANS_NOT_FOUND_ERR_EID 131`  
*CF Command Channel Invalid Event ID.*
- `#define CF_CMD_TSN_CHAN_INVALID_ERR_EID 132`  
*CF Command Transaction Invalid Event ID.*
- `#define CF_CMD_SUSPRES_SAME_INF_EID 133`  
*CF Command All Transaction Channel Invalid Event ID.*
- `#define CF_CMD_SUSPRES_CHAN_ERR_EID 134`  
*CF Suspend/Resume Command For Single Transaction State Unchanged Event ID.*
- `#define CF_CMD_POLLDIR_INVALID_ERR_EID 135`  
*CF Suspend/Resume Command No Matching Transaction Event ID.*
- `#define CF_CMD_PURGE_ARG_ERR_EID 136`  
*CF Enable/Disable Polling Directory Command Invalid Polling Directory Index Event ID.*
- `#define CF_CMD_WQ_CHAN_ERR_EID 137`  
*CF Purge Queue Command Invalid Argument Event ID.*
- `#define CF_CMD_WQ_ARGS_ERR_EID 138`  
*CF Write Queue Command Invalid Channel Event ID.*
- `#define CF_CMD_WQ_OPEN_ERR_EID 139`  
*CF Write Queue Command Invalid Queue Event ID.*
- `#define CF_CMD_WQ_WRITEQ_RX_ERR_EID 140`  
*CF Write Queue Command File Open Failed Event ID.*
- `#define CF_CMD_WQ_WRITEHIST_RX_ERR_EID 141`  
*CF Write Queue Command RX Active File Write Failed Event ID.*
- `#define CF_CMD_WQ_WRITEQ_TX_ERR_EID 142`  
*CF Write Queue Command RX History File Write Failed Event ID.*
- `#define CF_CMD_WQ_WRITEHIST_TX_ERR_EID 143`  
*CF Write Queue Command TX Active File Write Failed Event ID.*
- `#define CF_CMD_WQ_WRITEHIST_PEND_ERR_EID 144`  
*CF Write Queue Command TX Pending File Write Failed Event ID.*
- `#define CF_CMD_GETSET_VALIDATE_ERR_EID 145`  
*CF Set Parameter Command Parameter Validation Failed Event ID.*
- `#define CF_CMD_GETSET_PARAM_ERR_EID 146`  
*CF Set/Get Parameter Command Invalid Parameter ID Event ID.*
- `#define CF_CMD_GETSET_CHAN_ERR_EID 147`  
*CF Set/Get Parameter Command Invalid Channel Event ID.*
- `#define CF_CMD_ENABLE_ENGINE_ERR_EID 148`  
*CF Enable Engine Command Failed Event ID.*

- #define CF\_CMD\_ENG\_ALREADY\_ENA\_INF\_EID 149  
*CF Enable Engine Command Engine Already Enabled Event ID.*
- #define CF\_CMD\_ENG\_ALREADY\_DIS\_INF\_EID 150  
*CF Disable Engine Command Engine Already Disabled Event ID.*
- #define CF\_CMD\_LEN\_ERR\_EID 151  
*CF Command Length Verification Failed Event ID.*
- #define CF\_CC\_ERR\_EID 152  
*CF Command Code Invalid Event ID.*
- #define CF\_CMD\_WHIST\_WRITE\_ERR\_EID 153  
*CF Write Entry To File Failed Event ID.*
- #define CF\_CMD\_BAD\_PARAM\_ERR\_EID 154  
*CF Playback Dir Or TX File Command Bad Parameter Event ID.*
- #define CF\_CMD\_CANCEL\_CHAN\_ERR\_EID 155  
*CF Cancel Command No Matching Transaction Event ID.*
- #define CF\_CMD\_ABANDON\_CHAN\_ERR\_EID 156  
*CF Abandon Command No Matching Transaction Event ID.*
- #define CF\_CMD\_TX\_FILE\_ERR\_EID 157  
*CF Transfer File Command Failed Event ID.*
- #define CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID 158  
*CF Playback Directory Command Failed Event ID.*
- #define CF\_CMD\_FREEZE\_ERR\_EID 159  
*CF Freeze Command Failed Event ID.*
- #define CF\_CMD\_THAW\_ERR\_EID 160  
*CF Thaw Command Failed Event ID.*
- #define CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID 161  
*CF Enable Dequeue Command Failed Event ID.*
- #define CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID 162  
*CF Disable Dequeue Command Failed Event ID.*
- #define CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID 163  
*CF Enable Polldir Command Failed Event ID.*
- #define CF\_CMD\_DISABLE\_POLLDIR\_ERR\_EID 164  
*CF Disable Polldir Command Failed Event ID.*
- #define CF\_CMD\_PURGE\_QUEUE\_ERR\_EID 165  
*CF Purge Queue Command Failed Event ID.*
- #define CF\_EID\_INF\_CFDP\_BUF\_EXCEED 166  
*CF Move Path Length Verification Too Long Event ID.*

### 10.1.1 Detailed Description

### 10.1.2 Macro Definition Documentation

**10.1.2.1 CF\_CC\_ERR\_EID** #define CF\_CC\_ERR\_EID 152  
CF Command Code Invalid Event ID.

Type: ERROR

Cause:

Received command code unrecognized  
Definition at line 1424 of file cf\_eventids.h.

**10.1.2.2 CF\_CFDP\_CLOSE\_ERR\_EID** #define CF\_CFDP\_CLOSE\_ERR\_EID 68  
CF Close File Failed Event ID.

Type: ERROR

Cause:

Failure from file close call  
Definition at line 491 of file cf\_eventids.h.

**10.1.2.3 CF\_CFDP\_DIR\_SLOT\_ERR\_EID** #define CF\_CFDP\_DIR\_SLOT\_ERR\_EID 66  
CF Playback Request Rejected Due to Max Playback Directories Reached Event ID.

Type: ERROR

Cause:

Command request to playback a directory received when channel is already handling the maximum number of concurrent playback directories  
Definition at line 469 of file cf\_eventids.h.

**10.1.2.4 CF\_CFDP\_FD\_UNHANDLED\_ERR\_EID** #define CF\_CFDP\_FD\_UNHANDLED\_ERR\_EID 63  
CF Non-metadata File Directive PDU Received On Idle Transaction Event ID.

Type: ERROR

Cause:

File Directive PDU received without the metadata directive code on an idle transaction  
Definition at line 434 of file cf\_eventids.h.

**10.1.2.5 CF\_CFDP\_IDLE\_MD\_ERR\_EID** #define CF\_CFDP\_IDLE\_MD\_ERR\_EID 62  
CF Invalid Metadata PDU Received On Idle Transaction Event ID.

Type: ERROR

Cause:

Metadata PDU received for an idle transaction failed decoding  
Definition at line 423 of file cf\_eventids.h.

**10.1.2.6 CF\_CFDP\_INVALID\_DST\_ERR\_EID** #define CF\_CFDP\_INVALID\_DST\_ERR\_EID 61  
CF PDU Received With Invalid Destination Entity ID Event ID.

Type: ERROR

Cause:

PDU without a matching/existing transaction received with an entity ID that doesn't match the receiving channel's entity ID  
Definition at line 412 of file cf\_eventids.h.

**10.1.2.7 CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID** #define CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID 64  
CF Transmission Request Rejected Due To Max Commanded TX Reached Event ID.

Type: ERROR

Cause:

Command request to transmit a file received when channel is already handling the maximum number of concurrent command transmit transactions  
Definition at line 446 of file cf\_eventids.h.

**10.1.2.8 CF\_CFDP\_NO\_CHUNKLIST\_AVAIL\_EID** #define CF\_CFDP\_NO\_CHUNKLIST\_AVAIL\_EID 69  
CF No chunklist available.

Type: ERROR

Cause:

Engine has aborted a transaction due to lack of an available resource to track the chunks associated with the file.  
Definition at line 504 of file cf\_eventids.h.

**10.1.2.9 CF\_CFDP\_NO\_MSG\_ERR\_EID** #define CF\_CFDP\_NO\_MSG\_ERR\_EID 67  
CF No Message Buffer Available Event ID.

Type: ERROR

Cause:

Failure from SB allocate message buffer call when constructing PDU  
Definition at line 480 of file cf\_eventids.h.

**10.1.2.10 CF\_CFDP\_OPENDIR\_ERR\_EID** #define CF\_CFDP\_OPENDIR\_ERR\_EID 65  
CF Playback/Polling Directory Open Failed Event ID.

Type: ERROR

Cause:

Failure opening directory during playback or polling initialization  
Definition at line 457 of file cf\_eventids.h.

**10.1.2.11 CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_EID** #define CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_EID 73  
CF RX Transaction ACK Limit Reached Event ID.

Type: ERROR

Cause:

Condition that triggers an ACK occurred that would meet or exceed the ACK limit  
Definition at line 554 of file cf\_eventids.h.

**10.1.2.12 CF\_CFDP\_R\_CRC\_ERR\_EID** #define CF\_CFDP\_R\_CRC\_ERR\_EID 74  
CF RX Transaction CRC Mismatch Event ID.

Type: ERROR

Cause:

RX Transaction final CRC mismatch  
Definition at line 565 of file cf\_eventids.h.

**10.1.2.13 CF\_CFDP\_R\_CREAT\_ERR\_EID** #define CF\_CFDP\_R\_CREAT\_ERR\_EID 80  
CF RX Transaction File Create Failed Event ID.

Type: ERROR

Cause:

Failure in opencreate file call for an RX transaction  
Definition at line 632 of file cf\_eventids.h.

**10.1.2.14 CF\_CFDP\_R\_DC\_INV\_ERR\_EID** #define CF\_CFDP\_R\_DC\_INV\_ERR\_EID 87  
CF RX Invalid File Directive PDU Code Received Event ID.

Type: ERROR

Cause:

Unrecognized file directive PDU directive code received for a current transaction  
Definition at line 713 of file cf\_eventids.h.

**10.1.2.15 CF\_CFDP\_R\_EOF\_MD\_SIZE\_ERR\_EID** #define CF\_CFDP\_R\_EOF\_MD\_SIZE\_ERR\_EID 82  
CF RX Class 2 Metadata PDU Size Mismatch Event ID.

Type: ERROR

Cause:

Out-of-order RX Class 2 Metadata PDU received with file size that doesn't match already received EOF PDU file size  
Definition at line 655 of file cf\_eventids.h.

**10.1.2.16 CF\_CFDP\_R\_FILE\_RETAINED\_EID** #define CF\_CFDP\_R\_FILE\_RETAINED\_EID (84)  
CF File retained.

Type: INFORMATION

Cause:

Engine has fully retained the file after a successful transaction  
Definition at line 677 of file cf\_eventids.h.

**10.1.2.17 CF\_CFDP\_R\_INACT\_TIMER\_ERR\_EID** #define CF\_CFDP\_R\_INACT\_TIMER\_ERR\_EID 88  
CF RX Inactivity Timer Expired Event ID.

Type: ERROR

Cause:

Expiration of the RX inactivity timer  
Definition at line 724 of file cf\_eventids.h.

**10.1.2.18 CF\_CFDP\_R\_NAK\_LIMIT\_ERR\_EID** #define CF\_CFDP\_R\_NAK\_LIMIT\_ERR\_EID 72  
CF RX Transaction NAK Limit Reached Event ID.

Type: ERROR

Cause:

Condition that triggers a NAK occurred that would meet or exceed the NAK limit  
Definition at line 543 of file cf\_eventids.h.

**10.1.2.19 CF\_CFDP\_R\_NOT\_RETAINED\_EID** #define CF\_CFDP\_R\_NOT\_RETAINED\_EID (85)  
CF RX File not retained.

Type: INFORMATION

Cause:

Temporary file associated with a receive transaction was discarded without being retained. This may be due to an error in the transaction, failure to validate, or cancellation.  
Definition at line 690 of file cf\_eventids.h.

**10.1.2.20 CF\_CFDP\_R\_PDU\_EOF\_ERR\_EID** #define CF\_CFDP\_R\_PDU\_EOF\_ERR\_EID 79  
CF Invalid End-Of-File PDU Event ID.

Type: ERROR

Cause:

End-of-file PDU failed decoding  
Definition at line 621 of file cf\_eventids.h.

**10.1.2.21 CF\_CFDP\_R\_PDU\_FINACK\_ERR\_EID** #define CF\_CFDP\_R\_PDU\_FINACK\_ERR\_EID 81  
CF Class 2 RX Transaction Invalid FIN-ACK PDU Event ID.

Type: ERROR

Cause:

ACK PDU failed decoding during Class 2 RX Transaction  
Definition at line 643 of file cf\_eventids.h.

**10.1.2.22 CF\_CFDP\_R\_READ\_ERR\_EID** #define CF\_CFDP\_R\_READ\_ERR\_EID (86)  
CF Class 2 CRC Read From File Failed Event ID.

Type: ERROR

Cause:

Failure from file read call during RX Class 2 CRC calculation  
Definition at line 701 of file cf\_eventids.h.

**10.1.2.23 CF\_CFDP\_R\_RENAME\_ERR\_EID** #define CF\_CFDP\_R\_RENAME\_ERR\_EID 83  
CF RX Class 2 Metadata PDU File Rename Failed Event ID.

Type: ERROR

Cause:

Failure from file rename call after end of transaction  
Definition at line 666 of file cf\_eventids.h.

**10.1.2.24 CF\_CFDP\_R\_REQUEST\_MD\_INF\_EID** #define CF\_CFDP\_R\_REQUEST\_MD\_INF\_EID 70  
CF Requesting RX Metadata Event ID.

Type: INFORMATION

Cause:

RX transaction missing metadata which results in a NAK being sent to request a metadata PDU for the transaction  
Definition at line 520 of file cf\_eventids.h.

**10.1.2.25 CF\_CFDP\_R\_SEEK\_CRC\_ERR\_EID** #define CF\_CFDP\_R\_SEEK\_CRC\_ERR\_EID 76  
CF RX Class 2 CRC Seek Failed Event ID.

Type: ERROR

Cause:

Failure of lseek call when calculating CRC from the file at the end of a Class 2 RX transaction  
Definition at line 588 of file cf\_eventids.h.

**10.1.2.26 CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID** #define CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID 75  
CF RX File Data PDU Seek Failed Event ID.

Type: ERROR

Cause:

Failure of lseek call when processing out of order file data PDUs  
Definition at line 576 of file cf\_eventids.h.

**10.1.2.27 CF\_CFDP\_R\_SIZE\_MISMATCH\_ERR\_EID** #define CF\_CFDP\_R\_SIZE\_MISMATCH\_ERR\_EID 78  
CF RX End-Of-File PDU File Size Mismatch Event ID.

Type: ERROR

Cause:

End-of-file PDU file size does not match transaction expected file size  
Definition at line 610 of file cf\_eventids.h.

**10.1.2.28 CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID** #define CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID 71  
CF Creating Temp File For RX Transaction.

Type: INFORMATION

Cause:

RX transaction creation of a temporary filename to store the data  
Definition at line 532 of file cf\_eventids.h.

**10.1.2.29 CF\_CFDP\_R\_WRITE\_ERR\_EID** #define CF\_CFDP\_R\_WRITE\_ERR\_EID 77  
CF RX File Data PDU Write Failed Event ID.

Type: ERROR

Cause:

Failure of write to file call when processing file data PDUs  
Definition at line 599 of file cf\_eventids.h.

**10.1.2.30 CF\_CFDP\_RX\_DROPPED\_ERR\_EID** #define CF\_CFDP\_RX\_DROPPED\_ERR\_EID 60  
CF PDU Received Without Existing Transaction, Dropped Due To Max RX Reached Event ID.

Type: ERROR

Cause:

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

**10.1.2.31 CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID** #define CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID 106  
CF TX EOF PDU Send Limit Reached Event ID.

Type: ERROR

Cause:

Timed out the limit number of times waiting for an ACK PDU for the EOF PDU on a current transaction  
Definition at line 911 of file cf\_eventids.h.

**10.1.2.32 CF\_CFDP\_S\_ALREADY\_OPEN\_ERR\_EID** #define CF\_CFDP\_S\_ALREADY\_OPEN\_ERR\_EID 94  
CF TX Metadata PDU File Already Open Event ID.

Type: ERROR

Cause:

Failure to send metadata PDU due to file already being open  
Definition at line 783 of file cf\_eventids.h.

**10.1.2.33 CF\_CFDP\_S\_DC\_INV\_ERR\_EID** #define CF\_CFDP\_S\_DC\_INV\_ERR\_EID 104  
CF Invalid TX File Directive PDU Code Event ID.

Type: ERROR

Cause:

Unrecognized file directive PDU directive code received for a current transaction  
Definition at line 888 of file cf\_eventids.h.

**10.1.2.34 CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID** #define CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID 103  
CF TX Received Early FIN PDU Event ID.

Type: ERROR

Cause:

Early FIN PDU received prior to completion of a current transaction  
Definition at line 876 of file cf\_eventids.h.

**10.1.2.35 CF\_CFDP\_S\_FILE\_MOVED\_EID** #define CF\_CFDP\_S\_FILE\_MOVED\_EID (108)  
CF TX File Moved.

Type: INFORMATION

Cause:

Source File has been moved after a TX transaction This occurs when the move directory is configured.  
Definition at line 934 of file cf\_eventids.h.

**10.1.2.36 CF\_CFDP\_S\_FILE\_REMOVED\_EID** #define CF\_CFDP\_S\_FILE\_REMOVED\_EID (109)  
CF TX File Removed.

Type: INFORMATION

Cause:

Source File has been removed after a successful TX transaction where the "keep" flag was false and there is no move directory configured.  
Definition at line 947 of file cf\_eventids.h.

**10.1.2.37 CF\_CFDP\_S\_INACT\_TIMER\_ERR\_EID** #define CF\_CFDP\_S\_INACT\_TIMER\_ERR\_EID 107  
CF TX Inactivity Timer Expired Event ID.

Type: ERROR

Cause:

Send transaction activity timeout expired  
Definition at line 922 of file cf\_eventids.h.

**10.1.2.38 CF\_CFDP\_S\_INVALID\_SR\_ERR\_EID** #define CF\_CFDP\_S\_INVALID\_SR\_ERR\_EID 100  
CF TX Received NAK PDU Bad Segment Request Event ID.

Type: ERROR

Cause:

Bad segment request values in received NAK PDU relating to a current transaction  
Definition at line 841 of file cf\_eventids.h.

**10.1.2.39 CF\_CFDP\_S\_NON\_FD\_PDU\_ERR\_EID** #define CF\_CFDP\_S\_NON\_FD\_PDU\_ERR\_EID 105  
CF Received TX Non-File Directive PDU Event ID.

Type: ERROR

Cause:

Received a non-file directive PDU on a send transaction  
Definition at line 899 of file cf\_eventids.h.

**10.1.2.40 CF\_CFDP\_S\_OPEN\_ERR\_EID** #define CF\_CFDP\_S\_OPEN\_ERR\_EID 95  
CF TX Metadata PDU File Open Failed Event ID.

Type: ERROR

Cause:

Failure in file open call when preparing to send metadata PDU  
Definition at line 794 of file cf\_eventids.h.

**10.1.2.41 CF\_CFDP\_S\_PDU\_EOF\_ERR\_EID** #define CF\_CFDP\_S\_PDU\_EOF\_ERR\_EID 102  
CF TX Received EOF ACK PDU Invalid Event ID.

Type: ERROR

Cause:

Failure processing received ACK PDU relating to a current transaction  
Definition at line 865 of file cf\_eventids.h.

**10.1.2.42 CF\_CFDP\_S\_PDU\_NAK\_ERR\_EID** #define CF\_CFDP\_S\_PDU\_NAK\_ERR\_EID 101  
CF TX Received NAK PDU Invalid Event ID.

Type: ERROR

Cause:

Failure processing received NAK PDU relating to a current transaction  
Definition at line 853 of file cf\_eventids.h.

**10.1.2.43 CF\_CFDP\_S\_READ\_ERR\_EID** #define CF\_CFDP\_S\_READ\_ERR\_EID 92  
CF TX File Data PDU Read Failed Event ID.

Type: ERROR

Cause:

Failure of read file call when preparing to send file data PDU  
Definition at line 761 of file cf\_eventids.h.

**10.1.2.44 CF\_CFDP\_S\_SEEK\_BEG\_ERR\_EID** #define CF\_CFDP\_S\_SEEK\_BEG\_ERR\_EID 97  
CF TX Metadata PDU File Seek Beginning Failed Event ID.

Type: ERROR

Cause:

Failure in file lseek to beginning of file call when preparing to send metadata PDU  
Definition at line 818 of file cf\_eventids.h.

**10.1.2.45 CF\_CFDP\_S\_SEEK\_END\_ERR\_EID** #define CF\_CFDP\_S\_SEEK\_END\_ERR\_EID 96  
CF TX Metadata PDU File Seek End Failed Event ID.

Type: ERROR

Cause:

Failure in file lseek to end of file call when preparing to send metadata PDU  
Definition at line 806 of file cf\_eventids.h.

**10.1.2.46 CF\_CFDP\_S\_SEEK\_FD\_ERR\_EID** #define CF\_CFDP\_S\_SEEK\_FD\_ERR\_EID 91  
CF TX File Data PDU Seek Failed Event ID.

Type: ERROR

Cause:

Failure of lseek call when preparing to send file data PDU  
Definition at line 750 of file cf\_eventids.h.

**10.1.2.47 CF\_CFDP\_S\_SEND\_FD\_ERR\_EID** #define CF\_CFDP\_S\_SEND\_FD\_ERR\_EID 93  
CF TX File Data PDU Send Failed Event ID.

Type: ERROR

Cause:

Failure to send the file data PDU  
Definition at line 772 of file cf\_eventids.h.

**10.1.2.48 CF\_CFDP\_S\_SEND\_MD\_ERR\_EID** #define CF\_CFDP\_S\_SEND\_MD\_ERR\_EID 98  
CF TX Metadata PDU Send Failed Event ID.

Type: ERROR

Cause:

Failure to send the metadata PDU  
Definition at line 829 of file cf\_eventids.h.

**10.1.2.49 CF\_CFDP\_S\_START\_SEND\_INF\_EID** #define CF\_CFDP\_S\_START\_SEND\_INF\_EID 90  
CF TX Initiated Event ID.

Type: INFORMATION

Cause:

File TX transaction initiated  
Definition at line 739 of file cf\_eventids.h.

**10.1.2.50 CF\_CMD\_ABANDON\_CHAN\_ERR\_EID** #define CF\_CMD\_ABANDON\_CHAN\_ERR\_EID 156  
CF Abandon Command No Matching Transaction Event ID.

Type: ERROR

Cause:

Abandon command received without a matching transaction  
Definition at line 1468 of file cf\_eventids.h.

**10.1.2.51 CF\_CMD\_ABANDON\_INF\_EID** #define CF\_CMD\_ABANDON\_INF\_EID 123  
CF Abandon Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of abandon command  
Definition at line 1105 of file cf\_eventids.h.

**10.1.2.52 CF\_CMD\_BAD\_PARAM\_ERR\_EID** #define CF\_CMD\_BAD\_PARAM\_ERR\_EID 154  
CF Playback Dir Or TX File Command Bad Parameter Event ID.

Type: ERROR

Cause:

Bad parameter received in playback directory or transfer file command  
Definition at line 1446 of file cf\_eventids.h.

**10.1.2.53 CF\_CMD\_CANCEL\_CHAN\_ERR\_EID** #define CF\_CMD\_CANCEL\_CHAN\_ERR\_EID 155  
CF Cancel Command No Matching Transaction Event ID.

Type: ERROR

Cause:

Cancel command received without a matching transaction  
Definition at line 1457 of file cf\_eventids.h.

**10.1.2.54 CF\_CMD\_CANCEL\_INF\_EID** #define CF\_CMD\_CANCEL\_INF\_EID 122  
CF Cancel Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of cancel command  
Definition at line 1094 of file cf\_eventids.h.

**10.1.2.55 CF\_CMD\_CHAN\_PARAM\_ERR\_EID** #define CF\_CMD\_CHAN\_PARAM\_ERR\_EID 130  
CF Command Channel Invalid Event ID.

Type: ERROR

Cause:

Command received with channel parameter out of range  
Definition at line 1182 of file cf\_eventids.h.

**10.1.2.56 CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID** #define CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID 162  
CF Disable Dequeue Command Failed Event ID.

Type: ERROR

Cause:

Disable dequeue command was unsuccessful  
Definition at line 1534 of file cf\_eventids.h.

**10.1.2.57 CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID** #define CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID 125  
CF Disable Dequeue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of disable dequeue command  
Definition at line 1127 of file cf\_eventids.h.

**10.1.2.58 CF\_CMD\_DISABLE\_ENGINE\_INF\_EID** #define CF\_CMD\_DISABLE\_ENGINE\_INF\_EID 117  
CF Disable Engine Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of disable engine command  
Definition at line 1039 of file cf\_eventids.h.

**10.1.2.59 CF\_CMD\_DISABLE\_POLLDIR\_ERR\_EID** #define CF\_CMD\_DISABLE\_POLLDIR\_ERR\_EID 164  
CF Disable Polldir Command Failed Event ID.

Type: ERROR

Cause:

Disable polldir command was unsuccessful  
Definition at line 1556 of file cf\_eventids.h.

**10.1.2.60 CF\_CMD\_DISABLE\_POLLDIR\_INF\_EID** #define CF\_CMD\_DISABLE\_POLLDIR\_INF\_EID 127  
CF Disable Polldir Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of disable polldir command  
Definition at line 1149 of file cf\_eventids.h.

**10.1.2.61 CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID** #define CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID 161  
CF Enable Dequeue Command Failed Event ID.

Type: ERROR

Cause:

Enable Dequeue command was unsuccessful  
Definition at line 1523 of file cf\_eventids.h.

**10.1.2.62 CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID** #define CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID 124  
CF Enable Dequeue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of enable dequeue command  
Definition at line 1116 of file cf\_eventids.h.

**10.1.2.63 CF\_CMD\_ENABLE\_ENGINE\_ERR\_EID** #define CF\_CMD\_ENABLE\_ENGINE\_ERR\_EID 148  
CF Enable Engine Command Failed Event ID.

Type: ERROR

Cause:

Failed to initialize engine when processing engine enable command  
Definition at line 1380 of file cf\_eventids.h.

**10.1.2.64 CF\_CMD\_ENABLE\_ENGINE\_INF\_EID** #define CF\_CMD\_ENABLE\_ENGINE\_INF\_EID 116  
CF Enable Engine Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of enable engine command  
Definition at line 1028 of file cf\_eventids.h.

**10.1.2.65 CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID** #define CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID 163  
CF Enable Polldir Command Failed Event ID.

Type: ERROR

Cause:

Enable polldir command was unsuccessful  
Definition at line 1545 of file cf\_eventids.h.

**10.1.2.66 CF\_CMD\_ENABLE\_POLLDIR\_INF\_EID** #define CF\_CMD\_ENABLE\_POLLDIR\_INF\_EID 126  
CF Enable Polldir Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of enable polldir command  
Definition at line 1138 of file cf\_eventids.h.

**10.1.2.67 CF\_CMD\_ENG\_ALREADY\_DIS\_INF\_EID** #define CF\_CMD\_ENG\_ALREADY\_DIS\_INF\_EID 150  
CF Disable Engine Command Engine Already Disabled Event ID.

Type: INFORMATION

Cause:

Disable engine command received while engine is already disabled  
Definition at line 1402 of file cf\_eventids.h.

**10.1.2.68 CF\_CMD\_ENG\_ALREADY\_ENA\_INF\_EID** #define CF\_CMD\_ENG\_ALREADY\_ENA\_INF\_EID 149  
CF Enable Engine Command Engine Already Enabled Event ID.

Type: INFORMATION

Cause:

Enable engine command received while engine is already enabled  
Definition at line 1391 of file cf\_eventids.h.

**10.1.2.69 CF\_CMD\_FREEZE\_ERR\_EID** #define CF\_CMD\_FREEZE\_ERR\_EID 159  
CF Freeze Command Failed Event ID.

Type: ERROR

Cause:

Freeze command was unsuccessful  
Definition at line 1501 of file cf\_eventids.h.

**10.1.2.70 CF\_CMD\_FREEZE\_INF\_EID** #define CF\_CMD\_FREEZE\_INF\_EID 120  
CF Freeze Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of freeze command  
Definition at line 1072 of file cf\_eventids.h.

**10.1.2.71 CF\_CMD\_GETSET1\_INF\_EID** #define CF\_CMD\_GETSET1\_INF\_EID 112  
CF Set Parameter Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of set parameter command  
Definition at line 984 of file cf\_eventids.h.

**10.1.2.72 CF\_CMD\_GETSET2\_INF\_EID** #define CF\_CMD\_GETSET2\_INF\_EID 113  
CF Get Parameter Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of get parameter command  
Definition at line 995 of file cf\_eventids.h.

**10.1.2.73 CF\_CMD\_GETSET\_CHAN\_ERR\_EID** #define CF\_CMD\_GETSET\_CHAN\_ERR\_EID 147  
CF Set/Get Parameter Command Invalid Channel Event ID.

Type: ERROR

Cause:

Invalid channel value received in set or get parameter command  
Definition at line 1369 of file cf\_eventids.h.

**10.1.2.74 CF\_CMD\_GETSET\_PARAM\_ERR\_EID** #define CF\_CMD\_GETSET\_PARAM\_ERR\_EID 146  
CF Set/Get Parameter Command Invalid Parameter ID Event ID.

Type: ERROR

Cause:

Invalid parameter id value received in set or get parameter command  
Definition at line 1358 of file cf\_eventids.h.

**10.1.2.75 CF\_CMD\_GETSET\_VALIDATE\_ERR\_EID** #define CF\_CMD\_GETSET\_VALIDATE\_ERR\_EID 145  
CF Set Parameter Command Parameter Validation Failed Event ID.

Type: ERROR

Cause:

Parameter validation failed during processing of set parameter command  
Definition at line 1347 of file cf\_eventids.h.

**10.1.2.76 CF\_CMD\_LEN\_ERR\_EID** #define CF\_CMD\_LEN\_ERR\_EID 151  
CF Command Length Verification Failed Event ID.

Type: ERROR

Cause:

Received command length verification failure  
Definition at line 1413 of file cf\_eventids.h.

**10.1.2.77 CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID** #define CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID 158  
CF Playback Directory Command Failed Event ID.

Type: ERROR

Cause:

Playback directory command was unsuccessful  
Definition at line 1490 of file cf\_eventids.h.

**10.1.2.78 CF\_CMD\_PLAYBACK\_DIR\_INF\_EID** #define CF\_CMD\_PLAYBACK\_DIR\_INF\_EID 119  
CF Playback Directory Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of playback directory command  
Definition at line 1061 of file cf\_eventids.h.

**10.1.2.79 CF\_CMD\_POLLDIR\_INVALID\_ERR\_EID** #define CF\_CMD\_POLLDIR\_INVALID\_ERR\_EID 135  
CF Enable/Disable Polling Directory Command Invalid Polling Directory Index Event ID.

Type: ERROR

Cause:

Enable/disable polling directory command received with invalid polling directory index  
Definition at line 1237 of file cf\_eventids.h.

**10.1.2.80 CF\_CMD\_PURGE\_ARG\_ERR\_EID** #define CF\_CMD\_PURGE\_ARG\_ERR\_EID 136  
CF Purge Queue Command Invalid Argument Event ID.

Type: ERROR

Cause:

Purge Queue command received with invalid queue argument  
Definition at line 1248 of file cf\_eventids.h.

**10.1.2.81 CF\_CMD\_PURGE\_QUEUE\_ERR\_EID** #define CF\_CMD\_PURGE\_QUEUE\_ERR\_EID 165  
CF Purge Queue Command Failed Event ID.

Type: ERROR

Cause:

Purge queue command was unsuccessful  
Definition at line 1567 of file cf\_eventids.h.

**10.1.2.82 CF\_CMD\_PURGE\_QUEUE\_INF\_EID** #define CF\_CMD\_PURGE\_QUEUE\_INF\_EID 128  
CF Purge Queue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of purge queue command  
Definition at line 1160 of file cf\_eventids.h.

**10.1.2.83 CF\_CMD\_RESET\_INVALID\_ERR\_EID** #define CF\_CMD\_RESET\_INVALID\_ERR\_EID 129  
CF Reset Counters Command Invalid Event ID.

Type: ERROR

Cause:

Reset counters command received with invalid parameter  
Definition at line 1171 of file cf\_eventids.h.

**10.1.2.84 CF\_CMD\_SUSPRES\_CHAN\_ERR\_EID** #define CF\_CMD\_SUSPRES\_CHAN\_ERR\_EID 134  
CF Suspend/Resume Command No Matching Transaction Event ID.

Type: ERROR

Cause:

Suspend/resume command received without a matching transaction  
Definition at line 1226 of file cf\_eventids.h.

**10.1.2.85 CF\_CMD\_SUSPRES\_INF\_EID** #define CF\_CMD\_SUSPRES\_INF\_EID 114  
CF Suspend/Resume Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of suspend/resume command  
Definition at line 1006 of file cf\_eventids.h.

**10.1.2.86 CF\_CMD\_SUSPRES\_SAME\_INF\_EID** #define CF\_CMD\_SUSPRES\_SAME\_INF\_EID 133  
CF Suspend/Resume Command For Single Transaction State Unchanged Event ID.

Type: INFORMATION

Cause:

Suspend/resume command received affecting single transaction already set to that state  
Definition at line 1215 of file cf\_eventids.h.

**10.1.2.87 CF\_CMD\_THAW\_ERR\_EID** #define CF\_CMD\_THAW\_ERR\_EID 160  
CF Thaw Command Failed Event ID.

Type: ERROR

Cause:

Thaw command was unsuccessful  
Definition at line 1512 of file cf\_eventids.h.

**10.1.2.88 CF\_CMD\_THAW\_INF\_EID** #define CF\_CMD\_THAW\_INF\_EID 121  
CF Thaw Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of thaw command  
Definition at line 1083 of file cf\_eventids.h.

**10.1.2.89 CF\_CMD\_TRANS\_NOT\_FOUND\_ERR\_EID** #define CF\_CMD\_TRANS\_NOT\_FOUND\_ERR\_EID 131  
CF Command Transaction Invalid Event ID.

Type: ERROR

Cause:

Command received without a matching transaction  
Definition at line 1193 of file cf\_eventids.h.

**10.1.2.90 CF\_CMD\_TSN\_CHAN\_INVALID\_ERR\_EID** #define CF\_CMD\_TSN\_CHAN\_INVALID\_ERR\_EID 132  
CF Command All Transaction Channel Invalid Event ID.

Type: ERROR

Cause:

Command received to act on all transactions with invalid channel  
Definition at line 1204 of file cf\_eventids.h.

**10.1.2.91 CF\_CMD\_TX\_FILE\_ERR\_EID** #define CF\_CMD\_TX\_FILE\_ERR\_EID 157  
CF Transfer File Command Failed Event ID.

Type: ERROR

Cause:

Transfer file command was unsuccessful  
Definition at line 1479 of file cf\_eventids.h.

**10.1.2.92 CF\_CMD\_TX\_FILE\_INF\_EID** #define CF\_CMD\_TX\_FILE\_INF\_EID 118  
CF Transfer File Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of transfer file command  
Definition at line 1050 of file cf\_eventids.h.

**10.1.2.93 CF\_CMD\_WHIST\_WRITE\_ERR\_EID** #define CF\_CMD\_WHIST\_WRITE\_ERR\_EID 153  
CF Write Entry To File Failed Event ID.

Type: ERROR

Cause:

Write entry to file did not match expected length  
Definition at line 1435 of file cf\_eventids.h.

**10.1.2.94 CF\_CMD\_WQ\_ARGS\_ERR\_EID** #define CF\_CMD\_WQ\_ARGS\_ERR\_EID 138  
CF Write Queue Command Invalid Queue Event ID.

Type: ERROR

Cause:

Write Queue command received with invalid queue selection arguments  
Definition at line 1270 of file cf\_eventids.h.

**10.1.2.95 CF\_CMD\_WQ\_CHAN\_ERR\_EID** #define CF\_CMD\_WQ\_CHAN\_ERR\_EID 137  
CF Write Queue Command Invalid Channel Event ID.

Type: ERROR

Cause:

Write Queue command received with invalid channel argument  
Definition at line 1259 of file cf\_eventids.h.

**10.1.2.96 CF\_CMD\_WQ\_INF\_EID** #define CF\_CMD\_WQ\_INF\_EID 115  
CF Write Queue Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of write queue command  
Definition at line 1017 of file cf\_eventids.h.

**10.1.2.97 CF\_CMD\_WQ\_OPEN\_ERR\_EID** #define CF\_CMD\_WQ\_OPEN\_ERR\_EID 139  
CF Write Queue Command File Open Failed Event ID.

Type: ERROR

Cause:

Failure of open file call during processing of write queue command  
Definition at line 1281 of file cf\_eventids.h.

**10.1.2.98 CF\_CMD\_WQ\_WRITEHIST\_RX\_ERR\_EID** #define CF\_CMD\_WQ\_WRITEHIST\_RX\_ERR\_EID 141  
CF Write Queue Command RX History File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for RX history during processing of write queue command  
Definition at line 1303 of file cf\_eventids.h.

**10.1.2.99 CF\_CMD\_WQ\_WRITEHIST\_TX\_ERR\_EID** #define CF\_CMD\_WQ\_WRITEHIST\_TX\_ERR\_EID 144  
CF Write Queue Command TX History File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for TX history during processing of write queue command  
Definition at line 1336 of file cf\_eventids.h.

**10.1.2.100 CF\_CMD\_WQ\_WRITEQ\_PEND\_ERR\_EID** #define CF\_CMD\_WQ\_WRITEQ\_PEND\_ERR\_EID 143  
CF Write Queue Command TX Pending File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for TX pending transactions during processing of write queue command  
Definition at line 1325 of file cf\_eventids.h.

**10.1.2.101 CF\_CMD\_WQ\_WRITEQ\_RX\_ERR\_EID** #define CF\_CMD\_WQ\_WRITEQ\_RX\_ERR\_EID 140  
CF Write Queue Command RX Active File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for RX active transactions during processing of write queue command  
Definition at line 1292 of file cf\_eventids.h.

**10.1.2.102 CF\_CMD\_WQ\_WRITEQ\_TX\_ERR\_EID** #define CF\_CMD\_WQ\_WRITEQ\_TX\_ERR\_EID 142  
CF Write Queue Command TX Active File Write Failed Event ID.

Type: ERROR

Cause:

Failure of file write call for TX active transactions during processing of write queue command  
Definition at line 1314 of file cf\_eventids.h.

**10.1.2.103 CF\_CR\_CHANNEL\_PIPE\_ERR\_EID** #define CF\_CR\_CHANNEL\_PIPE\_ERR\_EID 31  
CF Channel Create Pipe Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.104 CF\_CR\_PIPE\_ERR\_EID** #define CF\_CR\_PIPE\_ERR\_EID 36  
CF Create SB Command Pipe at Initialization Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.105 CF\_EID\_INF\_CFDP\_BUF\_EXCEED** #define CF\_EID\_INF\_CFDP\_BUF\_EXCEED 166  
CF Move Path Length Verification Too Long Event ID.

Type: INFORMATION

Cause:

Combined move filename length exceeds buffer size  
Definition at line 1578 of file cf\_eventids.h.

**10.1.2.106 CF\_INIT\_CRC\_ALIGN\_ERR\_EID** #define CF\_INIT\_CRC\_ALIGN\_ERR\_EID 34  
CF CRC Bytes Per Wakeup Config Table Validation Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.107 CF\_INIT\_INF\_EID** #define CF\_INIT\_INF\_EID 20  
CF Initialization Event ID.

Type: INFORMATION

Cause:

Successful completion of application initialization  
Definition at line 47 of file cf\_eventids.h.

**10.1.2.108 CF\_INIT\_MSG\_RECV\_ERR\_EID** #define CF\_INIT\_MSG\_RECV\_ERR\_EID 29  
CF SB Receive Buffer Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.109 CF\_INIT\_OUTGOING\_SIZE\_ERR\_EID** #define CF\_INIT\_OUTGOING\_SIZE\_ERR\_EID 35  
CF Outgoing Chunk Size Config Table Validation Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.110 CF\_INIT\_SEM\_ERR\_EID** #define CF\_INIT\_SEM\_ERR\_EID 30  
CF Channel Semaphore Initialization Failed Event ID.

Type: ERROR

Cause:

Failure from get semaphore by name call during engine channel initialization, semaphore needs to exist before engine is initialized.

Definition at line 158 of file cf\_eventids.h.

**10.1.2.111 CF\_INIT\_SUB\_ERR\_EID** #define CF\_INIT\_SUB\_ERR\_EID 32  
CF Channel Message Subscription Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.112 CF\_INIT\_TBL\_CHECK\_GA\_ERR\_EID** #define CF\_INIT\_TBL\_CHECK\_GA\_ERR\_EID 23  
CF Check Table Get Address Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.113 CF\_INIT\_TBL\_CHECK\_MAN\_ERR\_EID** #define CF\_INIT\_TBL\_CHECK\_MAN\_ERR\_EID 22  
CF Check Table Manage Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.114 CF\_INIT\_TBL\_CHECK\_REL\_ERR\_EID** #define CF\_INIT\_TBL\_CHECK\_REL\_ERR\_EID 21  
CF Check Table Release Address Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.115 CF\_INIT\_TBL\_GETADDR\_ERR\_EID** #define CF\_INIT\_TBL\_GETADDR\_ERR\_EID 27  
CF Table Get Address At Initialization Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.116 CF\_INIT\_TBL\_LOAD\_ERR\_EID** #define CF\_INIT\_TBL\_LOAD\_ERR\_EID 25  
CF Table Load At Initialization Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.117 CF\_INIT\_TBL\_MANAGE\_ERR\_EID** #define CF\_INIT\_TBL\_MANAGE\_ERR\_EID 26  
CF Table Manage At Initialization Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.118 CF\_INIT\_TBL\_REG\_ERR\_EID** #define CF\_INIT\_TBL\_REG\_ERR\_EID 24  
CF Table Registration At Initialization Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.119 CF\_INIT\_TPS\_ERR\_EID** #define CF\_INIT\_TPS\_ERR\_EID 33  
CF Ticks Per Second Config Table Validation Failed Event ID.

Type: ERROR

Cause:

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

**10.1.2.120 CF\_MID\_ERR\_EID** #define CF\_MID\_ERR\_EID 28  
CF Message ID Invalid Event ID.

Type: ERROR

Cause:

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

**10.1.2.121 CF\_NOOP\_INF\_EID** #define CF\_NOOP\_INF\_EID 110  
CF NOOP Command Received Event ID.

Type: INFORMATION

Cause:

Receipt of NOOP command  
Definition at line 962 of file cf\_eventids.h.

**10.1.2.122 CF\_PDU\_ACK\_SHORT\_ERR\_EID** #define CF\_PDU\_ACK\_SHORT\_ERR\_EID 48  
CF Acknowledgment PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing acknowledgment PDU  
Definition at line 316 of file cf\_eventids.h.

**10.1.2.123 CF\_PDU\_EOF\_SHORT\_ERR\_EID** #define CF\_PDU\_EOF\_SHORT\_ERR\_EID 47  
CF End-Of-File PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing end-of-file PDU  
Definition at line 305 of file cf\_eventids.h.

**10.1.2.124 CF\_PDU\_FD\_SHORT\_ERR\_EID** #define CF\_PDU\_FD\_SHORT\_ERR\_EID 46  
CF File Data PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing file data PDU  
Definition at line 294 of file cf\_eventids.h.

**10.1.2.125 CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID** #define CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID 54  
CF File Data PDU Unsupported Option Event ID.

Type: ERROR

Cause:

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

**10.1.2.126 CF\_PDU\_FIN\_SHORT\_ERR\_EID** #define CF\_PDU\_FIN\_SHORT\_ERR\_EID 49  
CF Finished PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing finished PDU  
Definition at line 327 of file cf\_eventids.h.

**10.1.2.127 CF\_PDU\_INVALID\_DST\_LEN\_ERR\_EID** #define CF\_PDU\_INVALID\_DST\_LEN\_ERR\_EID 45  
CF Metadata PDU Destination Filename Length Invalid Event ID.

Type: ERROR

Cause:

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

**10.1.2.128 CF\_PDU\_INVALID\_SRC\_LEN\_ERR\_EID** #define CF\_PDU\_INVALID\_SRC\_LEN\_ERR\_EID 44  
CF Metadata PDU Source Filename Length Invalid Event ID.

Type: ERROR

Cause:

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

**10.1.2.129 CF\_PDU\_LARGE\_FILE\_ERR\_EID** #define CF\_PDU\_LARGE\_FILE\_ERR\_EID 55  
CF PDU Header Large File Flag Set Event ID.

Type: ERROR

Cause:

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

**10.1.2.130 CF\_PDU\_MD\_RECVD\_INF\_EID** #define CF\_PDU\_MD\_RECVD\_INF\_EID 40  
CF Metadata PDU Received Event ID.

Type: INFORMATION

Cause:

Successful processing of metadata PDU  
Definition at line 239 of file cf\_eventids.h.

**10.1.2.131 CF\_PDU\_MD\_SHORT\_ERR\_EID** #define CF\_PDU\_MD\_SHORT\_ERR\_EID 43  
CF Metadata PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing metadata PDU  
Definition at line 261 of file cf\_eventids.h.

**10.1.2.132 CF\_PDU\_NAK\_SHORT\_ERR\_EID** #define CF\_PDU\_NAK\_SHORT\_ERR\_EID 50  
CF Negative Acknowledgment PDU Too Short Event ID.

Type: ERROR

Cause:

Failure processing negative acknowledgment PDU  
Definition at line 338 of file cf\_eventids.h.

**10.1.2.133 CF\_PDU\_SHORT\_HEADER\_ERR\_EID** #define CF\_PDU\_SHORT\_HEADER\_ERR\_EID 41  
CF PDU Header Too Short Event ID.

Type: ERROR

Cause:

Failure processing PDU header  
Definition at line 250 of file cf\_eventids.h.

**10.1.2.134 CF\_PDU\_TRUNCATION\_ERR\_EID** #define CF\_PDU\_TRUNCATION\_ERR\_EID 56  
CF PDU Header Field Truncation.

Type: ERROR

Cause:

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

**10.1.2.135 CF\_RESET\_FREED\_XACT\_DBG\_EID** #define CF\_RESET\_FREED\_XACT\_DBG\_EID 59  
Attempt to reset a transaction that has already been freed.

Type: DEBUG

Cause:

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

**10.1.2.136 CF\_RESET\_INF\_EID** #define CF\_RESET\_INF\_EID 111  
CF Reset Counters Command Received Event ID.

Type: INFORMATION

Cause:

Receipt and successful processing of reset counters command  
Definition at line 973 of file cf\_eventids.h.

## 10.2 CFS CFDP Command Codes

### Macros

- `#define CF_NOOP_CC CF_CCVAL(NOOP)`  
*No-op.*
- `#define CF_RESET_CC CF_CCVAL(RESET_COUNTERS)`  
*Reset counters.*
- `#define CF_TX_FILE_CC CF_CCVAL(TX_FILE)`  
*Transmit file.*
- `#define CF_PLAYBACK_DIR_CC CF_CCVAL(PLAYBACK_DIR)`  
*Playback a directory.*
- `#define CF_FREEZE_CC CF_CCVAL(FREEZE)`  
*Freeze a channel.*
- `#define CF_THAW_CC CF_CCVAL(THAW)`  
*Thaw a channel.*
- `#define CF_SUSPEND_CC CF_CCVAL(SUSPEND)`  
*Suspend a transaction.*
- `#define CF_RESUME_CC CF_CCVAL(RESUME)`  
*Resume a transaction.*
- `#define CF_CANCEL_CC CF_CCVAL(CANCEL)`  
*Cancel a transaction.*
- `#define CF_ABANDON_CC CF_CCVAL(ABANDON)`  
*Abandon a transaction.*
- `#define CF_SET_PARAM_CC CF_CCVAL(SET_PARAM)`  
*Set parameter.*
- `#define CF_GET_PARAM_CC CF_CCVAL(GET_PARAM)`  
*Get parameter.*
- `#define CF_WRITE_QUEUE_CC CF_CCVAL(WRITE_QUEUE)`  
*Write queue.*
- `#define CF_ENABLE_DEQUEUE_CC CF_CCVAL(ENABLE_DEQUEUE)`  
*Enable dequeue.*
- `#define CF_DISABLE_DEQUEUE_CC CF_CCVAL(DISABLE_DEQUEUE)`  
*Disable dequeue.*
- `#define CF_ENABLE_DIR_POLLING_CC CF_CCVAL(ENABLE_DIR_POLLING)`  
*Enable directory polling.*
- `#define CF_DISABLE_DIR_POLLING_CC CF_CCVAL(DISABLE_DIR_POLLING)`  
*Disable directory polling.*
- `#define CF_PURGE_QUEUE_CC CF_CCVAL(PURGE_QUEUE)`  
*Purge queue.*
- `#define CF_ENABLE_ENGINE_CC CF_CCVAL(ENABLE_ENGINE)`  
*Enable engine.*
- `#define CF_DISABLE_ENGINE_CC CF_CCVAL(DISABLE_ENGINE)`  
*Disable engine.*
- `#define CF_NUM_COMMANDS 24`  
*Command code limit used for validity check and array sizing.*

### 10.2.1 Detailed Description

### 10.2.2 Macro Definition Documentation

#### 10.2.2.1 CF\_ABANDON\_CC `#define CF_ABANDON_CC CF_CCVAL(ABANDON)`

Abandon a transaction.

##### Description

Abandon transaction processing with an immediate reset (no close out attempted) for a single transaction, all channels and transactions, or all transactions on a specific channel.

##### Command Structure

[CF\\_Transaction\\_Payload\\_t](#)

##### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_ABANDON\\_INF\\_EID](#)

##### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Transaction not found using compound key, [CF\\_CMD\\_TRANS\\_NOT\\_FOUND\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_TSN\\_CHAN\\_INVALID\\_ERR\\_EID](#)
- No matching transaction, [CF\\_CMD\\_ABANDON\\_CHAN\\_ERR\\_EID](#)

##### Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.terr](#) will increment

##### Criticality

None

##### See also

[CF\\_SUSPEND\\_CC](#), [CF\\_RESUME\\_CC](#), [CF\\_CANCEL\\_CC](#)

Definition at line 373 of file cf\_fcncodes.h.

#### 10.2.2.2 CF\_CANCEL\_CC `#define CF_CANCEL_CC CF_CCVAL(CANCEL)`

Cancel a transaction.

##### Description

Cancel transaction processing by taking steps to close out cleanly (based on transaction type and direction) for a single transaction, all channels and transactions, or all transactions on a specific channel.

**Command Structure**

[CF\\_Transaction\\_Payload\\_t](#)

**Command Verification**

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_CANCEL\\_INF\\_EID](#)

**Error Conditions**

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Transaction not found using compound key, [CF\\_CMD\\_TRANS\\_NOT\\_FOUND\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_TSN\\_CHAN\\_INVALID\\_ERR\\_EID](#)
- No matching transaction, [CF\\_CMD\\_CANCEL\\_CHAN\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.err](#) will increment

**Criticality**

None

**See also**

[CF\\_SUSPEND\\_CC](#), [CF\\_RESUME\\_CC](#), [CF\\_ABANDON\\_CC](#)

Definition at line 339 of file cf\_fcncodes.h.

**10.2.2.3 CF\_DISABLE\_DEQUEUE\_CC** #define CF\_DISABLE\_DEQUEUE\_CC [CF\\_CCVAL](#)(DISABLE\_DEQUEUE)  
Disable dequeue.

**Description**

Disables the sending of file data PDUs.

**Command Structure**

[CF\\_UnionArgs\\_Payload\\_t](#) where byte[0] specifies the channel number or all channels

- 255 = all channels
- else = single channel

**Command Verification**

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_DISABLE\\_DEQUEUE\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_CHAN\\_PARAM\\_ERR\\_EID](#)
- Disable dequeue failed, [CF\\_CMD\\_DISABLE\\_DEQUEUE\\_INF\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.err](#) will increment

### Criticality

None

### See also

[CF\\_ENABLE\\_DEQUEUE\\_CC](#)

Definition at line 538 of file cf\_fcncodes.h.

**10.2.2.4 CF\_DISABLE\_DIR\_POLLING\_CC** #define CF\_DISABLE\_DIR\_POLLING\_CC [CF\\_CCVAL](#)(DISABLE\_DIR\_←  
POLLOING)

Disable directory polling.

### Description

Disable the processing of polling directories

### Command Structure

[CF\\_UnionArgs\\_Payload\\_t](#)

byte[0] specifies the channel number or all channels

- 255 = all channels
- else = single channel

byte[1] specifies the polling directory index

- 255 = all polling directories
- else = single polling directory index

### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_DISABLE\\_POLLDIR\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_CHAN\\_PARAM\\_ERR\\_EID](#)
- Invalid polling directory index, [CF\\_CMD\\_POLLDIR\\_INVALID\\_ERR\\_EID](#)
- Disable directory polling failed, [CF\\_CMD\\_DISABLE\\_POLLDIR\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_ENABLE\\_DIR\\_POLLING\\_CC](#)

Definition at line 618 of file cf\_fcncodes.h.

### **10.2.2.5 CF\_DISABLE\_ENGINE\_CC #define CF\_DISABLE\_ENGINE\_CC CF\_CCVAL(DISABLE\_ENGINE)**

Disable engine.

#### Description

Disable engine processing. Note configuration table updates can be performed while the engine is disabled, and when the engine is re-enabled the new configuration will take effect.

#### Command Structure

No Payload / Arguments

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_DISABLE\\_ENGINE\\_INF\\_EID](#)
  - [CF\\_CMD\\_ENG\\_ALREADY\\_DIS\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_DISABLE\\_ENGINE\\_CC](#)

Definition at line 723 of file cf\_fcncodes.h.

**10.2.2.6 CF\_ENABLE\_DEQUEUE\_CC** #define CF\_ENABLE\_DEQUEUE\_CC CF\_CCVAL(ENABLE\_DEQUEUE)  
Enable dequeue.

#### Description

Enables the sending of file data PDUs.

#### Command Structure

[CF\\_UnionArgs\\_Payload\\_t](#) where byte[0] specifies the channel number or all channels

- 255 = all channels
- else = single channel

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_ENABLE\\_DEQUEUE\\_INF\\_EID](#)

#### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_CHAN\\_PARAM\\_ERR\\_EID](#)
- Enable dequeue failed, [CF\\_CMD\\_ENABLE\\_DEQUEUE\\_ERR\\_EID](#)

#### Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.err](#) will increment

#### Criticality

None

#### See also

[CF\\_DISABLE\\_DEQUEUE\\_CC](#)

Definition at line 505 of file cf\_fcncodes.h.

**10.2.2.7 CF\_ENABLE\_DIR\_POLLING\_CC** #define CF\_ENABLE\_DIR\_POLLING\_CC CF\_CCVAL(ENABLE\_DIR\_POLLING)

Enable directory polling.

#### Description

Enables the processing of polling directories

#### Command Structure

##### [CF\\_UnionArgs\\_Payload\\_t](#)

byte[0] specifies the channel number or all channels

- 255 = all channels
- else = single channel

byte[1] specifies the polling directory index

- 255 = all polling directories
- else = single polling directory index

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_ENABLE\\_POLLDIR\\_INF\\_EID](#)

#### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_CHAN\\_PARAM\\_ERR\\_EID](#)
- Invalid polling directory index, [CF\\_CMD\\_POLLDIR\\_INVALID\\_ERR\\_EID](#)
- Enable directory polling failed, [CF\\_CMD\\_ENABLE\\_POLLDIR\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.err](#) will increment

#### Criticality

None

#### See also

##### [CF\\_DISABLE\\_DIR\\_POLLING\\_CC](#)

Definition at line 578 of file cf\_fcncodes.h.

**10.2.2.8 CF\_ENABLE\_ENGINE\_CC** #define CF\_ENABLE\_ENGINE\_CC [CF\\_CCVAL](#)(ENABLE\_ENGINE)  
Enable engine.

#### Description

Reinitialize engine and enable processing. Note configuration table updates are not processed while the engine is enabled.

#### Command Structure

No Payload / Arguments

### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_ENABLE\\_ENGINE\\_INF\\_EID](#)
- [CF\\_CMD\\_ENG\\_ALREADY\\_ENA\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Engine initialization failed, [CF\\_CMD\\_ENABLE\\_ENGINE\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_DISABLE\\_ENGINE\\_CC](#)

Definition at line 691 of file cf\_fcncodes.h.

### **10.2.2.9 CF\_FREEZE\_CC** #define CF\_FREEZE\_CC [CF\\_CCVAL\(FREEZE\)](#)

Freeze a channel.

#### Description

Disables the transmission of all PDUs and disables tick processing (timeouts, ACK/NAK, etc) for the specified channel, will still consume all received messages. Note this could cause failures for class 2 transactions in progress.

### Command Structure

[CF\\_UnionArgs\\_Payload\\_t](#) where byte[0] specifies the channel number or all channels

- 255 = all channels
- else = single channel

### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_FREEZE\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_CHAN\\_PARAM\\_ERR\\_EID](#)
- Command processing failure, [CF\\_CMD\\_FREEZE\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_THAW\\_CC](#)

Definition at line 199 of file cf\_fcncodes.h.

### 10.2.2.10 CF\_GET\_PARAM\_CC #define CF\_GET\_PARAM\_CC [CF\\_CCVAL](#)(GET\_PARAM)

Get parameter.

#### Description

Gets a configuration parameter

#### Command Structure

[CF\\_GetParamCmd\\_t](#)

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_GETSET2\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid configuration parameter key, [CF\\_CMD\\_GETSET\\_PARAM\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_GETSET\\_CHAN\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_SET\\_PARAM\\_CC](#)

Definition at line 436 of file cf\_fcncodes.h.

**10.2.2.11 CF\_NOOP\_CC** #define CF\_NOOP\_CC CF\_CCVAL(NOOP)  
No-op.

#### Description

No-operation command for aliveness verification and version reporting

#### Command Structure

No Payload / Arguments

#### Command Verification

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

- CF\_HkPacket\_Payload\_t.counters CF\_HkCmdCounters\_t.cmd will increment
- CF\_NOOP\_INF\_EID

#### Error Conditions

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

- Command packet length not as expected, CF\_CMD\_LEN\_ERR\_EID

#### Evidence of failure may be found in the following telemetry:

- CF\_HkPacket\_Payload\_t.counters CF\_HkCmdCounters\_t.err will increment

#### Criticality

None

Definition at line 68 of file cf\_fcncodes.h.

**10.2.2.12 CF\_NUM\_COMMANDS** #define CF\_NUM\_COMMANDS 24

Command code limit used for validity check and array sizing.

Definition at line 726 of file cf\_fcncodes.h.

**10.2.2.13 CF\_PLAYBACK\_DIR\_CC** #define CF\_PLAYBACK\_DIR\_CC CF\_CCVAL(PLAYBACK\_DIR)  
Playback a directory.

#### Description

Transmits all the files in a directory

#### Command Structure

CF\_PlaybackDirCmd\_t - note it's currently a typedef of CF\_TxFileCmd\_t, where the source filename and destination filename are directories

#### Command Verification

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

- CF\_HkPacket\_Payload\_t.counters CF\_HkCmdCounters\_t.cmd will increment
- CF\_CMD\_PLAYBACK\_DIR\_INF\_EID

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid parameter, [CF\\_CMD\\_BAD\\_PARAM\\_ERR\\_EID](#)
- Playback initialization failure, [CF\\_CMD\\_PLAYBACK\\_DIR\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.err](#) will increment

### Criticality

None

### See also

[CF\\_TX\\_FILE\\_CC](#)

Definition at line 164 of file cf\_fnccodes.h.

**10.2.2.14 CF\_PURGE\_QUEUE\_CC** #define CF\_PURGE\_QUEUE\_CC [CF\\_CCVAL](#)(PURGE\_QUEUE)  
Purge queue.

### Description

Purge the requested queue

### Command Structure

[CF\\_UnionArgs\\_Payload\\_t](#)

byte[0] specifies the channel number or all channels

- 255 = all channels
- else = single channel

byte[1] specifies the queue

- 0 = Pending queue
- 1 = History queue
- 2 = Both pending and history queue

### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_PURGE\\_QUEUE\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_CHAN\\_PARAM\\_ERR\\_EID](#)
- Invalid purge queue argument, [CF\\_CMD\\_PURGE\\_ARG\\_ERR\\_EID](#)
- Purge queue failed, [CF\\_CMD\\_PURGE\\_QUEUE\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_WRITE\\_QUEUE\\_CC](#)

Definition at line 659 of file cf\_fcnCodes.h.

### **10.2.2.15 CF\_RESET\_CC** #define CF\_RESET\_CC [CF\\_CCVAL](#)(RESET\_COUNTERS)

Reset counters.

#### Description

Resets the requested housekeeping counters

#### Command Structure

[CF\\_UnionArgs\\_Payload\\_t](#) where byte[0] specifies the counters type, byte[1-3] don't care:

- 0 = all counters
- 1 = command counters
- 2 = fault counters
- 3 = up counters
- 4 = down counters

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_RESET\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid counter type, [CF\\_CMD\\_RESET\\_INVALID\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- `CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.err` will increment

#### Criticality

None

Definition at line 101 of file cf\_fcncodes.h.

### **10.2.2.16 CF\_RESUME\_CC #define CF\_RESUME\_CC CF\_CCVAL(RESUME)**

Resume a transaction.

#### Description

Enables the transmission of all PDUs and resumes tick processing (timeouts, ACK/NAK, etc) on a single transaction, all channels and transactions, or all transactions on a specific channel. Note a suspended transaction still consume all received messages. Note suspension is tracked per transaction, whereas freeze/thaw are tracked per channel.

#### Command Structure

`CF_Transaction_Payload_t`

#### Command Verification

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

- `CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.cmd` will increment
- `CF_CMD_SUSPRES_INF_EID`
- `CF_CMD_SUSPRES_SAME_INF_EID`

#### Error Conditions

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

- Command packet length not as expected, `CF_CMD_LEN_ERR_EID`
- Transaction not found using compound key, `CF_CMD_TRANS_NOT_FOUND_ERR_EID`
- Invalid channel number, `CF_CMD_TSN_CHAN_INVALID_ERR_EID`
- No matching transaction, `CF_CMD_SUSPRES_CHAN_ERR_EID`

Evidence of failure may be found in the following telemetry:

- `CF_HkPacket_Payload_t.counters CF_HkCmdCounters_t.err` will increment

#### Criticality

None

#### See also

`CF_SUSPEND_CC`, `CF_CANCEL_CC`, `CF_ABANDON_CC`

Definition at line 305 of file cf\_fcncodes.h.

**10.2.2.17 CF\_SET\_PARAM\_CC** #define CF\_SET\_PARAM\_CC CF\_CCVAL(SET\_PARAM)  
Set parameter.

#### Description

Sets a configuration parameter

#### Command Structure

[CF\\_SetParamCmd\\_t](#)

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_GETSET1\\_INF\\_EID](#)

#### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid configuration parameter key, [CF\\_CMD\\_GETSET\\_PARAM\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_GETSET\\_CHAN\\_ERR\\_EID](#)
- Parameter value failed validation, [CF\\_CMD\\_GETSET\\_VALIDATE\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.err](#) will increment

#### Criticality

None

#### See also

[CF\\_GET\\_PARAM\\_CC](#)

Definition at line 405 of file cf\_fcncodes.h.

**10.2.2.18 CF\_SUSPEND\_CC** #define CF\_SUSPEND\_CC CF\_CCVAL(SUSPEND)  
Suspend a transaction.

#### Description

Disables the transmission of all PDUs and disables tick processing (timeouts, ACK/NAK, etc) on a single transaction, all channels and transactions, or all transactions on a specific channel. Will still consume all received messages. Note suspension is tracked per transaction, whereas freeze/thaw are tracked per channel.

#### Command Structure

[CF\\_Transaction\\_Payload\\_t](#)

### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_SUSPRES\\_INF\\_EID](#)
- [CF\\_CMD\\_SUSPRES\\_SAME\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Transaction not found using compound key, [CF\\_CMD\\_TRANS\\_NOT\\_FOUND\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_TSN\\_CHAN\\_INVALID\\_ERR\\_EID](#)
- No matching transaction, [CF\\_CMD\\_SUSPRES\\_CHAN\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_RESUME\\_CC](#), [CF\\_CANCEL\\_CC](#), [CF\\_ABANDON\\_CC](#)

Definition at line 269 of file cf\_fncodes.h.

#### 10.2.2.19 CF\_THAW\_CC #define CF\_THAW\_CC [CF\\_CCVAL](#) (THAW)

Thaw a channel.

### Description

Enables the transmission of all PDUs and resumes tick processing (timeouts, ACK/NAK, etc) for the specified channel, note received messages are consumed either way.

### Command Structure

[CF\\_UnionArgs\\_Payload\\_t](#) where byte[0] specifies the channel number or all channels

- 255 = all channels
- else = single channel

### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_THAW\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_CHAN\\_PARAM\\_ERR\\_EID](#)
- Command processing failure, [CF\\_CMD\\_THAW\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_FREEZE\\_CC](#)

Definition at line 233 of file cf\_fnccodes.h.

### **10.2.2.20 CF\_TX\_FILE\_CC #define CF\_TX\_FILE\_CC CF\_CCVAL(TX\_FILE)**

Transmit file.

#### Description

Requests transmission of a file

#### Command Structure

[CF\\_TxFileCmd\\_t](#)

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_TX\\_FILE\\_INF\\_EID](#)

### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid parameter, [CF\\_CMD\\_BAD\\_PARAM\\_ERR\\_EID](#)
- Transaction initialization failure, [CF\\_CMD\\_TX\\_FILE\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters CF\\_HkCmdCounters\\_t.terr](#) will increment

### Criticality

None

### See also

[CF\\_PLAYBACK\\_DIR\\_CC](#)

Definition at line 132 of file cf\_fnccodes.h.

**10.2.2.21 CF\_WRITE\_QUEUE\_CC** #define CF\_WRITE\_QUEUE\_CC CF\_CCVAL(WRITE\_QUEUE)  
Write queue.

#### Description

Writes requested queue(s) to a file

#### Command Structure

[CF\\_WriteQueueCmd\\_t](#)

#### Command Verification

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

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.cmd](#) will increment
- [CF\\_CMD\\_WQ\\_INF\\_EID](#)

#### Error Conditions

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

- Command packet length not as expected, [CF\\_CMD\\_LEN\\_ERR\\_EID](#)
- Invalid parameter combination, [CF\\_CMD\\_WQ\\_ARGS\\_ERR\\_EID](#)
- Invalid channel number, [CF\\_CMD\\_WQ\\_CHAN\\_ERR\\_EID](#)
- Open file to write failed, [CF\\_CMD\\_WQ\\_OPEN\\_ERR\\_EID](#)
- Write RX data failed, [CF\\_CMD\\_WQ\\_WRITEQ\\_RX\\_ERR\\_EID](#)
- Write RX history data failed, [CF\\_CMD\\_WQ\\_WRITEHIST\\_RX\\_ERR\\_EID](#)
- Write TX data failed, [CF\\_CMD\\_WQ\\_WRITEQ\\_TX\\_ERR\\_EID](#)
- Write TX history data failed, [CF\\_CMD\\_WQ\\_WRITEHIST\\_TX\\_ERR\\_EID](#)

Evidence of failure may be found in the following telemetry:

- [CF\\_HkPacket\\_Payload\\_t.counters](#) [CF\\_HkCmdCounters\\_t.err](#) will increment

#### Criticality

None

#### See also

[CF\\_PURGE\\_QUEUE\\_CC](#)

Definition at line 472 of file cf\_fcnodes.h.

## 10.3 CFS CFDP Platform Configuration

### Macros

- #define CF\_NUM\_CHANNELS CF\_INTERFACE\_CFGVAL(NUM\_CHANNELS)  
*Number of channels.*
- #define DEFAULT\_CF\_NUM\_CHANNELS 2
- #define CF\_NAK\_MAX\_SEGMENTS CF\_INTERFACE\_CFGVAL(NAK\_MAX\_SEGMENTS)  
*Max NAK segments supported in a NAK PDU.*

- #define DEFAULT\_CF\_NAK\_MAX\_SEGMENTS 58
- #define CF\_MAX\_POLLING\_DIR\_PER\_CHAN CF\_INTERFACE\_CFGVAL(MAX\_POLLING\_DIR\_PER\_CHAN)  
*Max number of polling directories per channel.*
- #define DEFAULT\_CF\_MAX\_POLLING\_DIR\_PER\_CHAN 5
- #define CF\_MAX\_PDU\_SIZE CF\_INTERFACE\_CFGVAL(MAX\_PDU\_SIZE)  
*Max PDU size.*
- #define DEFAULT\_CF\_MAX\_PDU\_SIZE 512
- #define CF\_FILENAME\_MAX\_NAME CF\_INTERFACE\_CFGVAL(FILENAME\_MAX\_NAME)  
*Maximum file name length.*
- #define DEFAULT\_CF\_FILENAME\_MAX\_NAME CFE\_MISSION\_MAX\_FILE\_LEN
- #define CF\_FILENAME\_MAX\_LEN CF\_INTERFACE\_CFGVAL(FILENAME\_MAX\_LEN)  
*Max filename and path length.*
- #define DEFAULT\_CF\_FILENAME\_MAX\_LEN CFE\_MISSION\_MAX\_PATH\_LEN
- #define CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES CF\_INTERFACE\_CFGVAL(PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES)  
*Number of trailing bytes to add to CFDP PDU.*
- #define DEFAULT\_CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES 0
- #define CF\_PIPE\_DEPTH CF\_INTERNAL\_CFGVAL(PIPE\_DEPTH)  
*Application Pipe Depth.*
- #define DEFAULT\_CF\_PIPE\_DEPTH 32
- #define CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN CF\_INTERNAL\_CFGVAL(MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN)  
*Number of max commanded playback files per chan.*
- #define DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN 10
- #define CF\_MAX\_SIMULTANEOUS\_RX CF\_INTERNAL\_CFGVAL(MAX\_SIMULTANEOUS\_RX)  
*Max number of simultaneous file receives.*
- #define DEFAULT\_CF\_MAX\_SIMULTANEOUS\_RX 5
- #define CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN CF\_INTERNAL\_CFGVAL(MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN)  
*Max number of commanded playback directories per channel.*
- #define DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN 2
- #define CF\_NUM\_HISTORIES\_PER\_CHANNEL CF\_INTERNAL\_CFGVAL(NUM\_HISTORIES\_PER\_CHANNEL)  
*Number of histories per channel.*
- #define DEFAULT\_CF\_NUM\_HISTORIES\_PER\_CHANNEL 256
- #define CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK CF\_INTERNAL\_CFGVAL(NUM\_TRANSACTIONS\_PER\_PLAYBACK)  
*Number of transactions per playback directory.*
- #define DEFAULT\_CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK 5
- #define CF\_CONFIG\_TABLE\_NAME CF\_INTERNAL\_CFGVAL(CONFIG\_TABLE\_NAME)  
*Name of the CF Configuration Table.*
- #define DEFAULT\_CF\_CONFIG\_TABLE\_NAME "config\_table"
- #define CF\_CONFIG\_TABLE\_FILENAME CF\_INTERNAL\_CFGVAL(CONFIG\_TABLE\_FILENAME)  
*CF Configuration Table Filename.*
- #define DEFAULT\_CF\_CONFIG\_TABLE\_FILENAME "/cf/cf\_def\_config.tbl"
- #define CF\_R2\_CRC\_CHUNK\_SIZE CF\_INTERNAL\_CFGVAL(R2\_CRC\_CHUNK\_SIZE)  
*R2 CRC calc chunk size.*
- #define DEFAULT\_CF\_R2\_CRC\_CHUNK\_SIZE 1024
- #define CF\_RCVMSG\_TIMEOUT CF\_INTERNAL\_CFGVAL(RCVMSG\_TIMEOUT)

- `#define DEFAULT_CF_RCVMSG_TIMEOUT 100`  
*Number of milliseconds to wait for a SB message.*
- `#define CF_STARTUP_SEM_MAX_RETRIES CF_INTERNAL_CFGVAL(STARTUP_SEM_MAX_RETRIES)`  
*Limits the number of retries to obtain the CF throttle sem.*
- `#define DEFAULT_CF_STARTUP_SEM_MAX_RETRIES 25`
- `#define CF_STARTUP_SEM_TASK_DELAY CF_INTERNAL_CFGVAL(STARTUP_SEM_TASK_DELAY)`  
*Number of milliseconds to wait if CF throttle sem is not available.*
- `#define DEFAULT_CF_STARTUP_SEM_TASK_DELAY 100`

### 10.3.1 Detailed Description

### 10.3.2 Macro Definition Documentation

**10.3.2.1 CF\_CONFIG\_TABLE\_FILENAME** `#define CF_CONFIG_TABLE_FILENAME CF_INTERNAL_CFGVAL(CONFIG_TABLE_FILENAME)`  
CF Configuration Table Filename.

Description:

The value of this constant defines the filename of the CF Config Table

Limits

The length of this string, including the NULL terminator cannot exceed the `OS_MAX_PATH_LEN` value.

Definition at line 139 of file cf\_internal\_cfg.h.

**10.3.2.2 CF\_CONFIG\_TABLE\_NAME** `#define CF_CONFIG_TABLE_NAME CF_INTERNAL_CFGVAL(CONFIG_TABLE_NAME)`  
Name of the CF Configuration Table.

Description:

This parameter defines the name of the CF Configuration Table.

Limits

The length of this string, including the NULL terminator cannot exceed the `OS_MAX_PATH_LEN` value.

Definition at line 126 of file cf\_internal\_cfg.h.

**10.3.2.3 CF\_FILENAME\_MAX\_LEN** `#define CF_FILENAME_MAX_LEN CF_INTERFACE_CFGVAL(FILENAME_MAX_LEN)`  
Max filename and path length.

Limits:

Definition at line 117 of file cf\_interface\_cfg.h.

**10.3.2.4 CF\_FILENAME\_MAX\_NAME** #define CF\_FILENAME\_MAX\_NAME CF\_INTERFACE\_CFGVAL(FILENAME\_MAX\_NAME)

Maximum file name length.

Limits:

Definition at line 108 of file cf\_interface\_cfg.h.

**10.3.2.5 CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN** #define CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN CF\_INTERNAL\_CFGVAL(MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN)

Max number of commanded playback directories per channel.

Description:

Each channel can support this number of ground commanded directory playbacks.

Limits:

Definition at line 88 of file cf\_internal\_cfg.h.

**10.3.2.6 CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN** #define CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN CF\_INTERNAL\_CFGVAL(MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN)

Number of max commanded playback files per chan.

Description:

This is the max number of outstanding ground commanded file transmits per channel.

Limits:

Definition at line 62 of file cf\_internal\_cfg.h.

**10.3.2.7 CF\_MAX\_PDU\_SIZE** #define CF\_MAX\_PDU\_SIZE CF\_INTERFACE\_CFGVAL(MAX\_PDU\_SIZE)

Max PDU size.

Description:

Limits the maximum possible Tx PDU size. Note the resulting CCSDS packet also includes a CCSDS header and CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES. The outgoing file data chunk size is also limited from the table configuration or by set parameter command, which is checked against this value (+ smallest possible PDU header).

Note:

This does NOT limit Rx PDUs, since the file data is written from the transport packet to the file.

Limits:

Since PDUs are wrapped in CCSDS packets, need to respect any CCSDS packet size limits on the system.

Definition at line 99 of file cf\_interface\_cfg.h.

**10.3.2.8 CF\_MAX\_POLLING\_DIR\_PER\_CHAN** #define CF\_MAX\_POLLING\_DIR\_PER\_CHAN CF\_INTERFACE\_CFGVAL (MAX←\_ POLLING\_DIR\_PER\_CHAN)  
Max number of polling directories per channel.

**Description:**

This affects the configuration table. There must be an entry (can be empty) for each of these polling directories per channel.

**Limits:**

Definition at line 77 of file cf\_interface\_cfg.h.

**10.3.2.9 CF\_MAX\_SIMULTANEOUS\_RX** #define CF\_MAX\_SIMULTANEOUS\_RX CF\_INTERNAL\_CFGVAL (MAX←\_ SIMULTANEOUS\_RX)  
Max number of simultaneous file receives.

**Description:**

Each channel can support this number of file receive transactions at a time.

**Limits:**

Definition at line 74 of file cf\_internal\_cfg.h.

**10.3.2.10 CF\_NAK\_MAX\_SEGMENTS** #define CF\_NAK\_MAX\_SEGMENTS CF\_INTERFACE\_CFGVAL (NAK\_MAX←\_ SEGMENTS)  
Max NAK segments supported in a NAK PDU.

**Description:**

When a NAK PDU is sent or received, this is the max number of segment requests supported. This number should match the ground CFDP engine configuration as well.

**Limits:**

Definition at line 64 of file cf\_interface\_cfg.h.

**10.3.2.11 CF\_NUM\_CHANNELS** #define CF\_NUM\_CHANNELS CF\_INTERFACE\_CFGVAL (NUM\_CHANNELS)  
Number of channels.

**Description:**

The number of channels in the engine. Changing this value changes the configuration table for the application.

**Limits:**

Must be less <= 200. Obviously it will be smaller than that.

Definition at line 50 of file cf\_interface\_cfg.h.

**10.3.2.12 CF\_NUM\_HISTORIES\_PER\_CHANNEL** #define CF\_NUM\_HISTORIES\_PER\_CHANNEL CF\_INTERNAL\_CFGVAL (NUM↔\_HISTORIES\_PER\_CHANNEL)  
Number of histories per channel.

**Description:**

Each channel can support this number of file receive transactions at a time.

**Limits:**

65536 is the current max.

Definition at line 100 of file cf\_internal\_cfg.h.

**10.3.2.13 CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK** #define CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK CF\_INTERNAL\_CFGVAL (NU↔\_TRANSACTIONS\_PER\_PLAYBACK)  
Number of transactions per playback directory.

**Description:**

Each playback/polling directory operation will be able to have this many active transfers at a time pending or active.

**Limits:**

Definition at line 113 of file cf\_internal\_cfg.h.

**10.3.2.14 CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES** #define CF\_PDU\_ENCAPSULATION\_EXTRA↔\_TRAILING\_BYTES CF\_INTERFACE\_CFGVAL (PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES)  
Number of trailing bytes to add to CFDP PDU.

**Description**

Additional padding bytes to be appended to the tail of CFDP PDUs. This reserves extra space to the software bus encapsulation buffer for every CFDP PDU such that platform-specific trailer information may be added. This includes, but is not limited to a separate CRC or error control field in addition to the error control field(s) within the the nominal CFDP protocol.

These extra bytes are added at the software bus encapsulation layer, they are not part of the CFDP PDU itself.  
Set to 0 to disable this feature, such that the software bus buffer encapsulates only the CFDP PDU and no extra bytes are added.

**Limits:**

Maximum value is the difference between the maximum size of a CFDP PDU and the maximum size of an SB message.

Definition at line 140 of file cf\_interface\_cfg.h.

**10.3.2.15 CF\_PIPE\_DEPTH** #define CF\_PIPE\_DEPTH CF\_INTERNAL\_CFGVAL (PIPE\_DEPTH)  
Application Pipe Depth.

**Description:**

Dictates the pipe depth of the cf command pipe.

**Limits:**

The minimum size of this parameter is 1 The maximum size dictated by cFE platform configuration parameter is OS\_QUEUE\_MAX\_DEPTH

Definition at line 50 of file cf\_internal\_cfg.h.

**10.3.2.16 CF\_R2\_CRC\_CHUNK\_SIZE** #define CF\_R2\_CRC\_CHUNK\_SIZE CF\_INTERNAL\_CFGVAL(R2\_CRC\_CHUNK\_SIZE)  
R2 CRC calc chunk size.

**Description**

R2 performs CRC calculation upon file completion in chunks. This is the size of the buffer. The larger the size the more stack will be used, but the faster it can go. The overall number of bytes calculated per wakeup is set in the configuration table.

**Limits:**

Definition at line 154 of file cf\_internal\_cfg.h.

**10.3.2.17 CF\_RCVMSG\_TIMEOUT** #define CF\_RCVMSG\_TIMEOUT CF\_INTERNAL\_CFGVAL(RCVMSG\_TIMEOUT)  
Number of milliseconds to wait for a SB message.  
Definition at line 160 of file cf\_internal\_cfg.h.

**10.3.2.18 CF\_STARTUP\_SEM\_MAX\_RETRIES** #define CF\_STARTUP\_SEM\_MAX\_RETRIES CF\_INTERNAL\_CFGVAL(STARTUP\_SEM\_MAX\_RETRIES)  
Limits the number of retries to obtain the CF throttle sem.

**Description**

If the CF throttle sem is not available during CF startup, the initialization will retry after a short delay.

**See also**

[CF\\_STARTUP\\_SEM\\_TASK\\_DELAY](#)

Definition at line 172 of file cf\_internal\_cfg.h.

**10.3.2.19 CF\_STARTUP\_SEM\_TASK\_DELAY** #define CF\_STARTUP\_SEM\_TASK\_DELAY CF\_INTERNAL\_CFGVAL(STARTUP\_SEM\_TASK\_DELAY)  
Number of milliseconds to wait if CF throttle sem is not available.

**Description**

If the CF throttle sem is not available during CF startup, the initialization will delay for this period of time before trying again

**See also**

[CF\\_STARTUP\\_SEM\\_MAX\\_RETRIES](#)

Definition at line 184 of file cf\_internal\_cfg.h.

**10.3.2.20 DEFAULT\_CF\_CONFIG\_TABLE\_FILENAME** #define DEFAULT\_CF\_CONFIG\_TABLE\_FILENAME "/cf/cf←  
\_def\_config.tbl"

Definition at line 140 of file cf\_internal\_cfg.h.

**10.3.2.21 DEFAULT\_CF\_CONFIG\_TABLE\_NAME** #define DEFAULT\_CF\_CONFIG\_TABLE\_NAME "config\_table"

Definition at line 127 of file cf\_internal\_cfg.h.

**10.3.2.22 DEFAULT\_CF\_FILENAME\_MAX\_LEN** #define DEFAULT\_CF\_FILENAME\_MAX\_LEN CFE\_MISSION\_MAX\_PATH\_LEN

Definition at line 118 of file cf\_interface\_cfg.h.

**10.3.2.23 DEFAULT\_CF\_FILENAME\_MAX\_NAME** #define DEFAULT\_CF\_FILENAME\_MAX\_NAME CFE\_MISSION\_MAX\_FILE\_LEN

Definition at line 109 of file cf\_interface\_cfg.h.

**10.3.2.24 DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN** #define DEFAULT\_CF←  
\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN 2

Definition at line 89 of file cf\_internal\_cfg.h.

**10.3.2.25 DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN** #define DEFAULT\_CF\_MAX←  
COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN 10

Definition at line 63 of file cf\_internal\_cfg.h.

**10.3.2.26 DEFAULT\_CF\_MAX\_PDU\_SIZE** #define DEFAULT\_CF\_MAX\_PDU\_SIZE 512

Definition at line 100 of file cf\_interface\_cfg.h.

**10.3.2.27 DEFAULT\_CF\_MAX\_POLLING\_DIR\_PER\_CHAN** #define DEFAULT\_CF\_MAX\_POLLING\_DIR\_PER←  
CHAN 5

Definition at line 78 of file cf\_interface\_cfg.h.

**10.3.2.28 DEFAULT\_CF\_MAX\_SIMULTANEOUS\_RX** #define DEFAULT\_CF\_MAX\_SIMULTANEOUS\_RX 5

Definition at line 75 of file cf\_internal\_cfg.h.

**10.3.2.29 DEFAULT\_CF\_NAK\_MAX\_SEGMENTS** #define DEFAULT\_CF\_NAK\_MAX\_SEGMENTS 58

Definition at line 65 of file cf\_interface\_cfg.h.

**10.3.2.30 DEFAULT\_CF\_NUM\_CHANNELS** #define DEFAULT\_CF\_NUM\_CHANNELS 2

Definition at line 51 of file cf\_interface\_cfg.h.

**10.3.2.31 DEFAULT\_CF\_NUM\_HISTORIES\_PER\_CHANNEL** #define DEFAULT\_CF\_NUM\_HISTORIES\_PER←  
CHANNEL 256

Definition at line 101 of file cf\_internal\_cfg.h.

**10.3.2.32 DEFAULT\_CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK** #define DEFAULT\_CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK 5  
Definition at line 114 of file cf\_internal\_cfg.h.

**10.3.2.33 DEFAULT\_CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES** #define DEFAULT\_CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES 0  
Definition at line 141 of file cf\_interface\_cfg.h.

**10.3.2.34 DEFAULT\_CF\_PIPE\_DEPTH** #define DEFAULT\_CF\_PIPE\_DEPTH 32  
Definition at line 51 of file cf\_internal\_cfg.h.

**10.3.2.35 DEFAULT\_CF\_R2\_CRC\_CHUNK\_SIZE** #define DEFAULT\_CF\_R2\_CRC\_CHUNK\_SIZE 1024  
Definition at line 155 of file cf\_internal\_cfg.h.

**10.3.2.36 DEFAULT\_CF\_RCVMSG\_TIMEOUT** #define DEFAULT\_CF\_RCVMSG\_TIMEOUT 100  
Definition at line 161 of file cf\_internal\_cfg.h.

**10.3.2.37 DEFAULT\_CF\_STARTUP\_SEM\_MAX\_RETRIES** #define DEFAULT\_CF\_STARTUP\_SEM\_MAX\_RETRIES 25  
Definition at line 173 of file cf\_internal\_cfg.h.

**10.3.2.38 DEFAULT\_CF\_STARTUP\_SEM\_TASK\_DELAY** #define DEFAULT\_CF\_STARTUP\_SEM\_TASK\_DELAY 100  
Definition at line 185 of file cf\_internal\_cfg.h.

## 10.4 CFS CFDP Mission Configuration

### Macros

- #define CF\_PERF\_ID\_APPMAIN (11)  
*Main application performance ID.*
- #define CF\_PERF\_ID\_FSEEK (12)  
*File seek performance ID.*
- #define CF\_PERF\_ID\_FOPEN (13)  
*File open performance ID.*
- #define CF\_PERF\_ID\_FCLOSE (14)  
*File close performance ID.*
- #define CF\_PERF\_ID\_FREAD (15)  
*File read performance ID.*
- #define CF\_PERF\_ID\_FWRITE (16)  
*File write performance ID.*
- #define CF\_PERF\_ID\_CYCLE\_ENG (17)  
*Cycle engine performance ID.*
- #define CF\_PERF\_ID\_DIRREAD (18)  
*Directory read performance ID.*
- #define CF\_PERF\_ID\_CREAT (19)

*Create performance ID.*

- `#define CF_PERF_ID_RENAME (20)`  
*Rename performance ID.*
- `#define CF_PERF_ID_PDURCVD(x) (30 + x)`  
*PDU Received performance ID.*
- `#define CF_PERF_ID_PDUSENT(x) (40 + x)`  
*PDU Sent performance ID.*

#### 10.4.1 Detailed Description

#### 10.4.2 Macro Definition Documentation

##### 10.4.2.1 **CF\_PERF\_ID\_APPMAIN** `#define CF_PERF_ID_APPMAIN (11)`

Main application performance ID.

Definition at line 33 of file cf\_perfid.h.

##### 10.4.2.2 **CF\_PERF\_ID\_CREAT** `#define CF_PERF_ID_CREAT (19)`

Create performance ID.

Definition at line 41 of file cf\_perfid.h.

##### 10.4.2.3 **CF\_PERF\_ID\_CYCLE\_ENG** `#define CF_PERF_ID_CYCLE_ENG (17)`

Cycle engine performance ID.

Definition at line 39 of file cf\_perfid.h.

##### 10.4.2.4 **CF\_PERF\_ID\_DIRREAD** `#define CF_PERF_ID_DIRREAD (18)`

Directory read performance ID.

Definition at line 40 of file cf\_perfid.h.

##### 10.4.2.5 **CF\_PERF\_ID\_FCLOSE** `#define CF_PERF_ID_FCLOSE (14)`

File close performance ID.

Definition at line 36 of file cf\_perfid.h.

##### 10.4.2.6 **CF\_PERF\_ID\_FOPEN** `#define CF_PERF_ID_FOPEN (13)`

File open performance ID.

Definition at line 35 of file cf\_perfid.h.

##### 10.4.2.7 **CF\_PERF\_ID\_FREAD** `#define CF_PERF_ID_FREAD (15)`

File read performance ID.

Definition at line 37 of file cf\_perfid.h.

##### 10.4.2.8 **CF\_PERF\_ID\_FSEEK** `#define CF_PERF_ID_FSEEK (12)`

File seek performance ID.

Definition at line 34 of file cf\_perfid.h.

**10.4.2.9 CF\_PERF\_ID\_FWRITE** #define CF\_PERF\_ID\_FWRITE (16)

File write performance ID.

Definition at line 38 of file cf\_perfids.h.

**10.4.2.10 CF\_PERF\_ID\_PDURCVD** #define CF\_PERF\_ID\_PDURCVD (

x ) (30 + x)

PDU Received performance ID.

Definition at line 44 of file cf\_perfids.h.

**10.4.2.11 CF\_PERF\_ID\_PDUSENT** #define CF\_PERF\_ID\_PDUSENT (

x ) (40 + x)

PDU Sent performance ID.

Definition at line 45 of file cf\_perfids.h.

**10.4.2.12 CF\_PERF\_ID\_RENAME** #define CF\_PERF\_ID\_RENAME (20)

Rename performance ID.

Definition at line 42 of file cf\_perfids.h.

## 10.5 CFS CFDP Version

### Macros

- #define CF\_MAJOR\_VERSION (7)  
*Major version number.*
- #define CF\_MINOR\_VERSION (0)  
*Minor version number.*
- #define CF\_REVISION (0)  
*Revision number.*

### 10.5.1 Detailed Description

#### Version Numbers

### 10.5.2 Macro Definition Documentation

**10.5.2.1 CF\_MAJOR\_VERSION** #define CF\_MAJOR\_VERSION (7)

Major version number.

Definition at line 35 of file cf\_version.h.

**10.5.2.2 CF\_MINOR\_VERSION** #define CF\_MINOR\_VERSION (0)

Minor version number.

Definition at line 36 of file cf\_version.h.

**10.5.2.3 CF\_REVISION** #define CF\_REVISION (0)

Revision number.

Definition at line 37 of file cf\_version.h.

## 10.6 CFS CFDP Telemetry

### Data Structures

- struct `CF_HkCmdCounters`  
*Housekeeping command counters.*
- struct `CF_HkSent`  
*Housekeeping sent counters.*
- struct `CF_HkRecv`  
*Housekeeping received counters.*
- struct `CF_HkFault`  
*Housekeeping fault counters.*
- struct `CF_HkCounters`  
*Housekeeping counters.*
- struct `CF_HkChannel_Data`  
*Housekeeping channel data.*
- struct `CF_HkPacket_Payload`  
*Housekeeping packet.*
- struct `CF_EotPacket_Payload`  
*End of transaction packet.*
- struct `CF_HkPacket`  
*Housekeeping packet.*
- struct `CF_EotPacket`  
*End of transaction packet.*

### Typedefs

- typedef struct `CF_HkCmdCounters` `CF_HkCmdCounters_t`  
*Housekeeping command counters.*
- typedef struct `CF_HkSent` `CF_HkSent_t`  
*Housekeeping sent counters.*
- typedef struct `CF_HkRecv` `CF_HkRecv_t`  
*Housekeeping received counters.*
- typedef struct `CF_HkFault` `CF_HkFault_t`  
*Housekeeping fault counters.*
- typedef struct `CF_HkCounters` `CF_HkCounters_t`  
*Housekeeping counters.*
- typedef struct `CF_HkChannel_Data` `CF_HkChannel_Data_t`  
*Housekeeping channel data.*
- typedef struct `CF_HkPacket_Payload` `CF_HkPacket_Payload_t`  
*Housekeeping packet.*
- typedef struct `CF_EotPacket_Payload` `CF_EotPacket_Payload_t`  
*End of transaction packet.*
- typedef struct `CF_HkPacket` `CF_HkPacket_t`  
*Housekeeping packet.*
- typedef struct `CF_EotPacket` `CF_EotPacket_t`  
*End of transaction packet.*

**10.6.1 Detailed Description****10.6.2 Typedef Documentation**

**10.6.2.1 CF\_EotPacket\_Payload\_t** `typedef struct CF_EotPacket_Payload CF_EotPacket_Payload_t`  
End of transaction packet.

**10.6.2.2 CF\_EotPacket\_t** `typedef struct CF_EotPacket CF_EotPacket_t`  
End of transaction packet.

**10.6.2.3 CF\_HkChannel\_Data\_t** `typedef struct CF_HkChannel_Data CF_HkChannel_Data_t`  
Housekeeping channel data.

**10.6.2.4 CF\_HkCmdCounters\_t** `typedef struct CF_HkCmdCounters CF_HkCmdCounters_t`  
Housekeeping command counters.

**10.6.2.5 CF\_HkCounters\_t** `typedef struct CF_HkCounters CF_HkCounters_t`  
Housekeeping counters.

**10.6.2.6 CF\_HkFault\_t** `typedef struct CF_HkFault CF_HkFault_t`  
Housekeeping fault counters.

**10.6.2.7 CF\_HkPacket\_Payload\_t** `typedef struct CF_HkPacket_Payload CF_HkPacket_Payload_t`  
Housekeeping packet.

**10.6.2.8 CF\_HkPacket\_t** `typedef struct CF_HkPacket CF_HkPacket_t`  
Housekeeping packet.

**10.6.2.9 CF\_HkRecv\_t** `typedef struct CF_HkRecv CF_HkRecv_t`  
Housekeeping received counters.

**10.6.2.10 CF\_HkSent\_t** `typedef struct CF_HkSent CF_HkSent_t`  
Housekeeping sent counters.

## 10.7 CFS CFDP Command Structures

### Data Structures

- union `CF_UnionArgs_Payload`  
*Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.*
- struct `CF_GetParam_Payload`

- struct **CF\_SetParam\_Payload**  
*Get parameter command structure.*
- struct **CF\_TxFile\_Payload**  
*Set parameter command structure.*
- struct **CF\_WriteQueue\_Payload**  
*Transmit file command structure.*
- struct **CF\_Transaction\_Payload**  
*Write Queue command structure.*
- struct **CF\_NoopCmd**  
*Noop command structure.*
- struct **CF\_EnableEngineCmd**  
*EnableEngine command structure.*
- struct **CF\_DisableEngineCmd**  
*DisableEngine command structure.*
- struct **CF\_ResetCountersCmd**  
*Reset command structure.*
- struct **CF\_FreezeCmd**  
*Freeze command structure.*
- struct **CF\_ThawCmd**  
*Thaw command structure.*
- struct **CF\_EnableDequeueCmd**  
*EnableDequeue command structure.*
- struct **CF\_DisableDequeueCmd**  
*DisableDequeue command structure.*
- struct **CF\_EnableDirPollingCmd**  
*EnableDirPolling command structure.*
- struct **CF\_DisableDirPollingCmd**  
*DisableDirPolling command structure.*
- struct **CF\_PurgeQueueCmd**  
*PurgeQueue command structure.*
- struct **CF\_GetParamCmd**  
*Get parameter command structure.*
- struct **CF\_SetParamCmd**  
*Set parameter command structure.*
- struct **CF\_TxFileCmd**  
*Transmit file command structure.*
- struct **CF\_WriteQueueCmd**  
*Write Queue command structure.*
- struct **CF\_PlaybackDirCmd**  
*Playback directory command structure.*
- struct **CF\_SuspendCmd**  
*Suspend command structure.*
- struct **CF\_ResumeCmd**  
*Resume command structure.*
- struct **CF\_CancelCmd**  
*Cancel command structure.*

- struct `CF_AbandonCmd`  
*Abandon command structure.*
- struct `CF_SendHkCmd`  
*Send Housekeeping Command.*
- struct `CF_WakeupCmd`  
*Wake Up Command.*

## Typedefs

- typedef union `CF_UnionArgs_Payload` `CF_UnionArgs_Payload_t`  
*Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.*
- typedef struct `CF_GetParam_Payload` `CF_GetParam_Payload_t`  
*Get parameter command structure.*
- typedef struct `CF_SetParam_Payload` `CF_SetParam_Payload_t`  
*Set parameter command structure.*
- typedef struct `CF_TxFile_Payload` `CF_TxFile_Payload_t`  
*Transmit file command structure.*
- typedef struct `CF_WriteQueue_Payload` `CF_WriteQueue_Payload_t`  
*Write Queue command structure.*
- typedef struct `CF_Transaction_Payload` `CF_Transaction_Payload_t`  
*Transaction command structure.*
- typedef struct `CF_NoopCmd` `CF_NoopCmd_t`  
*Noop command structure.*
- typedef struct `CF_EnableEngineCmd` `CF_EnableEngineCmd_t`  
*EnableEngine command structure.*
- typedef struct `CF_DisableEngineCmd` `CF_DisableEngineCmd_t`  
*DisableEngine command structure.*
- typedef struct `CF_ResetCountersCmd` `CF_ResetCountersCmd_t`  
*Reset command structure.*
- typedef struct `CF_FreezeCmd` `CF_FreezeCmd_t`  
*Freeze command structure.*
- typedef struct `CF_ThawCmd` `CF_ThawCmd_t`  
*Thaw command structure.*
- typedef struct `CF_EnableDequeueCmd` `CF_EnableDequeueCmd_t`  
*EnableDequeue command structure.*
- typedef struct `CF_DisableDequeueCmd` `CF_DisableDequeueCmd_t`  
*DisableDequeue command structure.*
- typedef struct `CF_EnableDirPollingCmd` `CF_EnableDirPollingCmd_t`  
*EnableDirPolling command structure.*
- typedef struct `CF_DisableDirPollingCmd` `CF_DisableDirPollingCmd_t`  
*DisableDirPolling command structure.*
- typedef struct `CF_PurgeQueueCmd` `CF_PurgeQueueCmd_t`  
*PurgeQueue command structure.*
- typedef struct `CF_GetParamCmd` `CF_GetParamCmd_t`  
*Get parameter command structure.*
- typedef struct `CF_SetParamCmd` `CF_SetParamCmd_t`  
*Set parameter command structure.*
- typedef struct `CF_TxFileCmd` `CF_TxFileCmd_t`

*Transmit file command structure.*

- **typedef struct CF\_WriteQueueCmd CF\_WriteQueueCmd\_t**  
*Write Queue command structure.*
- **typedef struct CF\_PlaybackDirCmd CF\_PlaybackDirCmd\_t**  
*Playback directory command structure.*
- **typedef struct CF\_SuspendCmd CF\_SuspendCmd\_t**  
*Suspend command structure.*
- **typedef struct CF\_ResumeCmd CF\_ResumeCmd\_t**  
*Resume command structure.*
- **typedef struct CF\_CancelCmd CF\_CancelCmd\_t**  
*Cancel command structure.*
- **typedef struct CF\_AbandonCmd CF\_AbandonCmd\_t**  
*Abandon command structure.*
- **typedef struct CF\_SendHkCmd CF\_SendHkCmd\_t**  
*Send Housekeeping Command.*
- **typedef struct CF\_WakeupCmd CF\_WakeupCmd\_t**  
*Wake Up Command.*

## Enumerations

- **enum CF\_Reset\_t {**  
  **CF\_Reset\_all = 0 , CF\_Reset\_command = 1 , CF\_Reset\_fault = 2 , CF\_Reset\_up = 3 ,**  
  **CF\_Reset\_down = 4 }**  
*IDs for use for Reset cmd.*
- **enum CF\_Type\_t { CF\_Type\_all = 0 , CF\_Type\_up = 1 , CF\_Type\_down = 2 }**  
*Type IDs for use for Write Queue cmd.*
- **enum CF\_Queue\_t { CF\_Queue\_pend = 0 , CF\_Queue\_active = 1 , CF\_Queue\_history = 2 , CF\_Queue\_all = 3 }**  
*Queue IDs for use for Write Queue cmd.*
- **enum CF\_GetSet\_ValueID\_t {**  
  **CF\_GetSet\_ValueID\_ticks\_per\_second , CF\_GetSet\_ValueID\_rx\_crc\_calc\_bytes\_per\_wakeup , CF\_GetSet\_ValueID\_ack\_timer\_s ,**  
  **CF\_GetSet\_ValueID\_nak\_timer\_s ,**  
  **CF\_GetSet\_ValueID\_inactivity\_timer\_s , CF\_GetSet\_ValueID\_outgoing\_file\_chunk\_size , CF\_GetSet\_ValueID\_ack\_limit ,**  
  **CF\_GetSet\_ValueID\_nak\_limit ,**  
  **CF\_GetSet\_ValueID\_local\_eid , CF\_GetSet\_ValueID\_chan\_max\_outgoing\_messages\_per\_wakeup , CF\_GetSet\_ValueID\_MAX }**  
  
*Parameter IDs for use with Get/Set parameter messages.*

### 10.7.1 Detailed Description

### 10.7.2 Typedef Documentation

**10.7.2.1 CF\_AbandonCmd\_t** `typedef struct CF_AbandonCmd CF_AbandonCmd_t`  
Abandon command structure.  
For command details see [CF\\_ABANDON\\_CC](#)

**10.7.2.2 CF\_CancelCmd\_t** `typedef struct CF_CancelCmd CF_CancelCmd_t`  
Cancel command structure.  
For command details see [CF\\_CANCEL\\_CC](#)

**10.7.2.3 CF\_DisableDequeueCmd\_t** `typedef struct CF_DisableDequeueCmd CF_DisableDequeueCmd_t`  
DisableDequeue command structure.

For command details see [CF\\_DISABLE\\_DEQUEUE\\_CC](#)

**10.7.2.4 CF\_DisableDirPollingCmd\_t** `typedef struct CF_DisableDirPollingCmd CF_DisableDirPollingCmd_t`  
DisableDirPolling command structure.

For command details see [CF\\_DISABLE\\_DIR\\_POLLING\\_CC](#)

**10.7.2.5 CF\_DisableEngineCmd\_t** `typedef struct CF_DisableEngineCmd CF_DisableEngineCmd_t`  
DisableEngine command structure.

For command details see [CF\\_DISABLE\\_ENGINE\\_CC](#)

**10.7.2.6 CF\_EnableDequeueCmd\_t** `typedef struct CF_EnableDequeueCmd CF_EnableDequeueCmd_t`  
EnableDequeue command structure.

For command details see [CF\\_ENABLE\\_DEQUEUE\\_CC](#)

**10.7.2.7 CF\_EnableDirPollingCmd\_t** `typedef struct CF_EnableDirPollingCmd CF_EnableDirPollingCmd_t`  
EnableDirPolling command structure.

For command details see [CF\\_ENABLE\\_DIR\\_POLLING\\_CC](#)

**10.7.2.8 CF\_EnableEngineCmd\_t** `typedef struct CF_EnableEngineCmd CF_EnableEngineCmd_t`  
EnableEngine command structure.

For command details see [CF\\_ENABLE\\_ENGINE\\_CC](#)

**10.7.2.9 CF\_FreezeCmd\_t** `typedef struct CF_FreezeCmd CF_FreezeCmd_t`  
Freeze command structure.

For command details see [CF\\_FREEZE\\_CC](#)

**10.7.2.10 CF\_GetParam\_Payload\_t** `typedef struct CF_GetParam_Payload CF_GetParam_Payload_t`  
Get parameter command structure.

For command details see [CF\\_GET\\_PARAM\\_CC](#)

**10.7.2.11 CF\_GetParamCmd\_t** `typedef struct CF_GetParamCmd CF_GetParamCmd_t`  
Get parameter command structure.

For command details see [CF\\_GET\\_PARAM\\_CC](#)

**10.7.2.12 CF\_NoopCmd\_t** `typedef struct CF_NoopCmd CF_NoopCmd_t`  
Noop command structure.

For command details see [CF\\_NOOP\\_CC](#)

**10.7.2.13 CF\_PlaybackDirCmd\_t** `typedef struct CF_PlaybackDirCmd CF_PlaybackDirCmd_t`  
Playback directory command structure.

For command details see [CF\\_PLAYBACK\\_DIR\\_CC](#)

**10.7.2.14 CF\_PurgeQueueCmd\_t** `typedef struct CF_PurgeQueueCmd CF_PurgeQueueCmd_t`  
PurgeQueue command structure.

For command details see [CF\\_PURGE\\_QUEUE\\_CC](#)

**10.7.2.15 CF\_ResetCountersCmd\_t** `typedef struct CF_ResetCountersCmd CF_ResetCountersCmd_t`  
Reset command structure.

For command details see [CF\\_RESET\\_CC](#)

**10.7.2.16 CF\_ResumeCmd\_t** `typedef struct CF_ResumeCmd CF_ResumeCmd_t`  
Resume command structure.

For command details see [CF\\_RESUME\\_CC](#)

**10.7.2.17 CF\_SendHkCmd\_t** `typedef struct CF_SendHkCmd CF_SendHkCmd_t`  
Send Housekeeping Command.

Internal notification from SCH with no payload

**10.7.2.18 CF\_SetParam\_Payload\_t** `typedef struct CF_SetParam_Payload CF_SetParam_Payload_t`  
Set parameter command structure.

For command details see [CF\\_SET\\_PARAM\\_CC](#)

**10.7.2.19 CF\_SetParamCmd\_t** `typedef struct CF_SetParamCmd CF_SetParamCmd_t`  
Set parameter command structure.

For command details see [CF\\_SET\\_PARAM\\_CC](#)

**10.7.2.20 CF\_SuspendCmd\_t** `typedef struct CF_SuspendCmd CF_SuspendCmd_t`  
Suspend command structure.

For command details see [CF\\_SUSPEND\\_CC](#)

**10.7.2.21 CF\_ThawCmd\_t** `typedef struct CF_ThawCmd CF_ThawCmd_t`  
Thaw command structure.

For command details see [CF\\_THAW\\_CC](#)

**10.7.2.22 CF\_Transaction\_Payload\_t** `typedef struct CF_Transaction_Payload CF_Transaction_Payload_t`  
Transaction command structure.

For command details see [CF\\_SUSPEND\\_CC, CF\\_RESUME\\_CC, CF\\_CANCEL\\_CC, CF\\_ABANDON\\_CC](#)

**10.7.2.23 CF\_TxFile\_Payload\_t** `typedef struct CF_TxFile_Payload CF_TxFile_Payload_t`  
Transmit file command structure.

For command details see [CF\\_TX\\_FILE\\_CC](#)

**10.7.2.24 CF\_TxFileCmd\_t** `typedef struct CF_TxFileCmd CF_TxFileCmd_t`  
Transmit file command structure.

For command details see [CF\\_TX\\_FILE\\_CC](#)

**10.7.2.25 CF\_UnionArgs\_Payload\_t** `typedef union CF_UnionArgs_Payload CF_UnionArgs_Payload_t`  
Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.

**10.7.2.26 CF\_WakeupCmd\_t** `typedef struct CF_WakeupCmd CF_WakeupCmd_t`  
Wake Up Command.

Internal notification from SCH with no payload

**10.7.2.27 CF\_WriteQueue\_Payload\_t** `typedef struct CF_WriteQueue_Payload CF_WriteQueue_Payload_t`  
Write Queue command structure.

For command details see [CF\\_WRITE\\_QUEUE\\_CC](#)

**10.7.2.28 CF\_WriteQueueCmd\_t** `typedef struct CF_WriteQueueCmd CF_WriteQueueCmd_t`  
Write Queue command structure.

For command details see [CF\\_WRITE\\_QUEUE\\_CC](#)

### 10.7.3 Enumeration Type Documentation

#### 10.7.3.1 CF\_Set\_ValueID\_t enum [CF\\_Set\\_ValueID\\_t](#)

Parameter IDs for use with Get/Set parameter messages.

Specifically these are used for the "key" field within CF\_SetParamCmd\_t and CF\_SetParamCmd\_t message structures.

##### Enumerator

CF_Set_ValueID_ticks_per_second	Ticks per second key.
CF_Set_ValueID_rx_crc_calc_bytes_per_wakeup	Receive CRC calculated bytes per wake-up key.
CF_Set_ValueID_ack_timer_s	ACK timer in seconds key.
CF_Set_ValueID_nak_timer_s	NAK timer in seconds key.
CF_Set_ValueID_inactivity_timer_s	Inactivity timer in seconds key.
CF_Set_ValueID_outgoing_file_chunk_size	Outgoing file chunk size key.
CF_Set_ValueID_ack_limit	ACK retry limit key.
CF_Set_ValueID_nak_limit	NAK retry limit key.
CF_Set_ValueID_local_eid	Local entity id key.
CF_Set_ValueID_chan_max_outgoing_messages_per_wakeup	Max outgoing messages per wake-up key.
CF_Set_ValueID_MAX	Key limit used for validity check.

Definition at line 197 of file default\_cf\_msgdefs.h.

#### 10.7.3.2 CF\_Queue\_t enum [CF\\_Queue\\_t](#)

Queue IDs for use for Write Queue cmd.

##### Enumerator

CF_Queue_pend	Queue pending.
CF_Queue_active	Queue active.
CF_Queue_history	Queue history.
CF_Queue_all	Queue all.

Definition at line 183 of file default\_cf\_msgdefs.h.

#### 10.7.3.3 CF\_Reset\_t enum [CF\\_Reset\\_t](#)

IDs for use for Reset cmd.

##### Enumerator

CF_Reset_all	Reset all.
CF_Reset_command	Reset command.
CF_Reset_fault	Reset fault.
CF_Reset_up	Reset up.
CF_Reset_down	Reset down.

Definition at line 161 of file default\_cf\_msgdefs.h.

**10.7.3.4 CF\_Type\_t** enum [CF\\_Type\\_t](#)

Type IDs for use for Write Queue cmd.

**Enumerator**

CF_Type_all	Type all.
CF_Type_up	Type up.
CF_Type_down	Type down.

Definition at line 173 of file default\_cf\_msgdefs.h.

## 10.8 CFS CFDP Command Message IDs

**Macros**

- #define CF\_CMD\_MID [CFE\\_PLATFORM\\_CF\\_CMD\\_MIDVAL\(CMD\)](#)  
*Message ID for commands.*
- #define CF\_SEND\_HK\_MID [CFE\\_PLATFORM\\_CF\\_CMD\\_MIDVAL\(SEND\\_HK\)](#)  
*Message ID to request housekeeping telemetry.*
- #define CF\_WAKE\_UP\_MID [CFE\\_PLATFORM\\_CF\\_CMD\\_MIDVAL\(WAKE\\_UP\)](#)  
*Message ID for waking up the processing cycle.*

### 10.8.1 Detailed Description

### 10.8.2 Macro Definition Documentation

**10.8.2.1 CF\_CMD\_MID** #define CF\_CMD\_MID [CFE\\_PLATFORM\\_CF\\_CMD\\_MIDVAL\(CMD\)](#)

Message ID for commands.

Definition at line 36 of file default\_cf\_msgids.h.

**10.8.2.2 CF\_SEND\_HK\_MID** #define CF\_SEND\_HK\_MID [CFE\\_PLATFORM\\_CF\\_CMD\\_MIDVAL\(SEND\\_HK\)](#)

Message ID to request housekeeping telemetry.

Definition at line 39 of file default\_cf\_msgids.h.

**10.8.2.3 CF\_WAKE\_UP\_MID** #define CF\_WAKE\_UP\_MID [CFE\\_PLATFORM\\_CF\\_CMD\\_MIDVAL\(WAKE\\_UP\)](#)

Message ID for waking up the processing cycle.

Definition at line 42 of file default\_cf\_msgids.h.

## 10.9 CFS CFDP Telemetry Message IDs

**Macros**

- #define CF\_HK\_TLM\_MID [CFE\\_PLATFORM\\_CF\\_TLM\\_MIDVAL\(HK\\_TLM\)](#)  
*Message ID for housekeeping telemetry.*
- #define CF\_EOT\_TLM\_MID [CFE\\_PLATFORM\\_CF\\_TLM\\_MIDVAL\(EOT\\_TLM\)](#)  
*Message ID for end of transaction telemetry.*

### 10.9.1 Detailed Description

### 10.9.2 Macro Definition Documentation

#### 10.9.2.1 CF\_EOT\_TLM\_MID #define CF\_EOT\_TLM\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(EOT\_TLM)

Message ID for end of transaction telemetry.

Definition at line 55 of file default\_cf\_msgids.h.

#### 10.9.2.2 CF\_HK\_TLM\_MID #define CF\_HK\_TLM\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(HK\_TLM)

Message ID for housekeeping telemetry.

Definition at line 52 of file default\_cf\_msgids.h.

## 10.10 CFS CFDP Data Interface Message IDs

### Macros

- #define CF\_CH0\_TX\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(CH0\_TX)
- #define CF\_CH1\_TX\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(CH1\_TX)
- #define CF\_CH0\_RX\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(CH0\_RX)
- #define CF\_CH1\_RX\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(CH1\_RX)

### 10.10.1 Detailed Description

### 10.10.2 Macro Definition Documentation

#### 10.10.2.1 CF\_CH0\_RX\_MID #define CF\_CH0\_RX\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(CH0\_RX)

Definition at line 66 of file default\_cf\_msgids.h.

#### 10.10.2.2 CF\_CH0\_TX\_MID #define CF\_CH0\_TX\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(CH0\_TX)

Definition at line 64 of file default\_cf\_msgids.h.

#### 10.10.2.3 CF\_CH1\_RX\_MID #define CF\_CH1\_RX\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(CH1\_RX)

Definition at line 67 of file default\_cf\_msgids.h.

#### 10.10.2.4 CF\_CH1\_TX\_MID #define CF\_CH1\_TX\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(CH1\_TX)

Definition at line 65 of file default\_cf\_msgids.h.

## 10.11 cFE Return Code Defines

### Macros

- #define CFE\_SUCCESS ((CFE\_Status\_t)0)  
*Successful execution.*
- #define CFE\_STATUS\_NO\_COUNTER\_INCREMENT ((CFE\_Status\_t)0x48000001)  
*No Counter Increment.*
- #define CFE\_STATUS\_WRONG\_MSG\_LENGTH ((CFE\_Status\_t)0xc8000002)

*Wrong Message Length.*

- #define CFE\_STATUS\_UNKNOWN\_MSG\_ID ((CFE\_Status\_t)0xc8000003)  
*Unknown Message ID.*
- #define CFE\_STATUS\_BAD\_COMMAND\_CODE ((CFE\_Status\_t)0xc8000004)  
*Bad Command Code.*
- #define CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL ((CFE\_Status\_t)0xc8000005)  
*External failure.*
- #define CFE\_STATUS\_REQUEST\_ALREADY\_PENDING ((int32)0xc8000006)  
*Request already pending.*
- #define CFE\_STATUS\_VALIDATION\_FAILURE ((int32)0xc8000007)  
*Request or input value failed basic structural validation.*
- #define CFE\_STATUS\_RANGE\_ERROR ((int32)0xc8000008)  
*Request or input value is out of range.*
- #define CFE\_STATUS\_INCORRECT\_STATE ((int32)0xc8000009)  
*Cannot process request at this time.*
- #define CFE\_STATUS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc800ffff)  
*Not Implemented.*
- #define CFE\_EVS\_UNKNOWN\_FILTER ((CFE\_Status\_t)0xc2000001)  
*Unknown Filter.*
- #define CFE\_EVS\_APP\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000002)  
*Application Not Registered.*
- #define CFE\_EVS\_APP\_ILLEGAL\_APP\_ID ((CFE\_Status\_t)0xc2000003)  
*Illegal Application ID.*
- #define CFE\_EVS\_APP\_FILTER\_OVERLOAD ((CFE\_Status\_t)0xc2000004)  
*Application Filter Overload.*
- #define CFE\_EVS\_RESET\_AREA\_POINTER ((CFE\_Status\_t)0xc2000005)  
*Reset Area Pointer Failure.*
- #define CFE\_EVS\_EVT\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000006)  
*Event Not Registered.*
- #define CFE\_EVS\_FILE\_WRITE\_ERROR ((CFE\_Status\_t)0xc2000007)  
*File Write Error.*
- #define CFE\_EVS\_INVALID\_PARAMETER ((CFE\_Status\_t)0xc2000008)  
*Invalid Pointer.*
- #define CFE\_EVS\_APP\_SQUELCHED ((CFE\_Status\_t)0xc2000009)  
*Event squelched.*
- #define CFE\_EVS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc200ffff)  
*Not Implemented.*
- #define CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID ((CFE\_Status\_t)0xc4000001)  
*Resource ID is not valid.*
- #define CFE\_ES\_ERR\_NAME\_NOT\_FOUND ((CFE\_Status\_t)0xc4000002)  
*Resource Name Error.*
- #define CFE\_ES\_ERR\_APP\_CREATE ((CFE\_Status\_t)0xc4000004)  
*Application Create Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_CREATE ((CFE\_Status\_t)0xc4000005)  
*Child Task Create Error.*
- #define CFE\_ES\_ERR\_SYS\_LOG\_FULL ((CFE\_Status\_t)0xc4000006)  
*System Log Full.*

- #define CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE ((CFE\_Status\_t)0xc4000008)  
*Memory Block Size Error.*
- #define CFE\_ES\_ERR\_LOAD\_LIB ((CFE\_Status\_t)0xc4000009)  
*Load Library Error.*
- #define CFE\_ES\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc400000a)  
*Bad Argument.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER ((CFE\_Status\_t)0xc400000b)  
*Child Task Register Error.*
- #define CFE\_ES\_CDS\_ALREADY\_EXISTS ((CFE\_Status\_t)0x4400000d)  
*CDS Already Exists.*
- #define CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY ((CFE\_Status\_t)0xc400000e)  
*CDS Insufficient Memory.*
- #define CFE\_ES\_CDS\_INVALID\_NAME ((CFE\_Status\_t)0xc400000f)  
*CDS Invalid Name.*
- #define CFE\_ES\_CDS\_INVALID\_SIZE ((CFE\_Status\_t)0xc4000010)  
*CDS Invalid Size.*
- #define CFE\_ES\_CDS\_INVALID ((CFE\_Status\_t)0xc4000012)  
*CDS Invalid.*
- #define CFE\_ES\_CDS\_ACCESS\_ERROR ((CFE\_Status\_t)0xc4000013)  
*CDS Access Error.*
- #define CFE\_ES\_FILE\_IO\_ERR ((CFE\_Status\_t)0xc4000014)  
*File IO Error.*
- #define CFE\_ES\_RST\_ACCESS\_ERR ((CFE\_Status\_t)0xc4000015)  
*Reset Area Access Error.*
- #define CFE\_ES\_ERR\_APP\_REGISTER ((CFE\_Status\_t)0xc4000017)  
*Application Register Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE ((CFE\_Status\_t)0xc4000018)  
*Child Task Delete Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK ((CFE\_Status\_t)0xc4000019)  
*Child Task Delete Passed Main Task.*
- #define CFE\_ES\_CDS\_BLOCK\_CRC\_ERR ((CFE\_Status\_t)0xc400001A)  
*CDS Block CRC Error.*
- #define CFE\_ES\_MUT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001B)  
*Mutex Semaphore Delete Error.*
- #define CFE\_ES\_BIN\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001C)  
*Binary Semaphore Delete Error.*
- #define CFE\_ES\_COUNT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001D)  
*Counting Semaphore Delete Error.*
- #define CFE\_ES\_QUEUE\_DELETE\_ERR ((CFE\_Status\_t)0xc400001E)  
*Queue Delete Error.*
- #define CFE\_ES\_FILE\_CLOSE\_ERR ((CFE\_Status\_t)0xc400001F)  
*File Close Error.*
- #define CFE\_ES\_CDS\_WRONG\_TYPE\_ERR ((CFE\_Status\_t)0xc4000020)  
*CDS Wrong Type Error.*
- #define CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR ((CFE\_Status\_t)0xc4000022)  
*CDS Owner Active Error.*
- #define CFE\_ES\_APP\_CLEANUP\_ERR ((CFE\_Status\_t)0xc4000023)

- Application Cleanup Error.*
- #define CFE\_ES\_TIMER\_DELETE\_ERR ((CFE\_Status\_t)0xc4000024)  
*Timer Delete Error.*
  - #define CFE\_ES\_BUFFER\_NOT\_IN\_POOL ((CFE\_Status\_t)0xc4000025)  
*Buffer Not In Pool.*
  - #define CFE\_ES\_TASK\_DELETE\_ERR ((CFE\_Status\_t)0xc4000026)  
*Task Delete Error.*
  - #define CFE\_ES\_OPERATION\_TIMED\_OUT ((CFE\_Status\_t)0xc4000027)  
*Operation Timed Out.*
  - #define CFE\_ES\_LIB\_ALREADY\_LOADED ((CFE\_Status\_t)0x44000028)  
*Library Already Loaded.*
  - #define CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED ((CFE\_Status\_t)0x44000029)  
*System Log Message Truncated.*
  - #define CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE ((CFE\_Status\_t)0xc400002B)  
*Resource ID is not available.*
  - #define CFE\_ES\_POOL\_BLOCK\_INVALID ((CFE\_Status\_t)0xc400002C)  
*Invalid pool block.*
  - #define CFE\_ES\_ERR\_DUPLICATE\_NAME ((CFE\_Status\_t)0xc400002E)  
*Duplicate Name Error.*
  - #define CFE\_ES\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc400ffff)  
*Not Implemented.*
  - #define CFE\_FS\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc6000001)  
*Bad Argument.*
  - #define CFE\_FS\_INVALID\_PATH ((CFE\_Status\_t)0xc6000002)  
*Invalid Path.*
  - #define CFE\_FS\_FNAME\_TOO\_LONG ((CFE\_Status\_t)0xc6000003)  
*Filename Too Long.*
  - #define CFE\_FS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc600ffff)  
*Not Implemented.*
  - #define CFE\_SB\_TIME\_OUT ((CFE\_Status\_t)0xca000001)  
*Time Out.*
  - #define CFE\_SB\_NO\_MESSAGE ((CFE\_Status\_t)0xca000002)  
*No Message.*
  - #define CFE\_SB\_BAD\_ARGUMENT ((CFE\_Status\_t)0xca000003)  
*Bad Argument.*
  - #define CFE\_SB\_MAX\_PIPES\_MET ((CFE\_Status\_t)0xca000004)  
*Max Pipes Met.*
  - #define CFE\_SB\_PIPE\_CR\_ERR ((CFE\_Status\_t)0xca000005)  
*Pipe Create Error.*
  - #define CFE\_SB\_PIPE\_RD\_ERR ((CFE\_Status\_t)0xca000006)  
*Pipe Read Error.*
  - #define CFE\_SB\_MSG\_TOO\_BIG ((CFE\_Status\_t)0xca000007)  
*Message Too Big.*
  - #define CFE\_SB\_BUF\_ALOC\_ERR ((CFE\_Status\_t)0xca000008)  
*Buffer Allocation Error.*
  - #define CFE\_SB\_MAX\_MSGS\_MET ((CFE\_Status\_t)0xca000009)  
*Max Messages Met.*

- #define CFE\_SB\_MAX\_DESTS\_MET ((CFE\_Status\_t)0xca00000a)  
*Max Destinations Met.*
- #define CFE\_SB\_INTERNAL\_ERR ((CFE\_Status\_t)0xca00000c)  
*Internal Error.*
- #define CFE\_SB\_WRONG\_MSG\_TYPE ((CFE\_Status\_t)0xca00000d)  
*Wrong Message Type.*
- #define CFE\_SB\_BUFFER\_INVALID ((CFE\_Status\_t)0xca00000e)  
*Buffer Invalid.*
- #define CFE\_SB\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xca00ffff)  
*Not Implemented.*
- #define CFE\_TBL\_ERR\_INVALID\_HANDLE ((CFE\_Status\_t)0xcc000001)  
*Invalid Handle.*
- #define CFE\_TBL\_ERR\_INVALID\_NAME ((CFE\_Status\_t)0xcc000002)  
*Invalid Name.*
- #define CFE\_TBL\_ERR\_INVALID\_SIZE ((CFE\_Status\_t)0xcc000003)  
*Invalid Size.*
- #define CFE\_TBL\_INFO\_UPDATE\_PENDING ((CFE\_Status\_t)0x4c000004)  
*Update Pending.*
- #define CFE\_TBL\_ERR\_NEVER\_LOADED ((CFE\_Status\_t)0xcc000005)  
*Never Loaded.*
- #define CFE\_TBL\_ERR\_REGISTRY\_FULL ((CFE\_Status\_t)0xcc000006)  
*Registry Full.*
- #define CFE\_TBL\_WARN\_DUPLICATE ((CFE\_Status\_t)0x4c000007)  
*Duplicate Warning.*
- #define CFE\_TBL\_ERR\_NO\_ACCESS ((CFE\_Status\_t)0xcc000008)  
*No Access.*
- #define CFE\_TBL\_ERR\_UNREGISTERED ((CFE\_Status\_t)0xcc000009)  
*Unregistered.*
- #define CFE\_TBL\_ERR\_HANDLES\_FULL ((CFE\_Status\_t)0xcc00000B)  
*Handles Full.*
- #define CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE ((CFE\_Status\_t)0xcc00000C)  
*Duplicate Table With Different Size.*
- #define CFE\_TBL\_ERR\_DUPLICATE\_NOT\_OWNED ((CFE\_Status\_t)0xcc00000D)  
*Duplicate Table And Not Owned.*
- #define CFE\_TBL\_INFO\_UPDATED ((CFE\_Status\_t)0x4c00000E)  
*Updated.*
- #define CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL ((CFE\_Status\_t)0xcc00000F)  
*No Buffer Available.*
- #define CFE\_TBL\_ERR\_DUMP\_ONLY ((CFE\_Status\_t)0xcc000010)  
*Dump Only Error.*
- #define CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE ((CFE\_Status\_t)0xcc000011)  
*Illegal Source Type.*
- #define CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS ((CFE\_Status\_t)0xcc000012)  
*Load In Progress.*
- #define CFE\_TBL\_ERR\_FILE\_TOO\_LARGE ((CFE\_Status\_t)0xcc000014)  
*File Too Large.*
- #define CFE\_TBL\_WARN\_SHORT\_FILE ((CFE\_Status\_t)0x4c000015)

*Short File Warning.*

- #define CFE\_TBL\_ERR\_BAD\_CONTENT\_ID ((CFE\_Status\_t)0xcc000016)  
*Bad Content ID.*
- #define CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING ((CFE\_Status\_t)0x4c000017)  
*No Update Pending.*
- #define CFE\_TBL\_INFO\_TABLE\_LOCKED ((CFE\_Status\_t)0x4c000018)  
*Table Locked.*
- #define CFE\_TBL\_INFO\_VALIDATION\_PENDING ((CFE\_Status\_t)0x4c000019)
- #define CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING ((CFE\_Status\_t)0x4c00001A)
- #define CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID ((CFE\_Status\_t)0xcc00001B)  
*Bad Subtype ID.*
- #define CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT ((CFE\_Status\_t)0xcc00001C)  
*File Size Inconsistent.*
- #define CFE\_TBL\_ERR\_NO\_STD\_HEADER ((CFE\_Status\_t)0xcc00001D)  
*No Standard Header.*
- #define CFE\_TBL\_ERR\_NO\_TBL\_HEADER ((CFE\_Status\_t)0xcc00001E)  
*No Table Header.*
- #define CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG ((CFE\_Status\_t)0xcc00001F)  
*Filename Too Long.*
- #define CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE ((CFE\_Status\_t)0xcc000020)  
*File For Wrong Table.*
- #define CFE\_TBL\_ERR\_LOAD\_INCOMPLETE ((CFE\_Status\_t)0xcc000021)  
*Load Incomplete.*
- #define CFE\_TBL\_WARN\_PARTIAL\_LOAD ((CFE\_Status\_t)0x4c000022)  
*Partial Load Warning.*
- #define CFE\_TBL\_ERR\_PARTIAL\_LOAD ((CFE\_Status\_t)0xcc000023)  
*Partial Load Error.*
- #define CFE\_TBL\_INFO\_DUMP\_PENDING ((CFE\_Status\_t)0x4c000024)  
*Dump Pending.*
- #define CFE\_TBL\_ERR\_INVALID\_OPTIONS ((CFE\_Status\_t)0xcc000025)  
*Invalid Options.*
- #define CFE\_TBL\_WARN\_NOT\_CRITICAL ((CFE\_Status\_t)0x4c000026)  
*Not Critical Warning.*
- #define CFE\_TBL\_INFO\_RECOVERED\_TBL ((CFE\_Status\_t)0x4c000027)  
*Recovered Table.*
- #define CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID ((CFE\_Status\_t)0xcc000028)  
*Bad Spacecraft ID.*
- #define CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID ((CFE\_Status\_t)0xcc000029)  
*Bad Processor ID.*
- #define CFE\_TBL\_MESSAGE\_ERROR ((CFE\_Status\_t)0xcc00002a)  
*Message Error.*
- #define CFE\_TBL\_ERR\_SHORT\_FILE ((CFE\_Status\_t)0xcc00002b)
- #define CFE\_TBL\_ERR\_ACCESS ((CFE\_Status\_t)0xcc00002c)
- #define CFE\_TBL\_BAD\_ARGUMENT ((CFE\_Status\_t)0xcc00002d)  
*Bad Argument.*
- #define CFE\_TBL\_INFO\_NO\_DUMP\_PENDING ((CFE\_Status\_t)0x4c00002e)  
*No Dump Pending.*

- #define CFE\_TBL\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xcc00ffff)  
*Not Implemented.*
- #define CFE\_TIME\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xce00ffff)  
*Not Implemented.*
- #define CFE\_TIME\_INTERNAL\_ONLY ((CFE\_Status\_t)0xce000001)  
*Internal Only.*
- #define CFE\_TIME\_OUT\_OF\_RANGE ((CFE\_Status\_t)0xce000002)  
*Out Of Range.*
- #define CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS ((CFE\_Status\_t)0xce000003)  
*Too Many Sync Callbacks.*
- #define CFE\_TIME\_CALLBACK\_NOT\_REGISTERED ((CFE\_Status\_t)0xce000004)  
*Callback Not Registered.*
- #define CFE\_TIME\_BAD\_ARGUMENT ((CFE\_Status\_t)0xce000005)  
*Bad Argument.*

### 10.11.1 Detailed Description

### 10.11.2 Macro Definition Documentation

**10.11.2.1 CFE\_ES\_APP\_CLEANUP\_ERR** #define CFE\_ES\_APP\_CLEANUP\_ERR ((CFE\_Status\_t)0xc4000023)  
Application Cleanup Error.

Occurs when an attempt was made to Clean Up an application which involves calling Table, EVS, and SB cleanup functions, then deleting all ES resources, child tasks, and unloading the object module. The approach here is to keep going even though one of these steps had an error. There will be syslog messages detailing each problem.

Definition at line 588 of file cfe\_error.h.

**10.11.2.2 CFE\_ES\_BAD\_ARGUMENT** #define CFE\_ES\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc400000a)  
Bad Argument.

Bad parameter passed into an ES API.

Definition at line 399 of file cfe\_error.h.

**10.11.2.3 CFE\_ES\_BIN\_SEM\_DELETE\_ERR** #define CFE\_ES\_BIN\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001C)  
Binary Semaphore Delete Error.

Occurs when trying to delete a Binary Semaphore that belongs to a task that ES is cleaning up.

Definition at line 527 of file cfe\_error.h.

**10.11.2.4 CFE\_ES\_BUFFER\_NOT\_IN\_POOL** #define CFE\_ES\_BUFFER\_NOT\_IN\_POOL ((CFE\_Status\_t)0xc4000025)  
Buffer Not In Pool.

The specified address is not in the memory pool.

Definition at line 605 of file cfe\_error.h.

**10.11.2.5 CFE\_ES\_CDS\_ACCESS\_ERROR** #define CFE\_ES\_CDS\_ACCESS\_ERROR ((CFE\_Status\_t)0xc4000013)  
CDS Access Error.

The CDS was inaccessible

Definition at line 458 of file cfe\_error.h.

**10.11.2.6 CFE\_ES\_CDS\_ALREADY\_EXISTS** #define CFE\_ES\_CDS\_ALREADY\_EXISTS ((CFE\_Status\_t)0x4400000d)  
CDS Already Exists.

The Application is receiving the pointer to a CDS that was already present.

Definition at line 415 of file cfe\_error.h.

**10.11.2.7 CFE\_ES\_CDS\_BLOCK\_CRC\_ERR** #define CFE\_ES\_CDS\_BLOCK\_CRC\_ERR ((CFE\_Status\_t)0xc400001A)  
CDS Block CRC Error.

Occurs when trying to read a CDS Data block and the CRC of the current data does not match the stored CRC for the data. Either the contents of the CDS Data Block are corrupted or the CDS Control Block is corrupted.

Definition at line 509 of file cfe\_error.h.

**10.11.2.8 CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY** #define CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY ((CFE\_Status\_t)0xc400000e)  
CDS Insufficient Memory.

The Application is requesting a CDS Block that is larger than the remaining CDS memory.

Definition at line 424 of file cfe\_error.h.

**10.11.2.9 CFE\_ES\_CDS\_INVALID** #define CFE\_ES\_CDS\_INVALID ((CFE\_Status\_t)0xc4000012)  
CDS Invalid.

The CDS contents are invalid.

Definition at line 450 of file cfe\_error.h.

**10.11.2.10 CFE\_ES\_CDS\_INVALID\_NAME** #define CFE\_ES\_CDS\_INVALID\_NAME ((CFE\_Status\_t)0xc400000f)  
CDS Invalid Name.

The Application is requesting a CDS Block with an invalid ASCII string name. Either the name is too long (> CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH) or was an empty string.

Definition at line 433 of file cfe\_error.h.

**10.11.2.11 CFE\_ES\_CDS\_INVALID\_SIZE** #define CFE\_ES\_CDS\_INVALID\_SIZE ((CFE\_Status\_t)0xc4000010)  
CDS Invalid Size.

The Application is requesting a CDS Block or Pool with a size beyond the applicable limits, either too large or too small/zero.

Definition at line 442 of file cfe\_error.h.

**10.11.2.12 CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR** #define CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR ((CFE\_Status\_t)0xc4000022)  
CDS Owner Active Error.

Occurs when an attempt was made to delete a CDS when an application with the same name associated with the CDS is still present. CDSs can ONLY be deleted when Applications that created them are not present in the system.

Definition at line 575 of file cfe\_error.h.

**10.11.2.13 CFE\_ES\_CDS\_WRONG\_TYPE\_ERR** #define CFE\_ES\_CDS\_WRONG\_TYPE\_ERR ((CFE\_Status\_t)0xc4000020)  
CDS Wrong Type Error.

Occurs when Table Services is trying to delete a Critical Data Store that is not a Critical Table Image or when Executive Services is trying to delete a Critical Table Image.

Definition at line 564 of file cfe\_error.h.

**10.11.2.14 CFE\_ES\_COUNT\_SEM\_DELETE\_ERR** #define CFE\_ES\_COUNT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001D)  
Counting Semaphore Delete Error.  
Occurs when trying to delete a Counting Semaphore that belongs to a task that ES is cleaning up.  
Definition at line 536 of file cfe\_error.h.

**10.11.2.15 CFE\_ES\_ERR\_APP\_CREATE** #define CFE\_ES\_ERR\_APP\_CREATE ((CFE\_Status\_t)0xc4000004)  
Application Create Error.  
There was an error loading or creating the App.  
Definition at line 358 of file cfe\_error.h.

**10.11.2.16 CFE\_ES\_ERR\_APP\_REGISTER** #define CFE\_ES\_ERR\_APP\_REGISTER ((CFE\_Status\_t)0xc4000017)  
Application Register Error.  
Occurs when a task cannot be registered in ES global tables  
Definition at line 482 of file cfe\_error.h.

**10.11.2.17 CFE\_ES\_ERR\_CHILD\_TASK\_CREATE** #define CFE\_ES\_ERR\_CHILD\_TASK\_CREATE ((CFE\_Status\_t)0xc4000005)  
Child Task Create Error.  
There was an error creating a child task.  
Definition at line 366 of file cfe\_error.h.

**10.11.2.18 CFE\_ES\_ERR\_CHILD\_TASK\_DELETE** #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE ((CFE\_Status\_t)0xc4000018)  
Child Task Delete Error.  
There was an error deleting a child task.  
Definition at line 490 of file cfe\_error.h.

**10.11.2.19 CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK** #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_↔  
MAIN\_TASK ((CFE\_Status\_t)0xc4000019)  
Child Task Delete Passed Main Task.  
There was an attempt to delete a cFE App Main Task with the [CFE\\_ES\\_DeleteChildTask](#) API.  
Definition at line 499 of file cfe\_error.h.

**10.11.2.20 CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER** #define CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER ((CFE\_Status\_t)0xc400000b)  
Child Task Register Error.  
Errors occurred when trying to register a child task.  
Definition at line 407 of file cfe\_error.h.

**10.11.2.21 CFE\_ES\_ERR\_DUPLICATE\_NAME** #define CFE\_ES\_ERR\_DUPLICATE\_NAME ((CFE\_Status\_t)0xc400002E)  
Duplicate Name Error.  
Resource creation failed due to the name already existing in the system.  
Definition at line 668 of file cfe\_error.h.

**10.11.2.22 CFE\_ES\_ERR\_LOAD\_LIB** #define CFE\_ES\_ERR\_LOAD\_LIB ((CFE\_Status\_t)0xc4000009)  
Load Library Error.  
Could not load the shared library.

Definition at line 391 of file cfe\_error.h.

**10.11.2.23 CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE** #define CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE ((CFE\_Status\_t)0xc4000008)  
Memory Block Size Error.

The block size requested is invalid.

Definition at line 383 of file cfe\_error.h.

**10.11.2.24 CFE\_ES\_ERR\_NAME\_NOT\_FOUND** #define CFE\_ES\_ERR\_NAME\_NOT\_FOUND ((CFE\_Status\_t)0xc4000002)  
Resource Name Error.

There is no match in the system for the given name.

Definition at line 350 of file cfe\_error.h.

**10.11.2.25 CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID** #define CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID ((CFE\_Status\_t)0xc4000000)  
Resource ID is not valid.

This error indicates that the passed in resource identifier (App ID, Lib ID, Counter ID, etc) did not validate.

Definition at line 342 of file cfe\_error.h.

**10.11.2.26 CFE\_ES\_ERR\_SYS\_LOG\_FULL** #define CFE\_ES\_ERR\_SYS\_LOG\_FULL ((CFE\_Status\_t)0xc4000006)  
System Log Full.

The cFE system Log is full. This error means the message was not logged at all

Definition at line 375 of file cfe\_error.h.

**10.11.2.27 CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED** #define CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED ((CFE\_Status\_t)0x44000029)  
System Log Message Truncated.

This information code means the last syslog message was truncated due to insufficient space in the log buffer.

Definition at line 640 of file cfe\_error.h.

**10.11.2.28 CFE\_ES\_FILE\_CLOSE\_ERR** #define CFE\_ES\_FILE\_CLOSE\_ERR ((CFE\_Status\_t)0xc400001F)  
File Close Error.

Occurs when trying to close a file that belongs to a task that ES is cleaning up.

Definition at line 554 of file cfe\_error.h.

**10.11.2.29 CFE\_ES\_FILE\_IO\_ERR** #define CFE\_ES\_FILE\_IO\_ERR ((CFE\_Status\_t)0xc4000014)  
File IO Error.

Occurs when a file operation fails

Definition at line 466 of file cfe\_error.h.

**10.11.2.30 CFE\_ES\_LIB\_ALREADY\_LOADED** #define CFE\_ES\_LIB\_ALREADY\_LOADED ((CFE\_Status\_t)0x44000028)  
Library Already Loaded.

Occurs if CFE\_ES\_LoadLibrary detects that the requested library name is already loaded.

Definition at line 631 of file cfe\_error.h.

**10.11.2.31 CFE\_ES\_MUT\_SEM\_DELETE\_ERR** #define CFE\_ES\_MUT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001B)  
Mutex Semaphore Delete Error.

Occurs when trying to delete a Mutex that belongs to a task that ES is cleaning up.

Definition at line 518 of file cfe\_error.h.

**10.11.2.32 CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE** #define CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE ((CFE\_Status\_t)0xc400002A)  
Resource ID is not available.

This error indicates that the maximum resource identifiers (App ID, Lib ID, Counter ID, etc) has already been reached and a new ID cannot be allocated.

Definition at line 650 of file cfe\_error.h.

**10.11.2.33 CFE\_ES\_NOT\_IMPLEMENTED** #define CFE\_ES\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc400ffff)  
Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 679 of file cfe\_error.h.

**10.11.2.34 CFE\_ES\_OPERATION\_TIMED\_OUT** #define CFE\_ES\_OPERATION\_TIMED\_OUT ((CFE\_Status\_t)0xc4000027)  
Operation Timed Out.

Occurs if the timeout for a given operation was exceeded

Definition at line 622 of file cfe\_error.h.

**10.11.2.35 CFE\_ES\_POOL\_BLOCK\_INVALID** #define CFE\_ES\_POOL\_BLOCK\_INVALID ((CFE\_Status\_t)0xc400002C)  
Invalid pool block.

Software attempted to "put" a block back into a pool which does not appear to belong to that pool. This may mean the pool has become unusable due to memory corruption.

Definition at line 660 of file cfe\_error.h.

**10.11.2.36 CFE\_ES\_QUEUE\_DELETE\_ERR** #define CFE\_ES\_QUEUE\_DELETE\_ERR ((CFE\_Status\_t)0xc400001E)  
Queue Delete Error.

Occurs when trying to delete a Queue that belongs to a task that ES is cleaning up.

Definition at line 545 of file cfe\_error.h.

**10.11.2.37 CFE\_ES\_RST\_ACCESS\_ERR** #define CFE\_ES\_RST\_ACCESS\_ERR ((CFE\_Status\_t)0xc4000015)  
Reset Area Access Error.

Occurs when the BSP is not successful in returning the reset area address.

Definition at line 474 of file cfe\_error.h.

**10.11.2.38 CFE\_ES\_TASK\_DELETE\_ERR** #define CFE\_ES\_TASK\_DELETE\_ERR ((CFE\_Status\_t)0xc4000026)  
Task Delete Error.

Occurs when trying to delete a task that ES is cleaning up.

Definition at line 614 of file cfe\_error.h.

**10.11.2.39 CFE\_ES\_TIMER\_DELETE\_ERR** #define CFE\_ES\_TIMER\_DELETE\_ERR ((CFE\_Status\_t)0xc4000024)  
Timer Delete Error.

Occurs when trying to delete a Timer that belongs to a task that ES is cleaning up.

Definition at line 597 of file cfe\_error.h.

**10.11.2.40 CFE\_EVS\_APP\_FILTER\_OVERLOAD** #define CFE\_EVS\_APP\_FILTER\_OVERLOAD ((CFE\_Status\_t)0xc2000004)  
Application Filter Overload.

Number of Application event filters input upon registration is greater than CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS  
Definition at line 276 of file cfe\_error.h.

**10.11.2.41 CFE\_EVS\_APP\_ILLEGAL\_APP\_ID** #define CFE\_EVS\_APP\_ILLEGAL\_APP\_ID ((CFE\_Status\_t)0xc2000003)  
Illegal Application ID.

Application ID returned by CFE\_ES\_GetAppIDByName is greater than CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS

Definition at line 267 of file cfe\_error.h.

**10.11.2.42 CFE\_EVS\_APP\_NOT\_REGISTERED** #define CFE\_EVS\_APP\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000002)

Application Not Registered.

Calling application never previously called CFE\_EVS\_Register

Definition at line 258 of file cfe\_error.h.

**10.11.2.43 CFE\_EVS\_APP\_SQUELCHED** #define CFE\_EVS\_APP\_SQUELCHED ((CFE\_Status\_t)0xc2000009)

Event squelched.

Event squelched due to being sent at too high a rate

Definition at line 318 of file cfe\_error.h.

**10.11.2.44 CFE\_EVS\_EVT\_NOT\_REGISTERED** #define CFE\_EVS\_EVT\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000006)

Event Not Registered.

CFE\_EVS\_ResetFilter EventID argument was not found in any event filter registered by the calling application.

Definition at line 294 of file cfe\_error.h.

**10.11.2.45 CFE\_EVS\_FILE\_WRITE\_ERROR** #define CFE\_EVS\_FILE\_WRITE\_ERROR ((CFE\_Status\_t)0xc2000007)

File Write Error.

A file write error occurred while processing an EVS command

Definition at line 302 of file cfe\_error.h.

**10.11.2.46 CFE\_EVS\_INVALID\_PARAMETER** #define CFE\_EVS\_INVALID\_PARAMETER ((CFE\_Status\_t)0xc2000008)

Invalid Pointer.

Invalid parameter supplied to EVS command

Definition at line 310 of file cfe\_error.h.

**10.11.2.47 CFE\_EVS\_NOT\_IMPLEMENTED** #define CFE\_EVS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc200ffff)

Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 329 of file cfe\_error.h.

**10.11.2.48 CFE\_EVS\_RESET\_AREA\_POINTER** #define CFE\_EVS\_RESET\_AREA\_POINTER ((CFE\_Status\_t)0xc2000005)  
Reset Area Pointer Failure.

Could not get pointer to the ES Reset area, so we could not get the pointer to the EVS Log.

Definition at line 285 of file cfe\_error.h.

**10.11.2.49 CFE\_EVS\_UNKNOWN\_FILTER** #define CFE\_EVS\_UNKNOWN\_FILTER ((CFE\_Status\_t)0xc2000001)  
Unknown Filter.

**CFE\_EVS\_Register** FilterScheme parameter was illegal

Definition at line 250 of file cfe\_error.h.

**10.11.2.50 CFE\_FS\_BAD\_ARGUMENT** #define CFE\_FS\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc6000001)  
Bad Argument.

A parameter given by a caller to a File Services API did not pass validation checks.

Definition at line 692 of file cfe\_error.h.

**10.11.2.51 CFE\_FS\_FNAME\_TOO\_LONG** #define CFE\_FS\_FNAME\_TOO\_LONG ((CFE\_Status\_t)0xc6000003)  
Filename Too Long.

FS filename string is too long

Definition at line 708 of file cfe\_error.h.

**10.11.2.52 CFE\_FS\_INVALID\_PATH** #define CFE\_FS\_INVALID\_PATH ((CFE\_Status\_t)0xc6000002)  
Invalid Path.

FS was unable to extract a filename from a path string

Definition at line 700 of file cfe\_error.h.

**10.11.2.53 CFE\_FS\_NOT\_IMPLEMENTED** #define CFE\_FS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc600ffff)  
Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 719 of file cfe\_error.h.

**10.11.2.54 CFE\_SB\_BAD\_ARGUMENT** #define CFE\_SB\_BAD\_ARGUMENT ((CFE\_Status\_t)0xca000003)  
Bad Argument.

A parameter given by a caller to a Software Bus API did not pass validation checks.

Definition at line 750 of file cfe\_error.h.

**10.11.2.55 CFE\_SB\_BUF\_ALOC\_ERR** #define CFE\_SB\_BUF\_ALOC\_ERR ((CFE\_Status\_t)0xca000008)  
Buffer Allocation Error.

Returned when the memory in the SB message buffer pool has been depleted. The amount of memory in the pool is dictated by the configuration parameter **CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES** specified in the cfe\_platform\_cfg.h file. Also the memory statistics, including current utilization figures and high water marks for the SB Buffer memory pool can be monitored by sending a Software Bus command to send the SB statistics packet.

Definition at line 808 of file cfe\_error.h.

**10.11.2.56 CFE\_SB\_BUFFER\_INVALID** #define CFE\_SB\_BUFFER\_INVALID ((CFE\_Status\_t)0xca00000e)

Buffer Invalid.

This error code will be returned when a request to release or send a zero copy buffer is invalid, such as if the handle or buffer is not correct or the buffer was previously released.

Definition at line 859 of file cfe\_error.h.

**10.11.2.57 CFE\_SB\_INTERNAL\_ERR** #define CFE\_SB\_INTERNAL\_ERR ((CFE\_Status\_t)0xca00000c)

Internal Error.

This error code will be returned by the [CFE\\_SB\\_Subscribe](#) API if the code detects an internal index is out of range. The most likely cause would be a Single Event Upset.

Definition at line 840 of file cfe\_error.h.

**10.11.2.58 CFE\_SB\_MAX\_DESTS\_MET** #define CFE\_SB\_MAX\_DESTS\_MET ((CFE\_Status\_t)0xca00000a)

Max Destinations Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate another destination for a particular the given message ID. This occurs when the number of destinations in use meets the platform configuration parameter [CFE\\_PLATFORM\\_SB\\_MAX\\_DEST\\_PER\\_PKT](#).

Definition at line 830 of file cfe\_error.h.

**10.11.2.59 CFE\_SB\_MAX\_MSGS\_MET** #define CFE\_SB\_MAX\_MSGS\_MET ((CFE\_Status\_t)0xca000009)

Max Messages Met.

Will be returned when calling one of the SB subscription API's if the SB routing table cannot accommodate another unique message ID because the platform configuration parameter [CFE\\_PLATFORM\\_SB\\_MAX\\_MSG\\_IDS](#) has been met.

Definition at line 818 of file cfe\_error.h.

**10.11.2.60 CFE\_SB\_MAX\_PIPES\_MET** #define CFE\_SB\_MAX\_PIPES\_MET ((CFE\_Status\_t)0xca000004)

Max Pipes Met.

This error code will be returned from [CFE\\_SB\\_CreatePipe](#) when the SB cannot accommodate the request to create a pipe because the maximum number of pipes ([CFE\\_PLATFORM\\_SB\\_MAX\\_PIPES](#)) are in use. This configuration parameter is defined in the cfe\_platform\_cfg.h file.

Definition at line 761 of file cfe\_error.h.

**10.11.2.61 CFE\_SB\_MSG\_TOO\_BIG** #define CFE\_SB\_MSG\_TOO\_BIG ((CFE\_Status\_t)0xca000007)

Message Too Big.

The size field in the message header indicates the message exceeds the max Software Bus message size. The max size is defined by configuration parameter [CFE\\_MISSION\\_SB\\_MAX\\_SB\\_MSG\\_SIZE](#) in cfe\_mission\_cfg.h

Definition at line 795 of file cfe\_error.h.

**10.11.2.62 CFE\_SB\_NO\_MESSAGE** #define CFE\_SB\_NO\_MESSAGE ((CFE\_Status\_t)0xca000002)

No Message.

When "Polling" a pipe for a message in [CFE\\_SB\\_ReceiveBuffer](#), this return value indicates that there was not a message on the pipe.

Definition at line 741 of file cfe\_error.h.

**10.11.2.63 CFE\_SB\_NOT\_IMPLEMENTED** #define CFE\_SB\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xca00ffff)

Not Implemented.

Current version of cFE does not have the function or the feature of the function implemented. This could be due to either an early build of the cFE for this platform or the platform does not support the specified feature.

Definition at line 870 of file cfe\_error.h.

**10.11.2.64 CFE\_SB\_PIPE\_CR\_ERR** #define CFE\_SB\_PIPE\_CR\_ERR ((CFE\_Status\_t)0xca000005)

Pipe Create Error.

The maximum number of queues([OS\\_MAX\\_QUEUES](#)) are in use. Or possibly a lower level problem with creating the underlying queue has occurred such as a lack of memory. If the latter is the problem, the status code displayed in the event must be tracked.

Definition at line 772 of file cfe\_error.h.

**10.11.2.65 CFE\_SB\_PIPE\_RD\_ERR** #define CFE\_SB\_PIPE\_RD\_ERR ((CFE\_Status\_t)0xca000006)

Pipe Read Error.

This return value indicates an error at the Queue read level. This error typically cannot be corrected by the caller. Some possible causes are: queue was not properly initialized or created, the number of bytes read from the queue was not the number of bytes requested in the read. The queue id is invalid. Similar errors regarding the pipe will be caught by higher level code in the Software Bus.

Definition at line 785 of file cfe\_error.h.

**10.11.2.66 CFE\_SB\_TIME\_OUT** #define CFE\_SB\_TIME\_OUT ((CFE\_Status\_t)0xca000001)

Time Out.

In [CFE\\_SB\\_ReceiveBuffer](#), this return value indicates that a packet has not been received in the time given in the "timeout" parameter.

Definition at line 732 of file cfe\_error.h.

**10.11.2.67 CFE\_SB\_WRONG\_MSG\_TYPE** #define CFE\_SB\_WRONG\_MSG\_TYPE ((CFE\_Status\_t)0xca00000d)

Wrong Message Type.

This error code will be returned when a request such as [CFE\\_MSG\\_SetMsgTime](#) is made on a packet that does not include a field for msg time.

Definition at line 849 of file cfe\_error.h.

**10.11.2.68 CFE\_STATUS\_BAD\_COMMAND\_CODE** #define CFE\_STATUS\_BAD\_COMMAND\_CODE ((CFE\_Status\_t)0xc8000004)

Bad Command Code.

This error code will be returned when a message identification process determined that the command code is does not correspond to any known value

Definition at line 182 of file cfe\_error.h.

**10.11.2.69 CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL** #define CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL ((CFE\_Status\_t)0xc8000005)

External failure.

This error indicates that the operation failed for some reason outside the scope of CFE. The real failure may have been in OSAL, PSP, or another dependent library.

Details of the original failure should be written to syslog and/or a system event before returning this error.  
Definition at line 194 of file cfe\_error.h.

**10.11.2.70 CFE\_STATUS\_INCORRECT\_STATE** #define CFE\_STATUS\_INCORRECT\_STATE ((int32)0xc8000009)  
Cannot process request at this time.

The system is not currently in the correct state to accept the request at this time.  
Definition at line 227 of file cfe\_error.h.

**10.11.2.71 CFE\_STATUS\_NO\_COUNTER\_INCREMENT** #define CFE\_STATUS\_NO\_COUNTER\_INCREMENT ((CFE\_Status\_t)0x48000000)  
No Counter Increment.

Informational code indicating that a command was processed successfully but that the command counter should *not* be incremented.  
Definition at line 155 of file cfe\_error.h.

**10.11.2.72 CFE\_STATUS\_NOT\_IMPLEMENTED** #define CFE\_STATUS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc800ffff)  
Not Implemented.

Current version does not have the function or the feature of the function implemented. This could be due to either an early build for this platform or the platform does not support the specified feature.  
Definition at line 238 of file cfe\_error.h.

**10.11.2.73 CFE\_STATUS\_RANGE\_ERROR** #define CFE\_STATUS\_RANGE\_ERROR ((int32)0xc8000008)

Request or input value is out of range.  
A message, table, or function call input contained a value that was outside the acceptable range, and the request was rejected.  
Definition at line 219 of file cfe\_error.h.

**10.11.2.74 CFE\_STATUS\_REQUEST\_ALREADY\_PENDING** #define CFE\_STATUS\_REQUEST\_ALREADY\_PENDING ((int32)0xc8000000)  
Request already pending.

Commands or requests are already pending or the pending request limit has been reached. No more requests can be made until the current request(s) complete.  
Definition at line 203 of file cfe\_error.h.

**10.11.2.75 CFE\_STATUS\_UNKNOWN\_MSG\_ID** #define CFE\_STATUS\_UNKNOWN\_MSG\_ID ((CFE\_Status\_t)0xc8000003)  
Unknown Message ID.

This error code will be returned when a message identification process determined that the message ID does not correspond to a known value  
Definition at line 173 of file cfe\_error.h.

**10.11.2.76 CFE\_STATUS\_VALIDATION\_FAILURE** #define CFE\_STATUS\_VALIDATION\_FAILURE ((int32)0xc8000007)  
Request or input value failed basic structural validation.

A message or table input was not in the proper format to be understood and processed by an application, and was rejected.  
Definition at line 211 of file cfe\_error.h.

**10.11.2.77 CFE\_STATUS\_WRONG\_MSG\_LENGTH** #define CFE\_STATUS\_WRONG\_MSG\_LENGTH (([CFE\\_Status\\_t](#)) 0xc8000002)  
Wrong Message Length.

This error code will be returned when a message validation process determined that the message length is incorrect  
Definition at line 164 of file `cfe_error.h`.

**10.11.2.78 CFE\_SUCCESS** #define CFE\_SUCCESS (([CFE\\_Status\\_t](#)) 0)

Successful execution.

Operation was performed successfully  
Definition at line 147 of file `cfe_error.h`.

**10.11.2.79 CFE\_TBL\_BAD\_ARGUMENT** #define CFE\_TBL\_BAD\_ARGUMENT (([CFE\\_Status\\_t](#)) 0xcc00002d)

Bad Argument.

A parameter given by a caller to a Table API did not pass validation checks.

Definition at line 1281 of file `cfe_error.h`.

**10.11.2.80 CFE\_TBL\_ERR\_ACCESS** #define CFE\_TBL\_ERR\_ACCESS (([CFE\\_Status\\_t](#)) 0xcc00002c)

Error code indicating that the TBL file could not be opened by the OS.

Definition at line 1272 of file `cfe_error.h`.

**10.11.2.81 CFE\_TBL\_ERR\_BAD\_CONTENT\_ID** #define CFE\_TBL\_ERR\_BAD\_CONTENT\_ID (([CFE\\_Status\\_t](#)) 0xcc000016)

Bad Content ID.

The calling Application called `CFE_TBL_Load` with a filename that specified a file whose content ID was not that of a table image.

Definition at line 1064 of file `cfe_error.h`.

**10.11.2.82 CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID** #define CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID (([CFE\\_Status\\_t](#)) 0xcc000029)

Bad Processor ID.

The selected table file failed validation for Processor ID. The platform configuration file has verification of table files enabled for Processor ID and an attempt was made to load a table with an invalid Processor ID in the table file header.  
Definition at line 1252 of file `cfe_error.h`.

**10.11.2.83 CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID** #define CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID (([CFE\\_Status\\_t](#)) 0xcc000028)

Bad Spacecraft ID.

The selected table file failed validation for Spacecraft ID. The platform configuration file has verification of table files enabled for Spacecraft ID and an attempt was made to load a table with an invalid Spacecraft ID in the table file header.  
Definition at line 1241 of file `cfe_error.h`.

**10.11.2.84 CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID** #define CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID (([CFE\\_Status\\_t](#)) 0xcc00001B)

Bad Subtype ID.

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.

Definition at line 1105 of file `cfe_error.h`.

**10.11.2.85 CFE\_TBL\_ERR\_DUMP\_ONLY** #define CFE\_TBL\_ERR\_DUMP\_ONLY ((CFE\_Status\_t)0xcc000010)  
Dump Only Error.

The calling Application has attempted to perform a load on a table that was created with "Dump Only" attributes.  
Definition at line 1016 of file cfe\_error.h.

**10.11.2.86 CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE** #define CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE ((CFE\_Status\_t)0xcc00000C)  
Duplicate Table With Different Size.

An application attempted to register a table with the same name as a table that is already in the registry. The size of the new table is different from the size already in the registry.

Definition at line 977 of file cfe\_error.h.

**10.11.2.87 CFE\_TBL\_ERR\_DUPLICATE\_NOT OWNED** #define CFE\_TBL\_ERR\_DUPLICATE\_NOT OWNED ((CFE\_Status\_t)0xcc00000B)  
Duplicate Table And Not Owned.

An application attempted to register a table with the same name as a table that is already in the registry. The previously registered table is owned by a different application.

Definition at line 987 of file cfe\_error.h.

**10.11.2.88 CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE** #define CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE ((CFE\_Status\_t)0xcc00000A)  
File For Wrong Table.

The calling Application tried to load a table using a file whose header indicated that it was for a different table.

Definition at line 1149 of file cfe\_error.h.

**10.11.2.89 CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT** #define CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT ((CFE\_Status\_t)0xcc000009)  
File Size Inconsistent.

The calling Application tried to access a table file whose Subtype identifier indicated it was not a table image file.

Definition at line 1114 of file cfe\_error.h.

**10.11.2.90 CFE\_TBL\_ERR\_FILE\_TOO\_LARGE** #define CFE\_TBL\_ERR\_FILE\_TOO\_LARGE ((CFE\_Status\_t)0xcc000014)  
File Too Large.

The calling Application called [CFE\\_TBL\\_Load](#) with a filename that specified a file that contained more data than the size of the table OR which contained more data than specified in the table header.

Definition at line 1044 of file cfe\_error.h.

**10.11.2.91 CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG** #define CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG ((CFE\_Status\_t)0xcc00001F)  
Filename Too Long.

The calling Application tried to load a table using a filename that was too long.

Definition at line 1140 of file cfe\_error.h.

**10.11.2.92 CFE\_TBL\_ERR\_HANDLES\_FULL** #define CFE\_TBL\_ERR\_HANDLES\_FULL ((CFE\_Status\_t)0xcc00000B)  
Handles Full.

An application attempted to create a table and the Table Handle Array already used all CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES in it.

Definition at line 967 of file cfe\_error.h.

**10.11.2.93 CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE** #define CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE ((CFE\_Status\_t)0xcc000011)  
Illegal Source Type.

The calling Application called [CFE\\_TBL\\_Load](#) with an illegal value for the second parameter.

Definition at line 1025 of file cfe\_error.h.

**10.11.2.94 CFE\_TBL\_ERR\_INVALID\_HANDLE** #define CFE\_TBL\_ERR\_INVALID\_HANDLE ((CFE\_Status\_t)0xcc000001)  
Invalid Handle.

The calling Application attempted to pass a Table handle that represented too large an index or identified a Table Access Descriptor that was not used.

Definition at line 884 of file cfe\_error.h.

**10.11.2.95 CFE\_TBL\_ERR\_INVALID\_NAME** #define CFE\_TBL\_ERR\_INVALID\_NAME ((CFE\_Status\_t)0xcc000002)  
Invalid Name.

The calling Application attempted to register a table whose name length exceeded the platform configuration value of [CFE\\_MISSION\\_TBL\\_MAX\\_NAME\\_LENGTH](#) or was zero characters long.

Definition at line 894 of file cfe\_error.h.

**10.11.2.96 CFE\_TBL\_ERR\_INVALID\_OPTIONS** #define CFE\_TBL\_ERR\_INVALID\_OPTIONS ((CFE\_Status\_t)0xcc000025)  
Invalid Options.

The calling Application has used an illegal combination of table options. A summary of the illegal combinations are as follows:

#[CFE\\_TBL\\_OPT\\_USR\\_DEF\\_ADDR](#) cannot be combined with any of the following:

1. [CFE\\_TBL\\_OPT\\_DBL\\_BUFFER](#)
2. [CFE\\_TBL\\_OPT\\_LOAD\\_DUMP](#)
3. [CFE\\_TBL\\_OPT\\_CRITICAL](#)

#[CFE\\_TBL\\_OPT\\_DBL\\_BUFFER](#) cannot be combined with the following:

1. [CFE\\_TBL\\_OPT\\_USR\\_DEF\\_ADDR](#)
2. [CFE\\_TBL\\_OPT\\_DUMP\\_ONLY](#)

Definition at line 1206 of file cfe\_error.h.

**10.11.2.97 CFE\_TBL\_ERR\_INVALID\_SIZE** #define CFE\_TBL\_ERR\_INVALID\_SIZE ((CFE\_Status\_t)0xcc000003)  
Invalid Size.

The calling Application attempted to register a table: a) that was a double buffered table with size greater than [CFE\\_PLATFORM\\_TBL\\_MAX\\_DBL\\_TABLE\\_SIZE](#) b) that was a single buffered table with size greater than [CFE\\_PLATFORM\\_TBL\\_MAX\\_SNGL\\_TABLE\\_SIZE](#) c) that had a size of zero

Definition at line 905 of file cfe\_error.h.

**10.11.2.98 CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS** #define CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS ((CFE\_Status\_t)0xcc000012)  
Load In Progress.

The calling Application called [CFE\\_TBL\\_Load](#) when another Application was trying to load the table.

Definition at line 1034 of file cfe\_error.h.

**10.11.2.99 CFE\_TBL\_ERR\_LOAD\_INCOMPLETE** #define CFE\_TBL\_ERR\_LOAD\_INCOMPLETE ((CFE\_Status\_t)0xcc000021)  
Load Incomplete.

The calling Application tried to load a table file whose header claimed the load was larger than what was actually read from the file.

Definition at line 1158 of file cfe\_error.h.

**10.11.2.100 CFE\_TBL\_ERR\_NEVER\_LOADED** #define CFE\_TBL\_ERR\_NEVER\_LOADED ((CFE\_Status\_t)0xcc000005)  
Never Loaded.

Table has not been loaded with data.

Definition at line 921 of file cfe\_error.h.

**10.11.2.101 CFE\_TBL\_ERR\_NO\_ACCESS** #define CFE\_TBL\_ERR\_NO\_ACCESS ((CFE\_Status\_t)0xcc000008)  
No Access.

The calling application either failed when calling [CFE\\_TBL\\_Register](#), failed when calling [CFE\\_TBL\\_Share](#) or forgot to call either one.

Definition at line 949 of file cfe\_error.h.

**10.11.2.102 CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL** #define CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL ((CFE\_Status\_t)0xcc00000F)  
No Buffer Available.

The calling Application has tried to allocate a working buffer but none were available.

Definition at line 1007 of file cfe\_error.h.

**10.11.2.103 CFE\_TBL\_ERR\_NO\_STD\_HEADER** #define CFE\_TBL\_ERR\_NO\_STD\_HEADER ((CFE\_Status\_t)0xcc00001D)  
No Standard Header.

The calling Application tried to access a table file whose standard cFE File Header was the wrong size, etc.

Definition at line 1122 of file cfe\_error.h.

**10.11.2.104 CFE\_TBL\_ERR\_NO\_TBL\_HEADER** #define CFE\_TBL\_ERR\_NO\_TBL\_HEADER ((CFE\_Status\_t)0xcc00001E)  
No Table Header.

The calling Application tried to access a table file whose standard cFE Table File Header was the wrong size, etc.

Definition at line 1131 of file cfe\_error.h.

**10.11.2.105 CFE\_TBL\_ERR\_PARTIAL\_LOAD** #define CFE\_TBL\_ERR\_PARTIAL\_LOAD ((CFE\_Status\_t)0xcc000023)  
Partial Load Error.

The calling Application tried to load a table file whose header claimed the load did not start with the first byte and the table image had NEVER been loaded before. Partial loads are not allowed on uninitialized tables. It should be noted that [CFE\\_TBL\\_WARN\\_SHORT\\_FILE](#) also indicates a partial load.

Definition at line 1180 of file cfe\_error.h.

**10.11.2.106 CFE\_TBL\_ERR\_REGISTRY\_FULL** #define CFE\_TBL\_ERR\_REGISTRY\_FULL ((CFE\_Status\_t)0xcc000006)  
Registry Full.

An application attempted to create a table and the Table registry already contained [CFE\\_PLATFORM\\_TBL\\_MAX\\_NUM\\_TABLES](#) in it.

Definition at line 930 of file cfe\_error.h.

**10.11.2.107 CFE\_TBL\_ERR\_SHORT\_FILE** #define CFE\_TBL\_ERR\_SHORT\_FILE ((CFE\_Status\_t)0xcc00002b)  
Error code indicating that the TBL file is shorter than indicated in the file header.  
Definition at line 1266 of file cfe\_error.h.

**10.11.2.108 CFE\_TBL\_ERR\_UNREGISTERED** #define CFE\_TBL\_ERR\_UNREGISTERED ((CFE\_Status\_t)0xcc000009)  
Unregistered.  
The calling application is trying to access a table that has been unregistered.  
Definition at line 958 of file cfe\_error.h.

**10.11.2.109 CFE\_TBL\_INFO\_DUMP\_PENDING** #define CFE\_TBL\_INFO\_DUMP\_PENDING ((CFE\_Status\_t)0x4c000024)  
Dump Pending.  
The calling Application should call [CFE\\_TBL\\_Manage](#) for the specified table. The ground has requested a dump of the Dump-Only table and needs to synchronize with the owning application.  
Definition at line 1190 of file cfe\_error.h.

**10.11.2.110 CFE\_TBL\_INFO\_NO\_DUMP\_PENDING** #define CFE\_TBL\_INFO\_NO\_DUMP\_PENDING ((CFE\_Status\_t)0x4c00002e)  
No Dump Pending.  
The calling Application invoked CFE\_TBL\_DumpToBuffer on a table that did not have a pending dump request  
Definition at line 1290 of file cfe\_error.h.

**10.11.2.111 CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING** #define CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING ((CFE\_Status\_t)0x4c000017)  
No Update Pending.  
The calling Application has attempted to update a table without a pending load.  
Definition at line 1072 of file cfe\_error.h.

**10.11.2.112 CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING** #define CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING ((CFE\_Status\_t)0x4c00002d)  
No Validation Pending  
The calling Application tried to validate a table that did not have a validation request pending.  
Definition at line 1096 of file cfe\_error.h.

**10.11.2.113 CFE\_TBL\_INFO\_RECOVERED\_TBL** #define CFE\_TBL\_INFO\_RECOVERED\_TBL ((CFE\_Status\_t)0x4c000027)  
Recovered Table.  
The calling Application registered a critical table whose previous contents were discovered in the Critical Data Store.  
The discovered contents were copied back into the newly registered table as the table's initial contents.  
**NOTE:** In this situation, the contents of the table are **NOT** validated using the table's validation function.  
Definition at line 1230 of file cfe\_error.h.

**10.11.2.114 CFE\_TBL\_INFO\_TABLE\_LOCKED** #define CFE\_TBL\_INFO\_TABLE\_LOCKED ((CFE\_Status\_t)0x4c000018)  
Table Locked.  
The calling Application tried to update a table that is locked by another user.  
Definition at line 1080 of file cfe\_error.h.

**10.11.2.115 CFE\_TBL\_INFO\_UPDATE\_PENDING** #define CFE\_TBL\_INFO\_UPDATE\_PENDING ((CFE\_Status\_t)0x4c000004)  
Update Pending.

The calling Application has identified a table that has a load pending.

Definition at line 913 of file cfe\_error.h.

**10.11.2.116 CFE\_TBL\_INFO\_UPDATED** #define CFE\_TBL\_INFO\_UPDATED ((CFE\_Status\_t)0x4c00000E)  
Updated.

The calling Application has identified a table that has been updated.

**NOTE:** This is a nominal return code informing the calling application that the table identified in the call has had its contents updated since the last time the application obtained its address or status.

Definition at line 998 of file cfe\_error.h.

**10.11.2.117 CFE\_TBL\_INFO\_VALIDATION\_PENDING** #define CFE\_TBL\_INFO\_VALIDATION\_PENDING ((CFE\_Status\_t)0x4c000001)  
Validation Pending

The calling Application should call [CFE\\_TBL\\_Validate](#) for the specified table.

Definition at line 1088 of file cfe\_error.h.

**10.11.2.118 CFE\_TBL\_MESSAGE\_ERROR** #define CFE\_TBL\_MESSAGE\_ERROR ((CFE\_Status\_t)0xcc00002a)  
Message Error.

Error code indicating that the TBL command was not processed successfully and that the error counter should be incremented.

Definition at line 1260 of file cfe\_error.h.

**10.11.2.119 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 1301 of file cfe\_error.h.

**10.11.2.120 CFE\_TBL\_WARN\_DUPLICATE** #define CFE\_TBL\_WARN\_DUPLICATE ((CFE\_Status\_t)0x4c000007)  
Duplicate Warning.

This is an error that the registration is trying to replace an existing table with the same name. The previous table stays in place and the new table is rejected.

Definition at line 940 of file cfe\_error.h.

**10.11.2.121 CFE\_TBL\_WARN\_NOT\_CRITICAL** #define CFE\_TBL\_WARN\_NOT\_CRITICAL ((CFE\_Status\_t)0x4c000026)  
Not Critical Warning.

The calling Application attempted to register a table as "Critical". Table Services failed to create an appropriate Critical Data Store (See System Log for reason) to save the table contents. The table will be treated as a normal table from now on.

Definition at line 1217 of file cfe\_error.h.

**10.11.2.122 CFE\_TBL\_WARN\_PARTIAL\_LOAD** #define CFE\_TBL\_WARN\_PARTIAL\_LOAD ((CFE\_Status\_t)0x4c000022)  
Partial Load Warning.

The calling Application tried to load a table file whose header claimed the load did not start with the first byte. It should be noted that [CFE\\_TBL\\_WARN\\_SHORT\\_FILE](#) also indicates a partial load.

Definition at line 1168 of file `cfe_error.h`.

**10.11.2.123 CFE\_TBL\_WARN\_SHORT\_FILE** `#define CFE_TBL_WARN_SHORT_FILE ((CFE_Status_t)0x4c000015)`

Short File Warning.

The calling Application called [CFE\\_TBL\\_Load](#) with a filename that specified a file that started with the first byte of the table but contained less data than the size of the table. It should be noted that [CFE\\_TBL\\_WARN\\_PARTIAL\\_LOAD](#) also indicates a partial load (one that starts at a non-zero offset).

Definition at line 1055 of file `cfe_error.h`.

**10.11.2.124 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 1373 of file `cfe_error.h`.

**10.11.2.125 CFE\_TIME\_CALLBACK\_NOT\_REGISTERED** `#define CFE_TIME_CALLBACK_NOT_REGISTERED ((CFE_Status_t)0xce000001)`

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 1364 of file `cfe_error.h`.

**10.11.2.126 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 1328 of file `cfe_error.h`.

**10.11.2.127 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 1316 of file `cfe_error.h`.

**10.11.2.128 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 1343 of file `cfe_error.h`.

**10.11.2.129 CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS** #define CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS ((CFE\_Status\_t)0x00000001)  
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 1354 of file cfe\_error.h.

## 10.12 cFE Resource ID APIs

### Functions

- **CFE\_Status\_t CFE\_ES\_AppID\_ToIndex (CFE\_ES\_AppId\_t AppID, uint32 \*Idx)**  
*Obtain an index value correlating to an ES Application ID.*
- **int32 CFE\_ES\_LibID\_ToIndex (CFE\_ES\_LibId\_t LibId, uint32 \*Idx)**  
*Obtain an index value correlating to an ES Library ID.*
- **CFE\_Status\_t CFE\_ES\_TaskID\_ToIndex (CFE\_ES\_TaskId\_t TaskID, uint32 \*Idx)**  
*Obtain an index value correlating to an ES Task ID.*
- **CFE\_Status\_t CFE\_ES\_CounterID\_ToIndex (CFE\_ES\_CounterId\_t CounterId, uint32 \*Idx)**  
*Obtain an index value correlating to an ES Counter ID.*

### 10.12.1 Detailed Description

### 10.12.2 Function Documentation

**10.12.2.1 CFE\_ES\_AppID\_ToIndex()** CFE\_Status\_t CFE\_ES\_AppID\_ToIndex ( CFES\_AppId\_t AppID,  
                                  uint32 \* Idx )

Obtain an index value correlating to an ES Application ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] application IDs will never overlap, but the index of an application and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

#### Note

There is no inverse of this function - indices cannot be converted back to the original AppID value. The caller should retain the original ID for future use.

#### Parameters

in	<i>AppID</i>	Application ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

#### Returns

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

#### Return values

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

```
10.12.2.2 CFE_ES_CounterID_ToIndex() CFE_Status_t CFE_ES_CounterID_ToIndex (
    CFE_ES_CounterId_t CounterId,
    uint32 * Idx )
```

Obtain an index value correlating to an ES Counter ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Counter IDs will never overlap, but the index of a Counter and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

#### Note

There is no inverse of this function - indices cannot be converted back to the original CounterID value. The caller should retain the original ID for future use.

#### Parameters

in	<i>CounterId</i>	Counter ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

#### Returns

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

#### Return values

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

```
10.12.2.3 CFE_ES_LibID_ToIndex() int32 CFE_ES_LibID_ToIndex (
    CFE_ES_LibId_t LibId,
    uint32 * Idx )
```

Obtain an index value correlating to an ES Library ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Library IDs will never overlap, but the index of an Library and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

#### Note

There is no inverse of this function - indices cannot be converted back to the original LibID value. The caller should retain the original ID for future use.

#### Parameters

in	<i>LibId</i>	Library ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

**Returns**

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

**Return values**

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

**10.12.2.4 CFE\_ES\_TaskID\_ToIndex()** *CFE\_Status\_t* CFE\_ES\_TaskID\_ToIndex (

```
    CFE_ES_TaskId_t TaskID,
    uint32 * Idx )
```

Obtain an index value correlating to an ES Task ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] Task IDs will never overlap, but the index of a Task and a library ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

**Note**

There is no inverse of this function - indices cannot be converted back to the original TaskID value. The caller should retain the original ID for future use.

**Parameters**

<i>in</i>	<i>TaskID</i>	Task ID to convert
<i>out</i>	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

**Returns**

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

**Return values**

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

## 10.13 cFE Entry/Exit APIs

**Functions**

- void *CFE\_ES\_Main* (*uint32* StartType, *uint32* StartSubtype, *uint32* ModelId, const char \*StartFilePath)  
*cFE Main Entry Point used by Board Support Package to start cFE*
- *CFE\_Status\_t* *CFE\_ES\_ResetCFE* (*uint32* ResetType)  
*Reset the cFE Core and all cFE Applications.*

### 10.13.1 Detailed Description

#### 10.13.2 Function Documentation

```
10.13.2.1 CFE_ES_Main() void CFE_ES_Main (
    uint32 StartType,
    uint32 StartSubtype,
    uint32 ModelId,
    const char * StartFilePath )
```

cFE Main Entry Point used by Board Support Package to start cFE

##### Description

cFE main entry point. This is the entry point into the cFE software. It is called only by the Board Support Package software.

##### Assumptions, External Events, and Notes:

None

##### Parameters

in	<i>StartType</i>	Identifies whether this was a <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](#)

```
10.13.2.2 CFE_ES_ResetCFE() CFE_Status_t CFE_ES_ResetCFE (
    uint32 ResetType )
```

Reset the cFE Core and all cFE Applications.

##### Description

This API causes an immediate reset of the cFE Kernel and all cFE Applications. The caller can specify whether the reset should clear all memory ([CFE\\_PSP\\_RST\\_TYPE\\_POWERON](#)) or try to retain volatile memory areas ([CFE\\_PSP\\_RST\\_TYPE\\_PROCESSOR](#)).

##### Assumptions, External Events, and Notes:

None

**Parameters**

in	<i>ResetType</i>	Identifies the type of reset desired. Allowable settings are: <ul style="list-style-type: none"><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](#)

## 10.14 cFE Application Control APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_ES\\_RestartApp \(CFE\\_ES\\_AppId\\_t AppID\)](#)  
*Restart a single cFE Application.*
- [CFE\\_Status\\_t CFE\\_ES\\_ReloadApp \(CFE\\_ES\\_AppId\\_t AppID, const char \\*AppFileName\)](#)  
*Reload a single cFE Application.*
- [CFE\\_Status\\_t CFE\\_ES\\_DeleteApp \(CFE\\_ES\\_AppId\\_t AppID\)](#)  
*Delete a cFE Application.*

### 10.14.1 Detailed Description

### 10.14.2 Function Documentation

#### 10.14.2.1 CFE\_ES\_DeleteApp() [CFE\\_Status\\_t CFE\\_ES\\_DeleteApp \( CFE\\_ES\\_AppId\\_t AppID \)](#)

Delete a cFE Application.

**Description**

This API causes a cFE Application to be stopped deleted.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>AppID</i>	Identifies the application to be reset.
----	--------------	---

**Returns**

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

**Return values**

<a href="#"><i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i></a>	Resource ID is not valid.
<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.

**See also**

[CFE\\_ES\\_RestartApp](#), [CFE\\_ES\\_ReloadApp](#)

**10.14.2.2 CFE\_ES\_ReloadApp()** [\*CFE\\_Status\\_t\*](#) *CFE\_ES\_ReloadApp* (

```
    CFE_ES_AppId_t AppID,
    const char * AppFileName )
```

Reload a single cFE Application.

**Description**

This API causes a cFE Application to be stopped and restarted from the specified file.

**Assumptions, External Events, and Notes:**

The filename is checked for existence prior to load. A missing file will be reported and the reload operation will be aborted prior to unloading the app.

Goes through the standard CFE\_ES\_CleanUpApp which unloads, then attempts a load using the specified file name. In the event that an application cannot be reloaded due to a corrupt file, the application may no longer be reloaded when given a valid load file (it has been deleted and no longer exists). To recover, the application may be started by loading the application via the ES\_STARTAPP command ([CFE\\_ES\\_START\\_APP\\_CC](#)).

**Parameters**

in	<i>AppID</i>	Identifies the application to be reset.
in	<i>AppFileName</i>	Identifies the new file to start (must not be null)

**Returns**

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

**Return values**

<a href="#"><i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i></a>	Resource ID is not valid.
<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_ES_FILE_IO_ERR</i></a>	File IO Error.

**See also**

[CFE\\_ES\\_RestartApp](#), [CFE\\_ES\\_DeleteApp](#), [CFE\\_ES\\_START\\_APP\\_CC](#)

**10.14.2.3 CFE\_ES\_RestartApp()** `CFE_Status_t CFE_ES_RestartApp (`  
`CFE_ES_AppId_t AppID )`

Restart a single cFE Application.

**Description**

This API causes a cFE Application to be unloaded and restarted from the same file name as the last start.

**Assumptions, External Events, and Notes:**

The filename is checked for existence prior to load. A missing file will be reported and the reload operation will be aborted prior to unloading the app.

Goes through the standard CFE\_ES\_CleanUpApp which unloads, then attempts a load using the original file name. In the event that an application cannot be reloaded due to a missing file or any other load issue, the application may no longer be restarted or reloaded when given a valid load file (the app has been deleted and no longer exists). To recover, the application may be started by loading the application via the ES\_STARTAPP command ([CFE\\_ES\\_START\\_APP\\_CC](#)).

**Parameters**

in	<i>AppID</i>	Identifies the application to be reset.
----	--------------	---

**Returns**

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

**Return values**

<a href="#">CFE_ES_ERR_RESOURCEID_NOT_VALID</a>	Resource ID is not valid.
<a href="#">CFE_ES_FILE_IO_ERR</a>	File IO Error.
<a href="#">CFE_SUCCESS</a>	Successful execution.

**See also**

[CFE\\_ES\\_ReloadApp](#), [CFE\\_ES\\_DeleteApp](#)

## 10.15 cFE Application Behavior APIs

**Functions**

- void [CFE\\_ES\\_ExitApp](#) (`uint32 ExitStatus`)  
*Exit a cFE Application.*
- bool [CFE\\_ES\\_RunLoop](#) (`uint32 *RunStatus`)  
*Check for Exit, Restart, or Reload commands.*
- [CFE\\_Status\\_t CFE\\_ES\\_WaitForSystemState](#) (`uint32 MinSystemState`, `uint32 TimeOutMilliseconds`)  
*Allow an Application to Wait for a minimum global system state.*
- void [CFE\\_ES\\_WaitForStartupSync](#) (`uint32 TimeOutMilliseconds`)

*Allow an Application to Wait for the "OPERATIONAL" global system state.*

- void [CFE\\_ES\\_IncrementTaskCounter](#) (void)

*Increments the execution counter for the calling task.*

### 10.15.1 Detailed Description

### 10.15.2 Function Documentation

#### 10.15.2.1 CFE\_ES\_ExitApp() void CFE\_ES\_ExitApp (

`uint32 ExitStatus` )

Exit a cFE Application.

##### Description

This API is the "Exit Point" for the cFE application

##### Assumptions, External Events, and Notes:

None

##### Parameters

in	<i>ExitStatus</i>	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 CF\_AppMain().

#### 10.15.2.2 CFE\_ES\_IncrementTaskCounter() void CFE\_ES\_IncrementTaskCounter (

`void` )

Increments the execution counter for the calling task.

##### Description

This routine increments the execution counter that is stored for the calling task. It can be called from cFE Application main tasks, child tasks, or cFE Core application main tasks. Normally, the call is not necessary from a cFE Application, since the [CFE\\_ES\\_RunLoop](#) call increments the counter for the Application.

**Assumptions, External Events, and Notes:**

NOTE: This API is not needed for Applications that call the CFE\_ES\_RunLoop call.

**See also**

[CFE\\_ES\\_RunLoop](#)

**10.15.2.3 CFE\_ES\_RunLoop()** `bool CFE_ES_RunLoop (`  
`uint32 * RunStatus )`

Check for Exit, Restart, or Reload commands.

**Description**

This is the API that allows an app to check for exit requests from the system, or request shutdown from the system.

**Assumptions, External Events, and Notes:**

This API updates the internal task counter tracked by ES for the calling task. For ES to report application counters correctly this API should be called from the main app task as part of it's main processing loop.

In the event of a externally initiated app shutdown request (such as the APP\_STOP, APP\_RELOAD, and APP\_RESTART 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**

<code>in, out</code>	<code>RunStatus</code>	Optional pointer to a variable containing the desired run status
----------------------	------------------------	--

**Returns**

Boolean indicating application should continue running

**Return values**

<code>true</code>	Application should continue running
<code>false</code>	Application should not continue running

**See also**

[CFE\\_ES\\_ExitApp](#)

Referenced by CF\_AppMain().

```
10.15.2.4 CFE_ES_WaitForStartupSync() void CFE_ES_WaitForStartupSync (
    uint32 TimeOutMilliseconds )
```

Allow an Application to Wait for the "OPERATIONAL" global system state.

#### Description

This is the API that allows an app to wait for the rest of the apps to complete their entire initialization before continuing. It is most useful for applications such as Health and Safety or the Scheduler that need to wait until applications exist and are running before sending out packets to them.

This is a specialized wrapper for CFE\_ES\_WaitForSystemState for compatibility with applications using this API.

#### Assumptions, External Events, and Notes:

This API should only be called as the last item of an Apps initialization. In addition, this API should only be called by an App that is started from the ES Startup file. It should not be used by an App that is started after the system is running. ( Although it will cause no harm )

#### Parameters

in	<i>TimeOutMilliseconds</i>	The timeout value in Milliseconds. This parameter must be at least 1000. Lower values will be rounded up. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.
----	----------------------------	---

#### See also

[CFE\\_ES\\_RunLoop](#)

```
10.15.2.5 CFE_ES_WaitForSystemState() CFE_Status_t CFE_ES_WaitForSystemState (
    uint32 MinSystemState,
    uint32 TimeOutMilliseconds )
```

Allow an Application to Wait for a minimum global system state.

#### Description

This is the API that allows an app to wait for the rest of the apps to complete a given stage of initialization before continuing.

This gives finer grained control than [CFE\\_ES\\_WaitForStartupSync](#)

#### Assumptions, External Events, and Notes:

This API assumes that the caller has also been initialized sufficiently to satisfy the global system state it is waiting for, and the apps own state will be updated accordingly.

#### Parameters

in	<i>MinSystemState</i>	Determine the state of the App
in	<i>TimeOutMilliseconds</i>	The timeout value in Milliseconds. There is not an option to wait indefinitely to avoid hanging a critical application because a non-critical app did not start.

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	State successfully achieved
<a href="#">CFE_ES_OPERATION_TIMED_OUT</a>	(return value only verified in coverage test) Timeout was reached

**See also**

[CFE\\_ES\\_RunLoop](#)

## 10.16 cFE Information APIs

**Functions**

- [int32 CFE\\_ES\\_GetResetType \(uint32 \\*ResetSubtypePtr\)](#)  
*Return the most recent Reset Type.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetAppID \(CFE\\_ES\\_AppId\\_t \\*AppIdPtr\)](#)  
*Get an Application ID for the calling Application.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetTaskID \(CFE\\_ES\\_TaskId\\_t \\*TaskIdPtr\)](#)  
*Get the task ID of the calling context.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetAppIDByName \(CFE\\_ES\\_AppId\\_t \\*AppIdPtr, const char \\*AppName\)](#)  
*Get an Application ID associated with a specified Application name.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetLibIdByName \(CFE\\_ES\\_LibId\\_t \\*LibIdPtr, const char \\*LibName\)](#)  
*Get a Library ID associated with a specified Library name.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetAppName \(char \\*AppName, CFE\\_ES\\_AppId\\_t AppId, size\\_t BufferLength\)](#)  
*Get an Application name for a specified Application ID.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetLibName \(char \\*LibName, CFE\\_ES\\_LibId\\_t LibId, size\\_t BufferLength\)](#)  
*Get a Library name for a specified Library ID.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetAppInfo \(CFE\\_ES\\_AppInfo\\_t \\*AppInfo, CFE\\_ES\\_AppId\\_t AppId\)](#)  
*Get Application Information given a specified App ID.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetTaskInfo \(CFE\\_ES\\_TaskInfo\\_t \\*TaskInfo, CFE\\_ES\\_TaskId\\_t TaskId\)](#)  
*Get Task Information given a specified Task ID.*
- [int32 CFE\\_ES\\_GetLibInfo \(CFE\\_ES\\_AppInfo\\_t \\*LibInfo, CFE\\_ES\\_LibId\\_t LibId\)](#)  
*Get Library Information given a specified Resource ID.*
- [int32 CFE\\_ES\\_GetModuleInfo \(CFE\\_ES\\_AppInfo\\_t \\*ModuleInfo, CFE\\_Resourceld\\_t Resourceld\)](#)  
*Get Information given a specified Resource ID.*

### 10.16.1 Detailed Description

### 10.16.2 Function Documentation

**10.16.2.1 CFE\_ES\_GetAppID()** `CFE_Status_t CFE_ES_GetAppID (`  
 `CFE_ES_AppId_t * AppIdPtr )`

Get an Application ID for the calling Application.

#### Description

This routine retrieves the cFE Application ID for the calling Application.

#### Assumptions, External Events, and Notes:

NOTE: All tasks associated with the Application would return the same Application ID.

#### Parameters

out	<i>AppIdPtr</i>	Pointer to variable that is to receive the Application's ID (must not be null). *AppIdPtr will be set to the application ID of the calling Application.
-----	-----------------	---

#### Returns

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

#### Return values

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

#### See also

[CFE\\_ES\\_GetResetType](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppName](#), [CFE\\_ES\\_GetTaskInfo](#)

**10.16.2.2 CFE\_ES\_GetAppIDByName()** `CFE_Status_t CFE_ES_GetAppIDByName (`  
 `CFE_ES_AppId_t * AppIdPtr,`  
 `const char * AppName )`

Get an Application ID associated with a specified Application name.

#### Description

This routine retrieves the cFE Application ID associated with a specified Application name.

#### Assumptions, External Events, and Notes:

None

#### Parameters

out	<i>AppIdPtr</i>	Pointer to variable that is to receive the Application's ID (must not be null).
in	<i>AppName</i>	Pointer to null terminated character string containing an Application name (must not be null).

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_NAME_NOT_FOUND</a>	Resource Name Error.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_ES\\_GetAppID](#), [CFE\\_ES\\_GetAppName](#), [CFE\\_ES\\_GetAppInfo](#)

**10.16.2.3 CFE\_ES\_GetAppInfo()** [CFE\\_Status\\_t](#) CFE\_ES\_GetAppInfo (

[CFE\\_ES\\_AppInfo\\_t](#) \* *AppInfo*,  
    [CFE\\_ES\\_AppId\\_t](#) *AppId* )

Get Application Information given a specified App ID.

**Description**

This routine retrieves the information about an App associated with a specified App ID. The information includes all of the information ES maintains for an application ( documented in the [CFE\\_ES\\_AppInfo\\_t](#) type )

**Assumptions, External Events, and Notes:**

None

**Parameters**

<a href="#">out</a>	<i>AppInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
<a href="#">in</a>	<i>AppId</i>	ID of application to obtain information about

**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\\_GetAppID](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppName](#)

```
10.16.2.4 CFE_ES_GetAppName() CFE\_Status\_t CFE_ES_GetAppName (
    char * AppName,
    CFE\_ES\_AppId\_t AppId,
    size_t BufferLength )
```

Get an Application name for a specified Application ID.

#### Description

This routine retrieves the cFE Application name associated with a specified Application ID.

#### Assumptions, External Events, and Notes:

In the case of a failure ([CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID](#)), an empty string is returned.

#### Parameters

<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

<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\\_GetAppID](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppInfo](#)

```
10.16.2.5 CFE_ES_GetLibIDByName() CFE\_Status\_t CFE_ES_GetLibIDByName (
    CFE\_ES\_LibId\_t * LibIdPtr,
    const char * LibName )
```

Get a Library ID associated with a specified Library name.

#### Description

This routine retrieves the cFE Library ID associated with a specified Library name.

#### Assumptions, External Events, and Notes:

None

**Parameters**

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

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_GetLibName](#)

**10.16.2.6 CFE\_ES\_GetLibInfo()** `int32 CFE_ES_GetLibInfo (`  
 `CFE_ES_AppInfo_t * LibInfo,`  
 `CFE_ES_LibId_t LibId )`

Get Library Information given a specified Resource ID.

**Description**

This routine retrieves the information about a Library associated with a specified ID. The information includes all of the information ES maintains for this resource type ( documented in the `CFE_ES_AppInfo_t` type ).

This shares the same output structure as `CFE_ES_GetAppInfo`, such that informational commands can be executed against either applications or libraries. When applied to a library, the task information in the structure will be omitted, as libraries do not have tasks associated.

**Assumptions, External Events, and Notes:**

None

**Parameters**

out	<i>LibInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	<i>LibId</i>	ID of application to obtain information about

**Returns**

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

**Return values**

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

**Return values**

<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.
----------------------------------	---------------

**See also**

[CFE\\_ES\\_GetLibIDByName](#), [CFE\\_ES\\_GetLibName](#)

```
10.16.2.7 CFE_ES_GetLibName() CFE_Status_t CFE_ES_GetLibName (
    char * LibName,
    CFE_ES_LibId_t LibId,
    size_t BufferLength )
```

Get a Library name for a specified Library ID.

**Description**

This routine retrieves the cFE Library name associated with a specified Library ID.

**Assumptions, External Events, and Notes:**

In the case of a failure ([CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID](#)), an empty string is returned.

**Parameters**

<code>out</code>	<code>LibName</code>	Pointer to a character array (must not be null) of at least <code>BufferLength</code> in size that will be filled with the Library name.
<code>in</code>	<code>LibId</code>	Library ID of Library whose name is being requested.
<code>in</code>	<code>BufferLength</code>	The maximum number of characters (must not be zero), including the null terminator, that can be put into the <code>LibName</code> buffer. This routine will truncate the name to this length, if necessary.

**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\\_GetLibIDByName](#)

```
10.16.2.8 CFE_ES_GetModuleInfo() int32 CFE_ES_GetModuleInfo (
    CFE_ES_AppInfo_t * ModuleInfo,
    CFE_ResourceId_t ResourceId )
```

Get Information given a specified Resource ID.

#### Description

This routine retrieves the information about an Application or Library associated with a specified ID.

This is a wrapper API that in turn calls either CFE\_ES\_GetAppInfo or CFE\_ES\_GetLibInfo if passed an AppId or LibId, respectively.

This allows commands originally targeted to operate on AppIDs to be easily ported to operate on either Libraries or Applications, where relevant.

#### Assumptions, External Events, and Notes:

None

#### Parameters

out	<i>ModuleInfo</i>	Pointer to a structure (must not be null) that will be filled with resource name and memory addresses information.
in	<i>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\\_GetAppInfo](#)

**10.16.2.9 CFE\_ES\_GetResetType()** `int32 CFE_ES_GetResetType (`  
`uint32 * ResetSubtypePtr )`

Return the most recent Reset Type.

#### Description

Provides the caller with codes that identifies the type of Reset the processor most recently underwent. The caller can also obtain information on what caused the reset by supplying a pointer to a variable that will be filled with the Reset Sub-Type.

#### Assumptions, External Events, and Notes:

None

**Parameters**

in, out	<i>ResetSubtypePtr</i>	Pointer to <code>uint32</code> type variable in which the Reset Sub-Type will be stored. The caller can set this pointer to NULL if the Sub-Type is of no interest. <i>ResetSubtypePtr</i> If the provided pointer was not NULL, the Reset Sub-Type is stored at the given address. For a list of possible Sub-Type values, see " <a href="#">Reset Sub-Types</a> ".
---------	------------------------	---

**Returns**

Processor reset type

**Return values**

<a href="#"><code>CFE_PSP_RST_TYPE_POWERON</code></a>	
<a href="#"><code>CFE_PSP_RST_TYPE_PROCESSOR</code></a>	

**See also**

[CFE\\_ES\\_GetAppID](#), [CFE\\_ES\\_GetAppIDByName](#), [CFE\\_ES\\_GetAppName](#), [CFE\\_ES\\_GetTaskInfo](#)

**10.16.2.10 CFE\_ES\_GetTaskID()** `CFE_Status_t CFE_ES_GetTaskID ( CFE_ES_TaskId_t * TaskIdPtr )`

Get the task ID of the calling context.

**Description**

This retrieves the current task context from OSAL

**Assumptions, External Events, and Notes:**

Applications which desire to call other CFE ES services such as `CFE_ES_TaskGetInfo()` should use this API rather than getting the ID from OSAL directly via [OS\\_TaskGetId\(\)](#).

**Parameters**

out	<i>TaskIdPtr</i>	Pointer to variable that is to receive the ID (must not be null). Will be set to the ID of the calling task.
-----	------------------	--

**Returns**

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

**Return values**

<a href="#"><code>CFE_SUCCESS</code></a>	Successful execution.
<a href="#"><code>CFE_ES_ERR_RESOURCEID_NOT_VALID</code></a>	Resource ID is not valid.
<a href="#"><code>CFE_ES_BAD_ARGUMENT</code></a>	Bad Argument.

---

```
10.16.2.11 CFE_ES_GetTaskInfo() CFE\_Status\_t CFE_ES_GetTaskInfo (
    CFE\_ES\_TaskInfo\_t * TaskInfo,
    CFE\_ES\_TaskId\_t TaskId )
```

Get Task Information given a specified Task ID.

#### Description

This routine retrieves the information about a Task associated with a specified Task ID. The information includes Task Name, and Parent/Creator Application ID.

#### Assumptions, External Events, and Notes:

None

#### Parameters

out	<i>TaskInfo</i>	Pointer to a <a href="#">CFE_ES_TaskInfo_t</a> structure (must not be null) that holds the specific task information. *TaskInfo is the filled out <a href="#">CFE_ES_TaskInfo_t</a> 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

<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\\_GetTaskID](#), [CFE\\_ES\\_GetTaskIDByName](#), [CFE\\_ES\\_GetTaskName](#)

## 10.17 cFE Child Task APIs

#### Functions

- [CFE\\_Status\\_t CFE\\_ES\\_CreateChildTask](#) ([CFE\\_ES\\_TaskId\\_t](#)\*TaskIdPtr, const char \*TaskName, [CFE\\_ES\\_ChildTaskMainFuncPtr](#) FunctionPtr, [CFE\\_ES\\_StackPointer\\_t](#) StackPtr, [size\\_t](#) StackSize, [CFE\\_ES\\_TaskPriority\\_Atom\\_t](#) Priority, [uint32](#) Flags)
 

*Creates a new task under an existing Application.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetTaskIDByName](#) ([CFE\\_ES\\_TaskId\\_t](#)\*TaskIdPtr, const char \*TaskName)
 

*Get a Task ID associated with a specified Task name.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetTaskName](#) (char \*TaskName, [CFE\\_ES\\_TaskId\\_t](#) TaskId, [size\\_t](#) BufferLength)
 

*Get a Task name for a specified Task ID.*
- [CFE\\_Status\\_t CFE\\_ES\\_DeleteChildTask](#) ([CFE\\_ES\\_TaskId\\_t](#) TaskId)
 

*Deletes a task under an existing Application.*

- void [CFE\\_ES\\_ExitChildTask](#) (void)

*Exits a child task.*

### 10.17.1 Detailed Description

### 10.17.2 Function Documentation

#### 10.17.2.1 [CFE\\_ES\\_CreateChildTask\(\)](#) `CFE_Status_t CFE_ES_CreateChildTask (`

```
    CFE_ES_TaskId_t * TaskIdPtr,
    const char * TaskName,
    CFE_ES_ChildTaskMainFuncPtr_t FunctionPtr,
    CFE_ES_StackPointer_t StackPtr,
    size_t StackSize,
    CFE_ES_TaskPriority_Atom_t Priority,
    uint32 Flags )
```

Creates a new task under an existing Application.

#### Description

This routine creates a new task (a separate execution thread) owned by the calling Application.

#### Assumptions, External Events, and Notes:

None

#### Parameters

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 <a href="#">OS_MAX_API_NAME</a> 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 <a href="#">CFE_ES_TASK_STACK_ALLOCATE</a> constant may be passed to indicate that the stack should be dynamically allocated.
in	<i>StackSize</i>	The number of bytes to allocate for the new task's stack (must not be zero).
in	<i>Priority</i>	The priority for the new task. Lower numbers are higher priority, with 0 being the highest priority.
in	<i>Flags</i>	Reserved for future expansion.

#### Returns

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

#### Return values

<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](#)

**10.17.2.2 CFE\_ES\_DeleteChildTask()** `CFE_Status_t CFE_ES_DeleteChildTask (`

`CFE_ES_TaskId_t TaskId )`

Deletes a task under an existing Application.

**Description**

This routine deletes a task under an Application specified by the `TaskId` obtained when the child task was created using the [CFE\\_ES\\_CreateChildTask](#) API.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TaskId</i>	The task ID previously obtained when the Child Task was created with the <a href="#">CFE_ES_CreateChildTask</a> API.
----	---------------	--

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_CreateChildTask](#), [CFE\\_ES\\_ExitChildTask](#)

**10.17.2.3 CFE\_ES\_ExitChildTask()** `void CFE_ES_ExitChildTask (`

`void )`

Exits a child task.

**Description**

This routine allows the current executing child task to exit and be deleted by ES.

**Assumptions, External Events, and Notes:**

This function cannot be called from an Application's Main Task.

**Note**

This function does not return a value, but if it does return at all, it is assumed that the Task was either unregistered or this function was called from a cFE Application's main task.

**See also**

[CFE\\_ES\\_CreateChildTask](#), [CFE\\_ES\\_DeleteChildTask](#)

**10.17.2.4 CFE\_ES\_GetTaskIDByName()** `CFE_Status_t CFE_ES_GetTaskIDByName (`  
 `CFE_ES_TaskId_t * TaskIdPtr,`  
 `const char * TaskName )`

Get a Task ID associated with a specified Task name.

**Description**

This routine retrieves the cFE Task ID associated with a specified Task name.

**Assumptions, External Events, and Notes:**

None

**Parameters**

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

**Returns**

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

**Return values**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_ERR_NAME_NOT_FOUND</code>	Resource Name Error.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.

**See also**

[CFE\\_ES\\_GetTaskName](#)

**10.17.2.5 CFE\_ES\_GetTaskName()** `CFE_Status_t CFE_ES_GetTaskName (`  
 `char * TaskName,`  
 `CFE_ES_TaskId_t TaskId,`  
 `size_t BufferLength )`

Get a Task name for a specified Task ID.

**Description**

This routine retrieves the cFE Task name associated with a specified Task ID.

**Assumptions, External Events, and Notes:**

In the case of a failure ([CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID](#)), an empty string is returned.

**Parameters**

out	<i>TaskName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Task name.
in	<i>TaskId</i>	Task ID of Task whose name is being requested.
in	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <i>TaskName</i> buffer. This routine will truncate the name to this length, if necessary.

**Returns**

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

**Return values**

<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\\_GetTaskIDByName](#)

## 10.18 cFE Miscellaneous APIs

**Functions**

- void [CFE\\_ES\\_BackgroundWakeup](#) (void)  
*Wakes up the CFE background task.*
- [CFE\\_Status\\_t CFE\\_ES\\_WriteToSysLog](#) (const char \*SpecStringPtr,...) [OS\\_PRINTF](#)(1  
*Write a string to the cFE System Log.*
- [CFE\\_Status\\_t uint32 CFE\\_ES\\_CalculateCRC](#) (const void \*DataPtr, size\_t DataLength, uint32 InputCRC, [CFE\\_ES\\_CrcType\\_Enum\\_t](#) TypeCRC)  
*Calculate a CRC on a block of memory.*
- void [CFE\\_ES\\_ProcessAsyncEvent](#) (void)  
*Notification that an asynchronous event was detected by the underlying OS/PSP.*

### 10.18.1 Detailed Description

### 10.18.2 Function Documentation

#### 10.18.2.1 [CFE\\_ES\\_BackgroundWakeup\(\)](#) void CFE\_ES\_BackgroundWakeup (

```
void )
```

Wakes up the CFE background task.

**Description**

Normally the ES background task wakes up at a periodic interval. Whenever new background work is added, this can be used to wake the task early, which may reduce the delay between adding the job and the job getting processed.

**Assumptions, External Events, and Notes:**

Note the amount of work that the background task will perform is pro-rated based on the amount of time elapsed since the last wakeup. Waking the task early will not cause the background task to do more work than it otherwise would - it just reduces the delay before work starts initially.

```
10.18.2.2 CFE_ES_CalculateCRC() CFE_Status_t uint32 CFE_ES_CalculateCRC (
    const void * DataPtr,
    size_t DataLength,
    uint32 InputCRC,
    CFE_ES_CrcType_Enum_t TypeCRC )
```

Calculate a CRC on a block of memory.

**Description**

This routine calculates a cyclic redundancy check (CRC) on a block of memory. The CRC algorithm used is determined by the last parameter.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>DataPtr</i>	Pointer to the base of the memory block.
in	<i>DataLength</i>	The number of bytes in the memory block.
in	<i>InputCRC</i>	A starting value for use in the CRC calculation. This parameter allows the user to calculate the CRC of non-contiguous blocks as a single value. Nominally, the user should set this value to zero.
in	<i>TypeCRC</i>	One of the following CRC algorithm selections defined in <code>CFE_ES_CrcType_Enum_t</code>

**Returns**

The result of the CRC calculation on the specified memory block. If the TypeCRC is unimplemented will return 0. If DataPtr is null or DataLength is 0, will return InputCRC

```
10.18.2.3 CFE_ES_ProcessAsyncEvent() void CFE_ES_ProcessAsyncEvent (
    void )
```

Notification that an asynchronous event was detected by the underlying OS/PSP.

**Description**

This hook routine is called from the PSP when an exception or other asynchronous system event occurs

**Assumptions, External Events, and Notes:**

The PSP must guarantee that this function is only invoked from a context which may use OSAL primitives. In general this means that it shouldn't be *directly* invoked from an ISR/signal context.

**10.18.2.4 CFE\_ES\_WriteToSysLog()** `CFE_Status_t CFE_ES_WriteToSysLog (`  
    `const char * SpecStringPtr,`  
    `... )`

Write a string to the cFE System Log.

**Description**

This routine writes a formatted string to the cFE system log. This can be used to record very low-level errors that can't be reported using the Event Services. This function is used in place of printf for flight software. It should be used for significant startup events, critical errors, and conditionally compiled debug software.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<code>SpecStringPtr</code>	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**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_ES_ERR_SYS_LOG_FULL</code>	System Log Full.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.

Referenced by CF\_AppInit(), and CF\_CFDP\_RecycleTransaction().

## 10.19 cFE Critical Data Store APIs

**Functions**

- `CFE_Status_t CFE_ES_RegisterCDS (CFE_ES_CDSHandle_t *CDSHandlePtr, size_t BlockSize, const char *Name)`  
*Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)*
- `CFE_Status_t CFE_ES_GetCDSBlockIDByName (CFE_ES_CDSHandle_t *BlockIdPtr, const char *BlockName)`  
*Get a CDS Block ID associated with a specified CDS Block name.*
- `CFE_Status_t CFE_ES_GetCDSBlockName (char *BlockName, CFE_ES_CDSHandle_t BlockId, size_t BufferLength)`  
*Get a Block name for a specified Block ID.*
- `CFE_Status_t CFE_ES_CopyToCDS (CFE_ES_CDSHandle_t Handle, const void *DataToCopy)`

*Save a block of data in the Critical Data Store (CDS)*

- [CFE\\_Status\\_t CFE\\_ES\\_RestoreFromCDS](#) (`void *RestoreToMemory, CFE_ES_CDSHandle_t Handle`)

*Recover a block of data from the Critical Data Store (CDS)*

### 10.19.1 Detailed Description

### 10.19.2 Function Documentation

#### 10.19.2.1 [CFE\\_ES\\_CopyToCDS\(\)](#) `CFE_Status_t CFE_ES_CopyToCDS (`

```
    CFE_ES_CDSHandle_t Handle,  
    const void * DataToCopy )
```

Save a block of data in the Critical Data Store (CDS)

##### Description

This routine copies a specified block of memory into the Critical Data Store that had been previously registered via [CFE\\_ES\\_RegisterCDS](#). The block of memory to be copied must be at least as big as the size specified when registering the CDS.

##### Assumptions, External Events, and Notes:

None

##### Parameters

in	<i>Handle</i>	The handle of the CDS block that was previously obtained from <a href="#">CFE_ES_RegisterCDS</a> .
in	<i>DataToCopy</i>	A Pointer to the block of memory to be copied into the CDS (must not be null).

##### Returns

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

##### Return values

<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\\_RegisterCDS](#), [CFE\\_ES\\_RestoreFromCDS](#)

#### 10.19.2.2 [CFE\\_ES\\_GetCDSBlockIDByName\(\)](#) `CFE_Status_t CFE_ES_GetCDSBlockIDByName (`

```
    CFE_ES_CDSHandle_t * BlockIdPtr,  
    const char * BlockName )
```

Get a CDS Block ID associated with a specified CDS Block name.

**Description**

This routine retrieves the CDS Block ID associated with a specified CDS Block name.

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>out</i>	<i>BlockIdPtr</i>	Pointer to variable that is to receive the CDS Block ID (must not be null).
<i>in</i>	<i>BlockName</i>	Pointer to null terminated character string containing a CDS Block name (must not be null).

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_ES_ERR_NAME_NOT_FOUND</a>	Resource Name Error.
<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\\_GetCDSBlockName](#)

```
10.19.2.3 CFE_ES_GetCDSBlockName() CFE_Status_t CFE_ES_GetCDSBlockName (
    char * BlockName,
    CFE_ES_CDSHandle_t BlockId,
    size_t BufferLength )
```

Get a Block name for a specified Block ID.

**Description**

This routine retrieves the cFE Block name associated with a specified Block ID.

**Assumptions, External Events, and Notes:**

In the case of a failure ([CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID](#)), an empty string is returned.

**Parameters**

<i>out</i>	<i>BlockName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the CDS Block name.
<i>in</i>	<i>BlockId</i>	Block ID/Handle of CDS registry entry whose name is being requested.
<i>in</i>	<i>BufferLength</i>	The maximum number of characters, including the null terminator, that can be put into the <i>BlockName</i> buffer. This routine will truncate the name to this length, if necessary.

**Returns**

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

**Return values**

<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](#)

```
10.19.2.4 CFE_ES_RegisterCDS() CFE\_Status\_t CFE_ES_RegisterCDS (
    CFE\_ES\_CDSHandle\_t * CDSHandlePtr,
    size_t BlockSize,
    const char * Name )
```

Reserve space (or re-obtain previously reserved space) in the Critical Data Store (CDS)

**Description**

This routine allocates a block of memory in the Critical Data Store and associates it with the calling Application. The memory can survive an Application restart as well as a Processor Reset.

**Assumptions, External Events, and Notes:**

This function does *not* clear or otherwise initialize/modify the data within the CDS block. If this function returns [CFE\\_ES\\_CDS\\_ALREADY\\_EXISTS](#) the block may already have valid data in it.

If a new CDS block is reserved (either because the name did not exist, or existed as a different size) it is the responsibility of the calling application to fill the CDS block with valid data. This is indicated by a [CFE\\_SUCCESS](#) return code, and in this case the calling application should ensure that it also calls [CFE\\_ES\\_CopyToCDS\(\)](#) to fill the block with valid data.

**Parameters**

<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.
-----------------------------	---

#### Return values

<code>CFE_ES_NOT_IMPLEMENTED</code>	The processor does not support a Critical Data Store.
<code>CFE_ES_CDS_ALREADY_EXISTS</code>	CDS Already Exists.
<code>CFE_ES_CDS_INVALID_SIZE</code>	CDS Invalid Size.
<code>CFE_ES_CDS_INVALID_NAME</code>	CDS Invalid Name.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.
<code>CFE_ES_CDS_INVALID</code>	(return value only verified in coverage test) CDS Invalid.

#### See also

[CFE\\_ES\\_CopyToCDS](#), [CFE\\_ES\\_RestoreFromCDS](#)

#### 10.19.2.5 `CFE_ES_RestoreFromCDS()`

```
CFE_Status_t CFE_ES_RestoreFromCDS (
    void * RestoreToMemory,
    CFE_ES_CDSHandle_t Handle )
```

Recover a block of data from the Critical Data Store (CDS)

#### Description

This routine copies data from the Critical Data Store identified with the `Handle` into the area of memory pointed to by the `RestoreToMemory` pointer. The area of memory to be copied into must be at least as big as the size specified when registering the CDS. The recovery will indicate an error if the data integrity check maintained by the CDS indicates the contents of the CDS have changed. However, the contents will still be copied into the specified area of memory.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>Handle</code>	The handle of the CDS block that was previously obtained from <a href="#">CFE_ES_RegisterCDS</a> .
out	<code>RestoreToMemory</code>	A Pointer to the block of memory (must not be null) that is to be restored with the contents of the CDS. <code>*RestoreToMemory</code> is the contents of the specified CDS.

#### 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_CDS_BLOCK_CRC_ERR</code>	(return value only verified in coverage test) CDS Block CRC Error.
<code>CFE_ES_BAD_ARGUMENT</code>	Bad Argument.

See also

[CFE\\_ES\\_RegisterCDS](#), [CFE\\_ES\\_CopyToCDS](#)

## 10.20 cFE Memory Manager APIs

### Functions

- [CFE\\_Status\\_t CFE\\_ES\\_PoolCreateNoSem \(CFE\\_ES\\_MemHandle\\_t \\*PoolID, void \\*MemPtr, size\\_t Size\)](#)  
*Initializes a memory pool created by an application without using a semaphore during processing.*
- [CFE\\_Status\\_t CFE\\_ES\\_PoolCreate \(CFE\\_ES\\_MemHandle\\_t \\*PoolID, void \\*MemPtr, size\\_t Size\)](#)  
*Initializes a memory pool created by an application while using a semaphore during processing.*
- [CFE\\_Status\\_t CFE\\_ES\\_PoolCreateEx \(CFE\\_ES\\_MemHandle\\_t \\*PoolID, void \\*MemPtr, size\\_t Size, uint16 NumBlockSizes, const size\\_t \\*BlockSizes, bool UseMutex\)](#)  
*Initializes a memory pool created by an application with application specified block sizes.*
- [CFE\\_Status\\_t CFE\\_ES\\_PoolCreateEx\\_WithAlignment \(CFE\\_ES\\_MemHandle\\_t \\*PoolID, void \\*MemPtr, size\\_t Size, uint16 NumBlockSizes, const size\\_t \\*BlockSizes, bool UseMutex, size\\_t Alignment\)](#)  
*Implements CFE\_ES\_PoolCreateEx with added param for alignment added for coverage purposes)*
- [int32 CFE\\_ES\\_PoolDelete \(CFE\\_ES\\_MemHandle\\_t PoolID\)](#)  
*Deletes a memory pool that was previously created.*
- [int32 CFE\\_ES\\_GetPoolBuf \(CFE\\_ES\\_MemPoolBuf\\_t \\*BufPtr, CFE\\_ES\\_MemHandle\\_t Handle, size\\_t Size\)](#)  
*Gets a buffer from the memory pool created by CFE\_ES\_PoolCreate or CFE\_ES\_PoolCreateNoSem.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetPoolBufInfo \(CFE\\_ES\\_MemHandle\\_t Handle, CFE\\_ES\\_MemPoolBuf\\_t BufPtr\)](#)  
*Gets info on a buffer previously allocated via CFE\_ES\_GetPoolBuf.*
- [int32 CFE\\_ES\\_PutPoolBuf \(CFE\\_ES\\_MemHandle\\_t Handle, CFE\\_ES\\_MemPoolBuf\\_t BufPtr\)](#)  
*Releases a buffer from the memory pool that was previously allocated via CFE\_ES\_GetPoolBuf.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetMemPoolStats \(CFE\\_ES\\_MemPoolStats\\_t \\*BufPtr, CFE\\_ES\\_MemHandle\\_t Handle\)](#)  
*Extracts the statistics maintained by the memory pool software.*

### 10.20.1 Detailed Description

### 10.20.2 Function Documentation

**10.20.2.1 CFE\_ES\_GetMemPoolStats()** [CFE\\_Status\\_t CFE\\_ES\\_GetMemPoolStats \(](#)  
[CFE\\_ES\\_MemPoolStats\\_t \\* BufPtr,](#)  
[CFE\\_ES\\_MemHandle\\_t Handle \)](#)

Extracts the statistics maintained by the memory pool software.

#### Description

This routine fills the [CFE\\_ES\\_MemPoolStats\\_t](#) data structure with the statistics maintained by the memory pool software. These statistics can then be telemetered by the calling Application.

#### Assumptions, External Events, and Notes:

None

#### Parameters

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

**Returns**

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

**Return values**

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

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_PoolCreateEx](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_PutPoolBuf](#)

**10.20.2.2 CFE\_ES\_GetPoolBuf()** *int32* CFE\_ES\_GetPoolBuf (

```
    CFE_ES_MemPoolBuf_t * BufPtr,  
    CFE_ES_MemHandle_t Handle,  
    size_t Size )
```

Gets a buffer from the memory pool created by [CFE\\_ES\\_PoolCreate](#) or [CFE\\_ES\\_PoolCreateNoSem](#).

**Description**

This routine obtains a block of memory from the memory pool supplied by the calling application.

**Assumptions, External Events, and Notes:**

1. The size allocated from the memory pool is, at a minimum, 12 bytes more than requested.

**Parameters**

<i>out</i>	<i>BufPtr</i>	A pointer to the Application's pointer (must not be null) in which will be stored the address of the allocated memory buffer. *BufPtr is the address of the requested buffer.
<i>in</i>	<i>Handle</i>	The handle to the memory pool as returned by <a href="#">CFE_ES_PoolCreate</a> or <a href="#">CFE_ES_PoolCreateNoSem</a> .
<i>in</i>	<i>Size</i>	The size of the buffer requested. NOTE: The size allocated may be larger.

**Returns**

Bytes Allocated, or error code [cFE Return Code Defines](#)

**Return values**

<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_ES_ERR_MEM_BLOCK_SIZE</i>	Memory Block Size Error.
<i>CFE_ES_BAD_ARGUMENT</i>	Bad Argument.

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_PoolCreateEx](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#), [CFE\\_ES\\_GetPoolBufInfo](#)

**10.20.2.3 CFE\_ES\_GetPoolBufInfo()** `CFE_Status_t CFE_ES_GetPoolBufInfo (`

```
    CFE_ES_MemHandle_t Handle,
    CFE_ES_MemPoolBuf_t BufPtr )
```

Gets info on a buffer previously allocated via [CFE\\_ES\\_GetPoolBuf](#).

**Description**

This routine gets info on a buffer in the memory pool.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Handle</i>	The handle to the memory pool as returned by <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](#)

**10.20.2.4 CFE\_ES\_PoolCreate()** `CFE_Status_t CFE_ES_PoolCreate (`

```
    CFE_ES_MemHandle_t * PoolID,
    void * MemPtr,
    size_t Size )
```

Initializes a memory pool created by an application while using a semaphore during processing.

**Description**

This routine initializes a pool of memory supplied by the calling application. When a memory pool created by this routine is processed, mutex handling will be performed.

**Assumptions, External Events, and Notes:**

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

**Parameters**

<i>out</i>	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
<i>in</i>	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The <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](#)

```
10.20.2.5 CFE_ES_PoolCreateEx() CFE\_Status\_t CFE_ES_PoolCreateEx (
    CFE\_ES\_MemHandle\_t * PoolID,
    void * MemPtr,
    size_t Size,
    uint16 NumBlockSizes,
    const size_t * BlockSizes,
    bool UseMutex )
```

Initializes a memory pool created by an application with application specified block sizes.

**Description**

This routine initializes a pool of memory supplied by the calling application.

**Assumptions, External Events, and Notes:**

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

**Parameters**

<i>out</i>	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
<i>in</i>	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The <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.
<i>in</i>	<i>NumBlockSizes</i>	The number of different block sizes specified in the <i>BlockSizes</i> array. If set larger than <a href="#">CFE_PLATFORM_ES_POOL_MAX_BUCKETS</a> , <a href="#">CFE_ES_BAD_ARGUMENT</a> will be returned. If BlockSizes is null and NumBlockSizes is 0, NubBlockSizes will be set to <a href="#">CFE_PLATFORM_ES_POOL_MAX_BUCKETS</a> .
<i>in</i>	<i>BlockSizes</i>	Pointer to an array of sizes to be used instead of the default block sizes specified by <a href="#">CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01</a> through <a href="#">CFE_PLATFORM_ES_MAX_BLOCK_SIZE</a> . If the pointer is equal to NULL, the default block sizes are used.
<i>in</i>	<i>UseMutex</i>	Flag indicating whether the new memory pool will be processing with mutex handling or not. Valid parameter values are <a href="#">CFE_ES_USE_MUTEX</a> and <a href="#">CFE_ES_NO_MUTEX</a>

**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_NO_RESOURCE_IDS_AVAILABLE</a>	Resource ID is not available.
<a href="#">CFE_STATUS_EXTERNAL_RESOURCE_FAIL</a>	(return value only verified in coverage test) External failure.

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#)

**10.20.2.6 CFE\_ES\_PoolCreateEx\_WithAlignment()** [CFE\\_Status\\_t](#) CFE\_ES\_PoolCreateEx\_WithAlignment (

```
    CFE\_ES\_MemHandle\_t * PoolID,
    void * MemPtr,
    size_t Size,
    uint16 NumBlockSizes,
    const size_t * BlockSizes,
    bool UseMutex,
    size_t Alignment )
```

Implements [CFE\\_ES\\_PoolCreateEx](#) with added param for alignment added for coverage purposes)

**Description**

This routine initializes a pool of memory supplied by the calling application.

**Assumptions, External Events, and Notes:**

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

**Parameters**

out	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
in	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The <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.
in	<i>NumBlockSizes</i>	The number of different block sizes specified in the BlockSizes array. If set larger than <a href="#">CFE_PLATFORM_ES_POOL_MAX_BUCKETS</a> , <a href="#">CFE_ES_BAD_ARGUMENT</a> will be returned. If BlockSizes is null and NumBlockSizes is 0, NubBlockSizes will be set to <a href="#">CFE_PLATFORM_ES_POOL_MAX_BUCKETS</a> .
in	<i>BlockSizes</i>	Pointer to an array of sizes to be used instead of the default block sizes specified by <a href="#">CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01</a> through <a href="#">CFE_PLATFORM_ES_MAX_BLOCK_SIZE</a> . 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 <a href="#">CFE_ES_USE_MUTEX</a> and <a href="#">CFE_ES_NO_MUTEX</a>
out	<i>Alignment</i>	Size of struct alignment

**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_NO_RESOURCE_IDS_AVAILABLE</a>	Resource ID is not available.
<a href="#">CFE_STATUS_EXTERNAL_RESOURCE_FAIL</a>	(return value only verified in coverage test) External failure.

**See also**

[CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#)

#### 10.20.2.7 CFE\_ES\_PoolCreateNoSem()

```
CFE_Status_t CFE_ES_PoolCreateNoSem (
    CFE_ES_MemHandle_t * PoolID,
    void * MemPtr,
    size_t Size )
```

Initializes a memory pool created by an application without using a semaphore during processing.

### Description

This routine initializes a pool of memory supplied by the calling application. When a memory pool created by this routine is processed, no mutex handling is performed.

### Assumptions, External Events, and Notes:

1. The size of the pool must be an integral number of 32-bit words
2. The start address of the pool must be 32-bit aligned
3. 168 bytes are used for internal bookkeeping, therefore, they will not be available for allocation.

### Parameters

<i>out</i>	<i>PoolID</i>	A pointer to the variable the caller wishes to have the memory pool handle kept in (must not be null). PoolID is the memory pool handle.
<i>in</i>	<i>MemPtr</i>	A Pointer to the pool of memory created by the calling application (must not be null). This address must be aligned suitably for the processor architecture. The <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\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateEx](#), [CFE\\_ES\\_GetPoolBuf](#), [CFE\\_ES\\_PutPoolBuf](#), [CFE\\_ES\\_GetMemPoolStats](#)

### 10.20.2.8 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](#)

**10.20.2.9 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](#)

## 10.21 cFE Performance Monitor APIs

### Macros

- `#define CFE_ES_PerfLogEntry(id) (CFE_ES_PerfLogAdd(id, 0))`  
*Entry marker for use with Software Performance Analysis Tool.*
- `#define CFE_ES_PerfLogExit(id) (CFE_ES_PerfLogAdd(id, 1))`  
*Exit marker for use with Software Performance Analysis Tool.*

### Functions

- `void CFE_ES_PerfLogAdd (uint32 Marker, uint32 EntryExit)`  
*Adds a new entry to the data buffer.*

#### 10.21.1 Detailed Description

#### 10.21.2 Macro Definition Documentation

##### 10.21.2.1 CFE\_ES\_PerfLogEntry `#define CFE_ES_PerfLogEntry(` `id  )  (CFE_ES_PerfLogAdd(id,  0))`

Entry marker for use with Software Performance Analysis Tool.

#### Description

This macro logs the entry or start event/marker for the specified entry `id`. This macro, in conjunction with the [CFE\\_ES\\_PerfLogExit](#), is used by the Software Performance Analysis tool.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<code>id</code>	Identifier of the specific event or marker.
----	-----------------	---

See also

[CFE\\_ES\\_PerfLogExit](#), [CFE\\_ES\\_PerfLogAdd](#)

Definition at line 1508 of file `cfe_es.h`.

##### 10.21.2.2 CFE\_ES\_PerfLogExit `#define CFE_ES_PerfLogExit(` `id  )  (CFE_ES_PerfLogAdd(id,  1))`

Exit marker for use with Software Performance Analysis Tool.

**Description**

This macro logs the exit or end event/marker for the specified entry *id*. This macro, in conjunction with the [CFE\\_ES\\_PerfLogEntry](#), is used by the Software Performance Analysis tool.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>id</i>	Identifier of the specific event or marker.
----	-----------	---

**See also**

[CFE\\_ES\\_PerfLogEntry](#), [CFE\\_ES\\_PerfLogAdd](#)

Definition at line 1527 of file cfe\_es.h.

**10.21.3 Function Documentation**

**10.21.3.1 CFE\_ES\_PerfLogAdd()** void CFE\_ES\_PerfLogAdd (

```
    uint32 Marker,
    uint32 EntryExit )
```

Adds a new entry to the data buffer.

Function called by [CFE\\_ES\\_PerfLogEntry](#) and [CFE\\_ES\\_PerfLogExit](#) macros

**Description**

This function logs the entry and exit marker for the specified *id*. This function is used by the Software Performance Analysis tool.

**Assumptions, External Events, and Notes:**

Marker limited to the range of 0 to [CFE\\_MISSION\\_ES\\_PERF\\_MAX\\_IDS](#) - 1. Any performance ids outside of this range will be ignored and will be flagged as an error.

This function implements a circular buffer using an array. DataStart points to first stored entry DataEnd points to next available entry if DataStart == DataEnd then the buffer is either empty or full depending on the value of the DataCount Time is stored as 2 32 bit integers, (TimerLower32, TimerUpper32): TimerLower32 is the current value of the hardware timer register. TimerUpper32 is the number of times the timer has rolled over.

**Parameters**

in	<i>Marker</i>	Identifier of the specific event or marker.
in	<i>EntryExit</i>	Used to specify Entry(0) or Exit(1)

See also

[CFE\\_ES\\_PerfLogEntry](#), [CFE\\_ES\\_PerfLogExit](#)

## 10.22 cFE Generic Counter APIs

### Functions

- [CFE\\_Status\\_t CFE\\_ES\\_RegisterGenCounter \(CFE\\_ES\\_CounterId\\_t \\*CounterIdPtr, const char \\*CounterName\)](#)  
*Register a generic counter.*
- [CFE\\_Status\\_t CFE\\_ES\\_DeleteGenCounter \(CFE\\_ES\\_CounterId\\_t CounterId\)](#)  
*Delete a generic counter.*
- [CFE\\_Status\\_t CFE\\_ES\\_IncrementGenCounter \(CFE\\_ES\\_CounterId\\_t CounterId\)](#)  
*Increments the specified generic counter.*
- [CFE\\_Status\\_t CFE\\_ES\\_SetGenCount \(CFE\\_ES\\_CounterId\\_t CounterId, uint32 Count\)](#)  
*Set the specified generic counter.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetGenCount \(CFE\\_ES\\_CounterId\\_t CounterId, uint32 \\*Count\)](#)  
*Get the specified generic counter count.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetGenCounterIDByName \(CFE\\_ES\\_CounterId\\_t \\*CounterIdPtr, const char \\*CounterName\)](#)  
*Get the Id associated with a generic counter name.*
- [CFE\\_Status\\_t CFE\\_ES\\_GetGenCounterName \(char \\*CounterName, CFE\\_ES\\_CounterId\\_t CounterId, size\\_t BufferLength\)](#)  
*Get a Counter name for a specified Counter ID.*

#### 10.22.1 Detailed Description

#### 10.22.2 Function Documentation

##### 10.22.2.1 CFE\_ES\_DeleteGenCounter() [CFE\\_Status\\_t CFE\\_ES\\_DeleteGenCounter \(CFE\\_ES\\_CounterId\\_t CounterId \)](#)

Delete a generic counter.

#### Description

This routine deletes a previously registered generic counter.

#### Assumptions, External Events, and Notes:

None.

#### Parameters

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

#### 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\\_IncrementGenCounter](#), [CFE\\_ES\\_RegisterGenCounter](#), [CFE\\_ES\\_SetGenCount](#), [CFE\\_ES\\_GetGenCount](#),  
[CFE\\_ES\\_GetGenCounterIDByName](#)

**10.22.2.2 CFE\_ES\_GetGenCount()** [CFE\\_Status\\_t](#) CFE\_ES\_GetGenCount (

```
    CFE_ES_CounterId_t CounterId,
    uint32 * Count )
```

Get the specified generic counter count.

**Description**

This routine gets the value of a generic counter.

**Assumptions, External Events, and Notes:**

None.

**Parameters**

in	<i>CounterId</i>	The Counter to get the value from.
out	<i>Count</i>	Buffer to store value of the Counter (must not be null).

**Returns**

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

**Return values**

<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\\_IncrementGenCounter](#),  
[CFE\\_ES\\_GetGenCounterIDByName](#)

**10.22.2.3 CFE\_ES\_GetGenCounterIDByName()** [CFE\\_Status\\_t](#) CFE\_ES\_GetGenCounterIDByName (

```
    CFE_ES_CounterId_t * CounterIdPtr,
    const char * CounterName )
```

Get the Id associated with a generic counter name.

**Description**

This routine gets the Counter Id for a generic counter specified by name.

**Assumptions, External Events, and Notes:**

None.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_ES_ERR_NAME_NOT_FOUND</i></a>	Resource Name Error.
<a href="#"><i>CFE_ES_BAD_ARGUMENT</i></a>	Bad Argument.

**See also**

[CFE\\_ES\\_GetGenCounterName](#)

**10.22.2.4 CFE\_ES\_GetGenCounterName()** [\*CFE\\_Status\\_t\*](#) *CFE\_ES\_GetGenCounterName* (

```
    char * CounterName,
    CFE\_ES\_CounterId\_t CounterId,
    size_t BufferLength )
```

Get a Counter name for a specified Counter ID.

**Description**

This routine retrieves the cFE Counter name associated with a specified Counter ID.

**Assumptions, External Events, and Notes:**

In the case of a failure ([\*CFE\\_ES\\_ERR\\_RESOURCEID\\_NOT\\_VALID\*](#)), an empty string is returned.

**Parameters**

<i>out</i>	<i>CounterName</i>	Pointer to a character array (must not be null) of at least <i>BufferLength</i> in size that will be filled with the Counter name.
<i>in</i>	<i>CounterId</i>	ID of Counter whose name is being requested.
<i>in</i>	<i>BufferLength</i>	The maximum number of characters, including the null terminator (must not be zero), that can be put into the <i>CounterName</i> buffer. This routine will truncate the name to this length, if necessary.

**Returns**

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

**Return values**

<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.
<a href="#"><i>CFE_ES_BAD_ARGUMENT</i></a>	Bad Argument.

**See also**

[CFE\\_ES\\_GetGenCounterIDByName](#)

**10.22.2.5 CFE\_ES\_IncrementGenCounter()** [\*CFE\\_Status\\_t\*](#) CFE\_ES\_IncrementGenCounter ( [\*CFE\\_ES\\_CounterId\\_t\*](#) CounterId )

Increments the specified generic counter.

**Description**

This routine increments the specified generic counter.

**Assumptions, External Events, and Notes:**

None.

**Parameters**

<a href="#"><i>in</i></a>	<i>CounterId</i>	The Counter to be incremented.
---------------------------	------------------	--------------------------------

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_ES_BAD_ARGUMENT</i></a>	Bad Argument.

**See also**

[CFE\\_ES\\_RegisterGenCounter](#), [CFE\\_ES\\_DeleteGenCounter](#), [CFE\\_ES\\_SetGenCount](#), [CFE\\_ES\\_GetGenCount](#),  
[CFE\\_ES\\_GetGenCounterIDByName](#)

**10.22.2.6 CFE\_ES\_RegisterGenCounter()** [\*CFE\\_Status\\_t\*](#) CFE\_ES\_RegisterGenCounter ( [\*CFE\\_ES\\_CounterId\\_t\*](#) \* CounterIdPtr,  
const char \* CounterName )

Register a generic counter.

**Description**

This routine registers a generic thread-safe counter which can be used for inter-task management.

**Assumptions, External Events, and Notes:**

The initial value of all newly registered counters is 0.

**Parameters**

<i>out</i>	<i>CounterIdPtr</i>	Buffer to store the Counter Id of the newly created counter (must not be null).
<i>in</i>	<i>CounterName</i>	The Name of the generic counter (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_BAD_ARGUMENT</i></a>	Bad Argument.
<a href="#"><i>CFE_ES_ERR_DUPLICATE_NAME</i></a>	Duplicate Name Error.
<a href="#"><i>CFE_ES_NO_RESOURCE_IDS_AVAILABLE</i></a>	Resource ID is not available.

**See also**

[CFE\\_ES\\_IncrementGenCounter](#), [CFE\\_ES\\_DeleteGenCounter](#), [CFE\\_ES\\_SetGenCount](#), [CFE\\_ES\\_GetGenCount](#), [CFE\\_ES\\_GetGenCounterIDByName](#)

**10.22.2.7 CFE\_ES\_SetGenCount()** [\*CFE\\_Status\\_t\*](#) *CFE\_ES\_SetGenCount* (

```
    CFE_ES_CounterId_t CounterId,
    uint32 Count )
```

Set the specified generic counter.

**Description**

This routine sets the specified generic counter to the specified value.

**Assumptions, External Events, and Notes:**

None.

**Parameters**

<i>in</i>	<i>CounterId</i>	The Counter to be set.
<i>in</i>	<i>Count</i>	The new value of the Counter.

**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\\_IncrementGenCounter](#), [CFE\\_ES\\_GetGenCount](#),  
[CFE\\_ES\\_GetGenCounterIDByName](#)

## 10.23 cFE Registration APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_EVS\\_Register](#) (const void \*Filters, uint16 NumEventFilters, uint16 FilterScheme)

*Register an application for receiving event services.*

### 10.23.1 Detailed Description

### 10.23.2 Function Documentation

**10.23.2.1 CFE\_EVS\_Register()** [CFE\\_Status\\_t](#) CFE\_EVS\_Register (

```
    const void * Filters,
    uint16 NumEventFilters,
    uint16 FilterScheme )
```

Register an application for receiving event services.

**Description**

This routine registers an application with event services and allocates/initializes the internal data structures used to support this application's events. An application may not send events unless it has called this routine. The routine also accepts a filter array structure for applications requiring event filtering. In the current implementation of the EVS, only the binary filtering scheme is supported. See section TBD of the cFE Application Programmer's Guide for a description of the behavior of binary filters. Applications may call [CFE\\_EVS\\_Register](#) more than once, but each call will wipe out all filters registered by previous calls (filter registration is NOT cumulative).

**Assumptions, External Events, and Notes:**

Note: Event filters can be added, deleted or modified by ground commands. All filtering schemes include a default setting that results in no filtering (such as [CFE\\_EVS\\_NO\\_FILTER](#) for binary filters).

**Filter Scheme:** Binary

**Code:** CFE\_EVS\_EventFilter\_BINARY

**Filter Structure:**

```
typedef struct CFE_EVS_BinFilter {
    uint16 EventID,
    uint16 Mask ;
} CFE_EVS_BinFilter_t;
```

**Parameters**

in	<i>Filters</i>	Pointer to an array of event message filters, or NULL if no filtering is desired. The structure of an event message filter depends on the FilterScheme selected. (see Filter Schemes mentioned above)
in	<i>NumEventFilters</i>	The number of event message filters included in this call. This must be less than or equal to the maximum number of events allowed per application ( <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**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_EVS_APP_FILTER_OVERLOAD</a>	Application Filter Overload.
<a href="#">CFE_EVS_UNKNOWN_FILTER</a>	Unknown Filter.
<a href="#">CFE_EVS_APP_ILLEGAL_APP_ID</a>	Illegal Application ID.
<a href="#">CFE_ES_BAD_ARGUMENT</a>	Bad Argument.

Referenced by CF\_ApplInit().

## 10.24 cFE Send Event APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_EVS\\_SendEvent \(uint16 EventID, CFE\\_EVS\\_EventType\\_Enum\\_t EventType, const char \\*Spec,...\) OS\\_PRINTF\(3\)](#)  
*Generate a software event.*
- [CFE\\_Status\\_t CFE\\_Status\\_t CFE\\_EVS\\_SendEventWithAppID \(uint16 EventID, CFE\\_EVS\\_EventType\\_Enum\\_t EventType, CFE\\_ES\\_AppId\\_t AppID, const char \\*Spec,...\) OS\\_PRINTF\(4\)](#)  
*Generate a software event given the specified Application ID.*
- [CFE\\_Status\\_t CFE\\_Status\\_t CFE\\_Status\\_t CFE\\_EVS\\_SendTimedEvent \(CFE\\_TIME\\_SysTime\\_t Time, uint16 EventID, CFE\\_EVS\\_EventType\\_Enum\\_t EventType, const char \\*Spec,...\) OS\\_PRINTF\(4\)](#)  
*Generate a software event with a specific time tag.*

### 10.24.1 Detailed Description

#### 10.24.2 Function Documentation

**10.24.2.1 CFE\_EVS\_SendEvent()** [CFE\\_Status\\_t CFE\\_EVS\\_SendEvent \(](#)  
[uint16 EventID,](#)  
[CFE\\_EVS\\_EventType\\_Enum\\_t EventType,](#)  
[const char \\* Spec,](#)  
[... \)](#)

Generate a software event.

## Description

This routine generates a software event message. If the EventID is not filtered, the event will be sent as a software bus message, optionally logged in the local event log, and optionally sent as an ASCII text string out the enabled output port(s).

### Assumptions, External Events, and Notes:

This API only works within the context of a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE\\_ES\\_WriteToSysLog](#) can be used for reporting.

## Parameters

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The <code>EventID</code> is defined and supplied by the application sending the event.
in	<i>EventType</i>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"> <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

<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\\_SendEventWithAppID](#), [CFE\\_EVS\\_SendTimedEvent](#)

Referenced by `CF_AbandonCmd()`, `CF_AppInit()`, `CF_AppMain()`, `CF_AppPipe()`, `CF_CancelCmd()`, `CF_CFDP_AllocChunkList()`, `CF_CFDP_CheckAckNakCount()`, `CF_CFDP_FinishTransaction()`, `CF_CFDP_GetMoveTarget()`, `CF_CFDP_InitEngine()`, `CF_CFDP_MsgOutGet()`, `CF_CFDP_PlaybackDir()`, `CF_CFDP_PlaybackDir_Initiate()`, `CF_CFDP_R2_SubstateRecvFinAck()`, `CF_CFDP_R_CalcCrcChunk()`, `CF_CFDP_R_CalcCrcStart()`, `CF_CFDP_R_CalcCrcStop()`

CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_ReceivePdu(), CF\_CFDP\_RecvAck(), CF\_CFDP\_RecvEof(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvFin(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvNak(), CF\_CFDP\_RecvPh(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_TxFile(), CF\_CFDP\_TxFile\_Initiate(), CF\_CheckTables(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DisableEngineCmd(), CF\_DoChanAction(), CF\_DoEnableDisablePolldir(), CF\_DoPurgeQueue(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(), CF\_FreezeCmd(), CF\_GetSetParamCmd(), CF\_NoopCmd(), CF\_PlaybackDirCmd(), CF\_ProcessGroundCommand(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_TableInit(), CF\_ThawCmd(), CF\_TsnChanAction(), CF\_TxFileCmd(), CF\_ValidateConfigTable(), CF\_WrappedClose(), CF\_WriteHistoryEntryToFile(), and CF\_WriteQueueCmd().

**10.24.2.2 CFE\_EVS\_SendEventWithAppID()** `CFE_Status_t CFE_Status_t CFE_EVS_SendEventWithAppID (`  
 `uint16 EventID,`  
 `CFE_EVS_EventType_Enum_t EventType,`  
 `CFE_ES_AppId_t AppID,`  
 `const char * Spec,`  
 `... )`

Generate a software event given the specified Application ID.

#### Description

This routine generates a software event message. If the EventID is not filtered, the event will be sent as a software bus message, optionally logged in the local event log, and optionally sent as an ASCII text string out the enabled output port(s). Note that this function should really only be used from within an API in order to preserve the context of an Application's event. In general, [CFE\\_EVS\\_SendEvent](#) should be used.

#### Assumptions, External Events, and Notes:

The Application ID must correspond to a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE\\_ES\\_WriteToSysLog](#) can be used for reporting.

#### Parameters

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The <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.

**Parameters**

in	<i>Spec</i>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter <code>CFE_MISSION_EVS_MAX_MESSAGE_LENGTH</code> . Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.
----	-------------	--

**Returns**

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

**Return values**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_EVS_APP_NOT_REGISTERED</code>	Application Not Registered.
<code>CFE_EVS_APP_ILLEGAL_APP_ID</code>	Illegal Application ID.
<code>CFE_EVS_INVALID_PARAMETER</code>	Invalid Pointer.

**See also**

[CFE\\_EVS\\_SendEvent](#), [CFE\\_EVS\\_SendTimedEvent](#)

```
10.24.2.3 CFE_EVS_SendTimedEvent() CFE_Status_t CFE_Status_t CFE_Status_t CFE_Status_t CFE_EVS_SendTimedEvent←
Event ( CFE_TIME_SysTime_t Time,
        uint16 EventID,
        CFE_EVS_EventType_Enum_t EventType,
        const char * Spec,
        ... )
```

Generate a software event with a specific time tag.

**Description**

This routine is the same as [CFE\\_EVS\\_SendEvent](#) except that the caller specifies the event time instead of having the EVS use the current spacecraft time. This routine should be used in situations where an error condition is detected at one time, but the event message is reported at a later time.

**Assumptions, External Events, and Notes:**

This API only works within the context of a registered application or core service. For messages outside the context of a registered application (for example early in app initialization or if registration fails) [CFE\\_ES\\_WriteToSysLog](#) can be used for reporting.

**Parameters**

in	<i>Time</i>	The time to include in the event. This will usually be a time returned by the function <a href="#">CFE_TIME_GetTime</a> .
----	-------------	---

**Parameters**

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The <code>EventID</code> is defined and supplied by the application sending the event.
in	<i>EventType</i>	A numeric literal used to classify an event, one of: <ul style="list-style-type: none"><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>
in	<i>Spec</i>	A pointer to a null terminated text string (must not be null) describing the output format for the event. This is the same type of format string used for the ANSI <code>printf</code> function. Nominally the post-conversion string is limited to 80 characters, but this limit is configurable through the parameter <code>CFE_MISSION_EVS_MAX_MESSAGE_LENGTH</code> . Characters beyond the limit will be truncated. Do not use floating point conversions (f, e, E, g, and G) in the format string unless your application will be running in a system that supports floating point arithmetic. Do not use non-printable characters (\t, \n, etc.) in the format string; they will mess up the formatting when the events are displayed on the ground system.

**Returns**

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

**Return values**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_EVS_APP_NOT_REGISTERED</code>	Application Not Registered.
<code>CFE_EVS_APP_ILLEGAL_APP_ID</code>	Illegal Application ID.
<code>CFE_EVS_INVALID_PARAMETER</code>	Invalid Pointer.

**See also**

[CFE\\_EVS\\_SendEvent](#), [CFE\\_EVS\\_SendEventWithAppID](#)

## 10.25 cFE Reset Event Filter APIs

**Functions**

- `CFE_Status_t CFE_EVS_ResetFilter (uint16 EventID)`  
*Resets the calling application's event filter for a single event ID.*
- `CFE_Status_t CFE_EVS_ResetAllFilters (void)`  
*Resets all of the calling application's event filters.*

### 10.25.1 Detailed Description

### 10.25.2 Function Documentation

**10.25.2.1 CFE\_EVS\_ResetAllFilters()** `CFE_Status_t CFE_EVS_ResetAllFilters ( void )`

Resets all of the calling application's event filters.

#### Description

This routine resets all the calling application's event filter counters to zero, providing a quick and convenient method for resetting event filters.

#### Assumptions, External Events, and Notes:

None

#### Returns

Execution status below or from [CFE\\_ES\\_GetAppID](#), see [cFE Return Code Defines](#)

#### Return values

<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](#)

**10.25.2.2 CFE\_EVS\_ResetFilter()** `CFE_Status_t CFE_EVS_ResetFilter ( uint16 EventID )`

Resets the calling application's event filter for a single event ID.

#### Description

Resets the filter such that the next event is treated like the first. For example, if the filter was set to only send the first event, the next event following the reset would be sent.

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>EventID</i>	A numeric literal used to uniquely identify an application event. The EventID is defined and supplied by the application sending the event.
----	----------------	---

#### Returns

Execution status below or from [CFE\\_ES\\_GetAppID](#), see [cFE Return Code Defines](#)

**Return values**

<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](#)

## 10.26 cFE File Header Management APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_FS\\_ReadHeader \(CFE\\_FS\\_Header\\_t \\*Hdr, osal\\_id\\_t FileDes\)](#)  
*Read the contents of the Standard cFE File Header.*
- [void CFE\\_FS\\_InitHeader \(CFE\\_FS\\_Header\\_t \\*Hdr, const char \\*Description, uint32 SubType\)](#)  
*Initializes the contents of the Standard cFE File Header.*
- [CFE\\_Status\\_t CFE\\_FS\\_WriteHeader \(osal\\_id\\_t FileDes, CFE\\_FS\\_Header\\_t \\*Hdr\)](#)  
*Write the specified Standard cFE File Header to the specified file.*
- [CFE\\_Status\\_t CFE\\_FS\\_SetTimestamp \(osal\\_id\\_t FileDes, CFE\\_TIME\\_SysTime\\_t NewTimestamp\)](#)  
*Modifies the Time Stamp field in the Standard cFE File Header for the specified file.*

### 10.26.1 Detailed Description

### 10.26.2 Function Documentation

**10.26.2.1 CFE\_FS\_InitHeader()** void CFE\_FS\_InitHeader (

<a href="#">CFE_FS_Header_t</a> *	<i>Hdr,</i>
const char *	<i>Description,</i>
uint32	<i>SubType</i> )

Initializes the contents of the Standard cFE File Header.

**Description**

This API will clear the specified [CFE\\_FS\\_Header\\_t](#) variable and initialize the description field with the specified value

**Parameters**

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

**See also**

[CFE\\_FS\\_WriteHeader](#)

**10.26.2.2 CFE\_FS\_ReadHeader()** `CFE_Status_t CFE_FS_ReadHeader (`  
    `CFE_FS_Header_t * Hdr,`  
    `osal_id_t FileDes )`

Read the contents of the Standard cFE File Header.

#### Description

This API will fill the specified `CFE_FS_Header_t` variable with the contents of the Standard cFE File Header of the file identified by the given File Descriptor.

#### Assumptions, External Events, and Notes:

1. The File has already been successfully opened using `OS_OpenCreate` and the caller has a legitimate File Descriptor.
2. File offset behavior: Agnostic on entry since it will move the offset to the start of the file, on success the offset will be at the end of the header, undefined offset behavior for error cases.

#### Parameters

out	<code>Hdr</code>	Pointer to a variable of type <code>CFE_FS_Header_t</code> (must not be null) that will be filled with the contents of the Standard cFE File Header. <code>*Hdr</code> is the contents of the Standard cFE File Header for the specified file.
in	<code>FileDes</code>	File Descriptor obtained from a previous call to <code>OS_OpenCreate</code> that is associated with the file whose header is to be read.

#### Returns

Bytes read or error status from OSAL

#### Return values

<code>CFE_FS_BAD_ARGUMENT</code>	Bad Argument.
----------------------------------	---------------

#### Note

This function invokes OSAL API routines and the current implementation may return OSAL error codes to the caller if failure occurs. In a future version of CFE, the status codes will be converted to a value in [cFE Return Code Defines](#).

#### See also

[CFE\\_FS\\_WriteHeader](#)

**10.26.2.3 CFE\_FS\_SetTimestamp()** `CFE_Status_t CFE_FS_SetTimestamp (`  
    `osal_id_t FileDes,`  
    `CFE_TIME_SysTime_t NewTimestamp )`

Modifies the Time Stamp field in the Standard cFE File Header for the specified file.

#### Description

This API will modify the `timestamp` found in the Standard cFE File Header of the specified file. The timestamp will be replaced with the time specified by the caller.

**Assumptions, External Events, and Notes:**

1. The File has already been successfully opened using [OS\\_OpenCreate](#) and the caller has a legitimate File Descriptor.
2. The NewTimestamp field has been filled appropriately by the Application.
3. File offset behavior: Agnostic on entry since it will move the offset, on success the offset will be at the end of the time stamp, undefined offset behavior for error cases.

**Parameters**

in	<i>FileDes</i>	File Descriptor obtained from a previous call to <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](#).

**10.26.2.4 CFE\_FS\_WriteHeader()** [CFE\\_Status\\_t](#) CFE\_FS\_WriteHeader (   
 osal\_id\_t *FileDes*,  
 CFE\_FS\_Header\_t \* *Hdr* )

Write the specified Standard cFE File Header to the specified file.

**Description**

This API will output the specified [CFE\\_FS\\_Header\\_t](#) variable, with some fields automatically updated, to the specified file as the Standard cFE File Header. This API will automatically populate the following fields in the specified [CFE\\_FS\\_Header\\_t](#):

1. [ContentType](#) - Filled with 0x63464531 ('cFE1')
2. [Length](#) - Filled with the sizeof(CFE\_FS\_Header\_t)
3. [SpacecraftID](#) - Filled with the Spacecraft ID
4. [ProcessorID](#) - Filled with the Processor ID
5. [ApplicationID](#) - Filled with the Application ID
6. [TimeSeconds](#) - Filled with the Time, in seconds, as obtained by [CFE\\_TIME\\_GetTime](#)
7. [TimeSubSeconds](#) - Filled with the Time, subseconds, as obtained by [CFE\\_TIME\\_GetTime](#)

**Assumptions, External Events, and Notes:**

1. The File has already been successfully opened using [OS\\_OpenCreate](#) and the caller has a legitimate File Descriptor.
2. The SubType field has been filled appropriately by the Application.
3. The Description field has been filled appropriately by the Application.
4. File offset behavior: Agnostic on entry since it will move the offset to the start of the file, on success the offset will be at the end of the header, undefined offset behavior for error cases.

**Parameters**

in	<i>FileDes</i>	File Descriptor obtained from a previous call to <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](#)

## 10.27 cFE File Utility APIs

**Functions**

- const char \* [CFE\\_FS\\_GetDefaultMountPoint](#) ([CFE\\_FS\\_FileCategory\\_t](#) FileCategory)  
*Get the default virtual mount point for a file category.*
- const char \* [CFE\\_FS\\_GetDefaultExtension](#) ([CFE\\_FS\\_FileCategory\\_t](#) FileCategory)  
*Get the default filename extension for a file category.*
- int32 [CFE\\_FS\\_ParseInputFileNameEx](#) (char \*OutputBuffer, const char \*InputBuffer, size\_t OutputBufSize, size\_t InputBufSize, const char \*DefaultInput, const char \*DefaultPath, const char \*DefaultExtension)  
*Parse a filename input from an input buffer into a local buffer.*
- int32 [CFE\\_FS\\_ParseInputFileName](#) (char \*OutputBuffer, const char \*InputName, size\_t OutputBufSize, [CFE\\_FS\\_FileCategory\\_t](#) FileCategory)  
*Parse a filename string from the user into a local buffer.*
- [CFE\\_Status\\_t CFE\\_FS\\_ExtractFilenameFromPath](#) (const char \*OriginalPath, char \*FileNameOnly)  
*Extracts the filename from a unix style path and filename string.*

- `int32 CFE_FS_BackgroundFileDumpRequest (CFE_FS_FileWriteMetaData_t *Meta)`  
*Register a background file dump request.*
- `bool CFE_FS_BackgroundFileDumpIsPending (const CFE_FS_FileWriteMetaData_t *Meta)`  
*Query if a background file write request is currently pending.*

### 10.27.1 Detailed Description

### 10.27.2 Function Documentation

#### 10.27.2.1 CFE\_FS\_BackgroundFileDumpIsPending() `bool CFE_FS_BackgroundFileDumpIsPending (`

```
    const CFE_FS_FileWriteMetaData_t * Meta )
```

Query if a background file write request is currently pending.

##### Description

This returns "true" while the request is on the background work queue. This returns "false" once the request is complete and removed from the queue.

##### Assumptions, External Events, and Notes:

None

##### Parameters

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

##### Returns

boolean value indicating if request is already pending

##### Return values

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

#### 10.27.2.2 CFE\_FS\_BackgroundFileDumpRequest() `int32 CFE_FS_BackgroundFileDumpRequest (`

```
    CFE_FS_FileWriteMetaData_t * Meta )
```

Register a background file dump request.

##### Description

Puts the previously-initialized metadata into the pending request queue

##### Assumptions, External Events, and Notes:

Metadata structure should be stored in a persistent memory area (not on stack) as it must remain accessible by the file writer task throughout the asynchronous job operation.

**Parameters**

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

**Returns**

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

**Return values**

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

**10.27.2.3 CFE\_FS\_ExtractFilenameFromPath()** `CFE_Status_t CFE_FS_ExtractFilenameFromPath (`

```
    const char * OriginalPath,  
    char * FileNameOnly )
```

Extracts the filename from a unix style path and filename string.

**Description**

This API will take the original unix path/filename combination and extract the base filename. Example: Given the path/filename : "/cf/apps/myapp.o.gz" this function will return the filename: "myapp.o.gz".

**Assumptions, External Events, and Notes:**

1. The paths and filenames used here are the standard unix style filenames separated by "/" characters.
2. The extracted filename (including terminator) is no longer than [`OS\_MAX\_PATH\_LEN`](#)

**Parameters**

in	<i>OriginalPath</i>	The original path (must not be null)
out	<i>FileNameOnly</i>	The filename that is extracted from the path (must not be null)

**Returns**

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

**Return values**

<i>CFE_FS_BAD_ARGUMENT</i>	Bad Argument.
<i>CFE_FS_FNAME_TOO_LONG</i>	Filename Too Long.
<i>CFE_FS_INVALID_PATH</i>	Invalid Path.
<i>CFE_SUCCESS</i>	Successful execution.

```
10.27.2.4 CFE_FS_GetDefaultExtension() const char* CFE_FS_GetDefaultExtension (
    CFE_FS_FileCategory_t FileCategory )
```

Get the default filename extension for a file category.

Certain file types may have an extension that varies from system to system. This is primarily an issue for application modules which are ".so" on Linux systems, ".dll" on Windows, ".o" on VxWorks, ".obj" on RTEMS, and so on.

This uses a combination of compile-time configuration and hints from the build environment to get the default/expected extension for a given file category.

#### Returns

String containing the extension

#### Return values

NULL	if no default extension is known for the given file category
------	--

```
10.27.2.5 CFE_FS_GetDefaultMountPoint() const char* CFE_FS_GetDefaultMountPoint (
    CFE_FS_FileCategory_t FileCategory )
```

Get the default virtual mount point for a file category.

Certain classes of files generally reside in a common directory, mainly either the persistent storage (/cf typically) or ram disk (/ram typically).

Ephemeral status files are generally in the ram disk while application modules and scripts are generally in the persistent storage.

This returns the expected directory for a given class of files in the form of a virtual OSAL mount point string.

#### Returns

String containing the mount point

#### Return values

NULL	if no mount point is known for the given file category
------	--

```
10.27.2.6 CFE_FS_ParseInputFileName() int32 CFE_FS_ParseInputFileName (
    char * OutputBuffer,
    const char * InputName,
    size_t OutputBufSize,
    CFE_FS_FileCategory_t FileCategory )
```

Parse a filename string from the user into a local buffer.

#### Description

Simplified API for [CFE\\_FS\\_ParseInputFileNameEx\(\)](#) where input is always known to be a non-empty, null terminated string and the fixed-length input buffer not needed. For instance this may be used where the input is a fixed string from cfe\_platform\_cfg.h or similar.

#### Assumptions, External Events, and Notes:

The parameters are organized such that this is basically like strncpy() with an extra argument, and existing file name accesses which use a direct copy can easily change to use this instead.

**See also**

[CFE\\_FS\\_ParseInputFileName\(\)](#)

**Parameters**

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

**Returns**

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

**10.27.2.7 CFE\_FS\_ParseInputFileNameEx()** `int32 CFE_FS_ParseInputFileNameEx (`

```
    char * OutputBuffer,
    const char * InputBuffer,
    size_t OutputBufSize,
    size_t InputBufSize,
    const char * DefaultInput,
    const char * DefaultPath,
    const char * DefaultExtension )
```

Parse a filename input from an input buffer into a local buffer.

**Description**

This provides a more user friendly way to specify file names, using default values for the path and extension, which can vary from system to system.

If InputBuffer is null or its length is zero, then DefaultInput is used as if it was the content of the input buffer.

If either the pathname or extension is missing from the input, it will be added from defaults, with the complete fully-qualified filename stored in the output buffer.

**Assumptions, External Events, and Notes:**

1. The paths and filenames used here are the standard unix style filenames separated by "/" (path) and "." (extension) characters.
2. Input Buffer has a fixed max length. Parsing will not exceed InputBufSize, and does not need to be null terminated. However parsing will stop at the first null char, when the input is shorter than the maximum.

**Parameters**

<i>out</i>	<i>OutputBuffer</i>	Buffer to store result (must not be null).
<i>in</i>	<i>InputBuffer</i>	A input buffer that may contain a file name (e.g. from command) (must not be null).
<i>in</i>	<i>OutputBufSize</i>	Maximum Size of output buffer (must not be zero).
<i>in</i>	<i>InputBufSize</i>	Maximum Size of input buffer.
<i>in</i>	<i>DefaultInput</i>	Default value to use for input if InputBffer is empty
<i>in</i>	<i>DefaultPath</i>	Default value to use for pathname if omitted from input
<i>in</i>	<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.

## 10.28 cFE Generic Message APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_MSG\\_Init \(CFE\\_MSG\\_Message\\_t \\*MsgPtr, CFE\\_SB\\_MsgId\\_t MsgId, CFE\\_MSG\\_Size\\_t Size\)](#)

*Initialize a message.*

### 10.28.1 Detailed Description

### 10.28.2 Function Documentation

#### 10.28.2.1 CFE\_MSG\_Init() [CFE\\_Status\\_t CFE\\_MSG\\_Init \(](#)

```
CFE_MSG_Message_t * MsgPtr,
CFE_SB_MsgId_t MsgId,
CFE_MSG_Size_t Size )
```

Initialize a message.

**Description**

This routine initialize a message. The entire message is set to zero (based on size), defaults are set, then the size and bits from MsgId are set.

**Parameters**

<b>out</b>	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
<b>in</b>	<i>MsgId</i>	MsgId that corresponds to message
<b>in</b>	<i>Size</i>	Total size of the message (used to set length field)

**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.

Referenced by CF\_AppInit(), CF\_CFDP\_MsgOutGet(), and CF\_CFDP\_SendEotPkt().

## 10.29 cFE Message Primary Header APIs

### Functions

- `CFE_Status_t CFE_MSG_GetSize (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t *Size)`  
*Gets the total size of a message.*
- `CFE_Status_t CFE_MSG_SetSize (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Size_t Size)`  
*Sets the total size of a message.*
- `CFE_Status_t CFE_MSG.GetType (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t *Type)`  
*Gets the message type.*
- `CFE_Status_t CFE_MSG_SetType (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Type_t Type)`  
*Sets the message type.*
- `CFE_Status_t CFE_MSG_GetHeaderVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t *Version)`  
*Gets the message header version.*
- `CFE_Status_t CFE_MSG_SetHeaderVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_HeaderVersion_t Version)`  
*Sets the message header version.*
- `CFE_Status_t CFE_MSG_GetHasSecondaryHeader (const CFE_MSG_Message_t *MsgPtr, bool *HasSecondary)`  
*Gets the message secondary header boolean.*
- `CFE_Status_t CFE_MSG_SetHasSecondaryHeader (CFE_MSG_Message_t *MsgPtr, bool HasSecondary)`  
*Sets the message secondary header boolean.*
- `CFE_Status_t CFE_MSG_GetApld (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t *Apld)`  
*Gets the message application ID.*
- `CFE_Status_t CFE_MSG_SetApld (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t Apld)`  
*Sets the message application ID.*
- `CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)`  
*Gets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)`  
*Sets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)`  
*Gets the message sequence count.*
- `CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)`  
*Sets the message sequence count.*
- `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)`  
*Gets the next sequence count value (rolls over if appropriate)*

#### 10.29.1 Detailed Description

#### 10.29.2 Function Documentation

```
10.29.2.1 CFE_MSG_GetApId() CFE_Status_t CFE_MSG_GetApId (
    const CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_ApId_t * ApId )
```

Gets the message application ID.

#### Description

This routine gets the message application ID.

#### Parameters

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

#### Returns

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

#### Return values

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

```
10.29.2.2 CFE_MSG_GetHasSecondaryHeader() CFE_Status_t CFE_MSG_GetHasSecondaryHeader (
    const CFE_MSG_Message_t * MsgPtr,
    bool * HasSecondary )
```

Gets the message secondary header boolean.

#### Description

This routine gets the message secondary header boolean.

#### Parameters

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

#### Returns

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

#### Return values

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

**10.29.2.3 CFE\_MSG\_GetHeaderVersion()** `CFE_Status_t CFE_MSG_GetHeaderVersion (`  
    `const CFE_MSG_Message_t * MsgPtr,`  
    `CFE_MSG_HeaderVersion_t * Version )`

Gets the message header version.

#### Description

This routine gets the message header version.

#### Parameters

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

#### Returns

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

#### Return values

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

**10.29.2.4 CFE\_MSG\_GetNextSequenceCount()** `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (`  
    `CFE_MSG_SequenceCount_t SeqCnt )`

Gets the next sequence count value (rolls over if appropriate)

#### Description

Abstract method to get the next valid sequence count value. Will roll over to zero for any input value greater than or equal to the maximum possible sequence count value given the field in the header.

#### Parameters

in	<i>SeqCnt</i>	Sequence count
----	---------------	----------------

#### Returns

The next valid sequence count value

**10.29.2.5 CFE\_MSG\_GetSegmentationFlag()** `CFE_Status_t CFE_MSG_GetSegmentationFlag (`  
    `const CFE_MSG_Message_t * MsgPtr,`  
    `CFE_MSG_SegmentationFlag_t * SegFlag )`

Gets the message segmentation flag.

#### Description

This routine gets the message segmentation flag

**Parameters**

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

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_MSG_BAD_ARGUMENT</i></a>	Error - bad argument.

**10.29.2.6 CFE\_MSG\_GetSequenceCount()** [\*CFE\\_Status\\_t\*](#) *CFE\_MSG\_GetSequenceCount* (

```
const CFE\_MSG\_Message\_t * MsgPtr,
CFE\_MSG\_SequenceCount\_t * SeqCnt )
```

Gets the message sequence count.

**Description**

This routine gets the message sequence count.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_MSG_BAD_ARGUMENT</i></a>	Error - bad argument.

**10.29.2.7 CFE\_MSG.GetSize()** [\*CFE\\_Status\\_t\*](#) *CFE\_MSG\_GetSize* (

```
const CFE\_MSG\_Message\_t * MsgPtr,
CFE\_MSG\_Size\_t * Size )
```

Gets the total size of a message.

**Description**

This routine gets the total size of the message.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

Referenced by CF\_AppPipe(), CF\_CFDP\_ReceiveMessage(), and CF\_ProcessGroundCommand().

**10.29.2.8 CFE\_MSG\_GetType()** [CFE\\_Status\\_t](#) CFE\_MSG\_GetType (

```
    const CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_Type_t * Type )
```

Gets the message type.

**Description**

This routine gets the message type.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

Referenced by CF\_CFDP\_ReceiveMessage().

**10.29.2.9 CFE\_MSG\_SetApId()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetApId (

```
    CFE_MSG_Message_t * MsgPtr,
    CFE_MSG_ApId_t ApId )
```

Sets the message application ID.

**Description**

This routine sets the message application ID. Typically set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

**Parameters**

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

**Returns**

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

**Return values**

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

**10.29.2.10 CFE\_MSG\_SetHasSecondaryHeader()** `CFE_Status_t CFE_MSG_SetHasSecondaryHeader (`  
 `CFE_MSG_Message_t * MsgPtr,`  
 `bool HasSecondary )`

Sets the message secondary header boolean.

**Description**

This routine sets the message secondary header boolean. Typically only set within message initialization and not used by APPs.

**Parameters**

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

**Returns**

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

**Return values**

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

**10.29.2.11 CFE\_MSG\_SetHeaderVersion()** `CFE_Status_t CFE_MSG_SetHeaderVersion (`  
 `CFE_MSG_Message_t * MsgPtr,`  
 `CFE_MSG_HeaderVersion_t Version )`

Sets the message header version.

#### Description

This routine sets the message header version. Typically only set within message initialization and not used by APPs.

#### Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message.
in	<i>Version</i>	Header version

#### Returns

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

#### Return values

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

**10.29.2.12 CFE\_MSG\_SetSegmentationFlag()** *CFE\_Status\_t* CFE\_MSG\_SetSegmentationFlag (   
     *CFE\_MSG\_Message\_t* \* *MsgPtr*,  
     *CFE\_MSG\_SegmentationFlag\_t* *SegFlag* )

Sets the message segmentation flag.

#### Description

This routine sets the message segmentation flag.

#### Parameters

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

#### Returns

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

#### Return values

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

**10.29.2.13 CFE\_MSG\_SetSequenceCount()** *CFE\_Status\_t* CFE\_MSG\_SetSequenceCount (   
     *CFE\_MSG\_Message\_t* \* *MsgPtr*,

`CFE_MSG_SequenceCount_t SeqCnt )`

Sets the message sequence count.

#### Description

This routine sets the message sequence count.

#### Parameters

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

#### Returns

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

#### Return values

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

**10.29.2.14 CFE\_MSG\_SetSize()** `CFE_Status_t CFE_MSG_SetSize (`  
`CFE_MSG_Message_t * MsgPtr,`  
`CFE_MSG_Size_t Size )`

Sets the total size of a message.

#### Description

This routine sets the total size of the message.

#### Parameters

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

#### Returns

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

#### Return values

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

Referenced by CF\_CFDP\_Send().

**10.29.2.15 CFE\_MSG\_SetType()** `CFE_Status_t CFE_MSG_SetType (`

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_Type_t Type )
```

Sets the message type.

#### Description

This routine sets the message type.

#### Parameters

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

#### Returns

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

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

## 10.30 cFE Message Extended Header APIs

#### Functions

- `CFE_Status_t CFE_MSG_GetEDSVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t *Version)`  
*Gets the message EDS version.*
- `CFE_Status_t CFE_MSG_SetEDSVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t Version)`  
*Sets the message EDS version.*
- `CFE_Status_t CFE_MSG_GetEndian (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t *Endian)`  
*Gets the message endian.*
- `CFE_Status_t CFE_MSG_SetEndian (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t Endian)`  
*Sets the message endian.*
- `CFE_Status_t CFE_MSG_GetPlaybackFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t *PlayFlag)`  
*Gets the message playback flag.*
- `CFE_Status_t CFE_MSG_SetPlaybackFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t PlayFlag)`  
*Sets the message playback flag.*
- `CFE_Status_t CFE_MSG_GetSubsystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t *Subsystem)`  
*Gets the message subsystem.*
- `CFE_Status_t CFE_MSG_SetSubsystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t Subsystem)`  
*Sets the message subsystem.*
- `CFE_Status_t CFE_MSG_GetSystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t *System)`  
*Gets the message system.*

- `CFE_Status_t CFE_MSG_SetSystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t System)`  
*Sets the message system.*

### 10.30.1 Detailed Description

### 10.30.2 Function Documentation

**10.30.2.1 CFE\_MSG\_GetEDSVersion()** `CFE_Status_t CFE_MSG_GetEDSVersion (`  
    `const CFE_MSG_Message_t * MsgPtr,`  
    `CFE_MSG_EDSVersion_t * Version )`

Gets the message EDS version.

#### Description

This routine gets the message EDS version.

#### Parameters

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

#### Returns

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

#### Return values

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

**10.30.2.2 CFE\_MSG\_GetEndian()** `CFE_Status_t CFE_MSG_GetEndian (`  
    `const CFE_MSG_Message_t * MsgPtr,`  
    `CFE_MSG_Endian_t * Endian )`

Gets the message endian.

#### Description

This routine gets the message endian.

#### Parameters

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

**10.30.2.3 CFE\_MSG\_GetPlaybackFlag()** [CFE\\_Status\\_t](#) CFE\_MSG\_GetPlaybackFlag (   
   const [CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr*,  
   [CFE\\_MSG\\_PlaybackFlag\\_t](#) \* *PlayFlag* )

Gets the message playback flag.

**Description**

This routine gets the message playback flag.

**Parameters**

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

**10.30.2.4 CFE\_MSG\_GetSubsystem()** [CFE\\_Status\\_t](#) CFE\_MSG\_GetSubsystem (   
   const [CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr*,  
   [CFE\\_MSG\\_Subsystem\\_t](#) \* *Subsystem* )

Gets the message subsystem.

**Description**

This routine gets the message subsystem

**Parameters**

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

**10.30.2.5 CFE\_MSG\_GetSystem()** [CFE\\_Status\\_t](#) CFE\_MSG\_GetSystem (

```
    const CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_System_t * System )
```

Gets the message system.

**Description**

This routine gets the message system id

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

**10.30.2.6 CFE\_MSG\_SetEDSVersion()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetEDSVersion (

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_EDSVersion_t Version )
```

Sets the message EDS version.

**Description**

This routine sets the message EDS version.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

**10.30.2.7 CFE\_MSG\_SetEndian()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetEndian (

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_Endian_t Endian )
```

Sets the message endian.

**Description**

This routine sets the message endian. Invalid endian selection will set big endian.

**Parameters**

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

**10.30.2.8 CFE\_MSG\_SetPlaybackFlag()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetPlaybackFlag (

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_PlaybackFlag_t PlayFlag )
```

Sets the message playback flag.

**Description**

This routine sets the message playback flag.

**Parameters**

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

**10.30.2.9 CFE\_MSG\_SetSubsystem()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetSubsystem (

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_Subsystem_t Subsystem )
```

Sets the message subsystem.

**Description**

This routine sets the message subsystem. Some bits may be set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

**Parameters**

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

**10.30.2.10 CFE\_MSG\_SetSystem()** [CFE\\_Status\\_t](#) CFE\_MSG\_SetSystem (

```
    CFE_MSG_Message_t * MsgPtr,  
    CFE_MSG_System_t System )
```

Sets the message system.

**Description**

This routine sets the message system id. Some bits may be set at initialization using the MsgId, but API available to set bits that may not be included in MsgId.

**Parameters**

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

**Returns**

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

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

## 10.31 cFE Message Secondary Header APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_MSG\\_GenerateChecksum \(CFE\\_MSG\\_Message\\_t \\*MsgPtr\)](#)  
*Calculates and sets the checksum of a message.*
- [CFE\\_Status\\_t CFE\\_MSG\\_ValidateChecksum \(const CFE\\_MSG\\_Message\\_t \\*MsgPtr, bool \\*isValid\)](#)  
*Validates the checksum of a message.*
- [CFE\\_Status\\_t CFE\\_MSG\\_SetFcnCode \(CFE\\_MSG\\_Message\\_t \\*MsgPtr, CFE\\_MSG\\_FcnCode\\_t FcnCode\)](#)  
*Sets the function code field in a message.*
- [CFE\\_Status\\_t CFE\\_MSG\\_GetFcnCode \(const CFE\\_MSG\\_Message\\_t \\*MsgPtr, CFE\\_MSG\\_FcnCode\\_t \\*FcnCode\)](#)  
*Gets the function code field from a message.*
- [CFE\\_Status\\_t CFE\\_MSG\\_GetMsgTime \(const CFE\\_MSG\\_Message\\_t \\*MsgPtr, CFE\\_TIME\\_SysTime\\_t \\*Time\)](#)  
*Gets the time field from a message.*
- [CFE\\_Status\\_t CFE\\_MSG\\_SetMsgTime \(CFE\\_MSG\\_Message\\_t \\*MsgPtr, CFE\\_TIME\\_SysTime\\_t NewTime\)](#)  
*Sets the time field in a message.*

### 10.31.1 Detailed Description

### 10.31.2 Function Documentation

#### 10.31.2.1 CFE\_MSG\_GenerateChecksum() [CFE\\_Status\\_t CFE\\_MSG\\_GenerateChecksum \(CFE\\_MSG\\_Message\\_t \\* MsgPtr \)](#)

Calculates and sets the checksum of a message.

**Description**

This routine calculates the checksum of a message according to an implementation-defined algorithm. Then, it sets the checksum field in the message with the calculated value. The contents and location of this field will depend on the underlying implementation of messages. It may be a checksum, a CRC, or some other algorithm.

**Assumptions, External Events, and Notes:**

- If the underlying implementation of messages does not include a checksum field, then this routine will return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#)

**Parameters**

<a href="#">in, out</a>	<a href="#">MsgPtr</a>	A pointer to the buffer that contains 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.

**10.31.2.2 CFE\_MSG\_GetFcnCode()** [CFE\\_Status\\_t](#) CFE\_MSG\_GetFcnCode (   
   const [CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr*,  
   [CFE\\_MSG\\_FcnCode\\_t](#) \* *FcnCode* )

Gets the function code field from a message.

**Description**

This routine gets the function code from a message.

**Assumptions, External Events, and Notes:**

- If the underlying implementation of messages does not include a function code field, then this routine will set *FcnCode* to zero and return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#)

**Parameters**

<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>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 CF\_AppPipe(), and CF\_ProcessGroundCommand().

**10.31.2.3 CFE\_MSG\_GetMsgTime()** [CFE\\_Status\\_t](#) CFE\_MSG\_GetMsgTime (   
   const [CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr*,  
   [CFE\\_TIME\\_SysTime\\_t](#) \* *Time* )

Gets the time field from a message.

**Description**

This routine gets the time from a message.

**Assumptions, External Events, and Notes:**

- If the underlying implementation of messages does not include a time field, then this routine will set Time to zero and return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#)
- Note default implementation of command messages do not have a time field.

**Parameters**

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

**Returns**

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

**Return values**

<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.

**10.31.2.4 CFE\_MSG\_SetFcnCode()** [`CFE\_Status\_t CFE\_MSG\_SetFcnCode \(`](#)  
[`CFE\_MSG\_Message\_t \* MsgPtr,`](#)  
[`CFE\_MSG\_FcnCode\_t FcnCode \)`](#)

Sets the function code field in a message.

**Description**

This routine sets the function code of a message.

**Assumptions, External Events, and Notes:**

- If the underlying implementation of messages does not include a function code field, then this routine will do nothing to the message contents and will return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#).

**Parameters**

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

**Returns**

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

**Return values**

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

**10.31.2.5 CFE\_MSG\_SetMsgTime()** *CFE\_Status\_t* CFE\_MSG\_SetMsgTime (

```
    CFE_MSG_Message_t * MsgPtr,
    CFE_TIME_SysTime_t NewTime )
```

Sets the time field in a message.

**Description**

This routine sets the time of a message. Most applications will want to use [CFE\\_SB\\_TimeStampMsg](#) instead of this function. But, when needed, this API can be used to set multiple messages with identical time stamps.

**Assumptions, External Events, and Notes:**

- If the underlying implementation of messages does not include a time field, then this routine will do nothing to the message contents and will return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#).
- Note default implementation of command messages do not have a time field.

**Parameters**

<i>in, out</i>	<i>MsgPtr</i>	A pointer to the message (must not be null).
<i>in</i>	<i>NewTime</i>	The time to include in the message. This will usually be a time from <a href="#">CFE_TIME_GetTime</a> .

**Returns**

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

**Return values**

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

Referenced by CF\_CFDP\_Send().

**10.31.2.6 CFE\_MSG\_ValidateChecksum()** *CFE\_Status\_t* CFE\_MSG\_ValidateChecksum (

```
    const CFE_MSG_Message_t * MsgPtr,
    bool * IsValid )
```

Validates the checksum of a message.

**Description**

This routine validates the checksum of a message according to an implementation-defined algorithm.

**Assumptions, External Events, and Notes:**

- If the underlying implementation of messages does not include a checksum field, then this routine will return [CFE\\_MSG\\_WRONG\\_MSG\\_TYPE](#) and set the *IsValid* parameter false.

**Parameters**

in	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null). This must point to the first byte of the message header.
out	<i>IsValid</i>	<p>Checksum validation result (must not be null)</p> <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**

<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.

## 10.32 cFE Message Id APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_MSG\\_GetMsgId](#) ([const CFE\\_MSG\\_Message\\_t \\*MsgPtr](#), [CFE\\_SB\\_MsgId\\_t \\*MsgId](#))  
*Gets the message id from a message.*
- [CFE\\_Status\\_t CFE\\_MSG\\_SetMsgId](#) ([CFE\\_MSG\\_Message\\_t \\*MsgPtr](#), [CFE\\_SB\\_MsgId\\_t MsgId](#))  
*Sets the message id bits in a message.*
- [CFE\\_Status\\_t CFE\\_MSG.GetTypeFromMsgId](#) ([CFE\\_SB\\_MsgId\\_t MsgId](#), [CFE\\_MSG\\_Type\\_t \\*Type](#))  
*Gets message type using message ID.*

### 10.32.1 Detailed Description

### 10.32.2 Function Documentation

**10.32.2.1 CFE\_MSG\_GetMsgId()** [CFE\\_Status\\_t](#) [CFE\\_MSG\\_GetMsgId](#) (

```
const CFE_MSG_Message_t * MsgPtr,
CFE_SB_MsgId_t * MsgId )
```

Gets the message id from a message.

**Description**

This routine gets the message id from a message. The message id is a hash of bits in the message header, used by the software bus for routing. Message id needs to be unique for each endpoint in the system.

**Parameters**

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

**Returns**

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

**Return values**

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

Referenced by CF\_AppPipe().

**10.32.2.2 CFE\_MSG\_GetTypeFromMsgId()** *CFE\_Status\_t* CFE\_MSG\_GetTypeFromMsgId (   
*CFE\_SB\_MsgId\_t* *MsgId*,  
*CFE\_MSG\_Type\_t* \* *Type* )

Gets message type using message ID.

**Description**

This routine gets the message type using the message ID

**Parameters**

in	<i>MsgId</i>	Message id
out	<i>Type</i>	Message type (must not be null)

**Returns**

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

**Return values**

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

**10.32.2.3 CFE\_MSG\_SetMsgId()** *CFE\_Status\_t* CFE\_MSG\_SetMsgId (   
*CFE\_MSG\_Message\_t* \* *MsgPtr*,  
*CFE\_SB\_MsgId\_t* *MsgId* )

Sets the message id bits in a message.

### Description

This routine sets the message id bits in a message. The message id is a hash of bits in the message header, used by the software bus for routing. Message id needs to be unique for each endpoint in the system.

### Note

This API only sets the bits in the header that make up the message ID. No other values in the header are modified.

The user should ensure that this function is only called with a valid MsgId parameter value. If called with an invalid value, the results are implementation-defined. The implementation may or may not return the error code [CFE\\_MSG\\_BAD\\_ARGUMENT](#) in this case.

### Parameters

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

### Returns

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

### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

## 10.33 cFE Message Integrity APIs

### Functions

- [CFE\\_Status\\_t CFE\\_MSG\\_OriginationAction](#) (*CFE\_MSG\_Message\_t* \**MsgPtr*, *size\_t* *BufferSize*, *bool* \**IsAcceptable*)  
*Perform any necessary actions on a newly-created message, prior to sending.*
- [CFE\\_Status\\_t CFE\\_MSG\\_VerificationAction](#) (*const CFE\_MSG\_Message\_t* \**MsgPtr*, *size\_t* *BufferSize*, *bool* \**IsAcceptable*)  
*Checks message integrity/acceptability.*

#### 10.33.1 Detailed Description

#### 10.33.2 Function Documentation

**10.33.2.1 CFE\_MSG\_OriginationAction()** [CFE\\_Status\\_t](#) CFE\_MSG\_OriginationAction (   
   [CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr*,  
   *size\_t* *BufferSize*,  
   *bool* \* *IsAcceptable* )

Perform any necessary actions on a newly-created message, prior to sending.

### Description

This routine updates and/or appends any necessary fields on a message, is invoked via SB just prior to broadcasting the message. The actions include updating any values that should be computed/updated per message, including:

- setting the sequence number
- updating the timestamp, if present
- computing any error control or checksum fields, if present

The MSG module implementation determines which header fields meet this criteria and how they should be computed. The BufferSize parameter indicates the allocation size message of the buffer that holds the message (i.e. the message envelope size). In some implementations, the allocated buffer may include extra space in order to append a CRC or digital signature.

### See also

[CFE\\_MSG\\_VerificationAction](#)

### Parameters

in, out	<i>MsgPtr</i>	A pointer to the buffer that contains the message (must not be null).
in	<i>BufferSize</i>	The size of the buffer encapsulating the message
out	<i>IsAcceptable</i>	Output variable to be set, indicates message acceptability (must not be null)

### Returns

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

### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

### 10.33.2.2 CFE\_MSG\_VerificationAction()

```
CFE_Status_t CFE_MSG_VerificationAction (
    const CFE_MSG_Message_t * MsgPtr,
    size_t BufferSize,
    bool * IsAcceptable )
```

Checks message integrity/acceptability.

### Description

This routine validates that any error-control field(s) in the message header matches the expected value.

The specific function of this API is entirely dependent on the header fields and may be a no-op if no error checking is implemented. In that case, it will always output "true".

### Note

Due to the fact that software bus uses a multicast architecture, this function must not modify the message, as the buffer may be shared among multiple receivers. This should generally be the inverse of [CFE\\_MSG\\_OriginationAction\(\)](#), but on the origination side it may update header fields and/or modify the message, on the verification/receive side it must only check those fields, not modify them.

**See also**[CFE\\_MSG\\_OriginationAction](#)**Parameters**

in	<i>MsgPtr</i>	Message Pointer (must not be null)
in	<i>BufferSize</i>	The size of the buffer encapsulating the message
out	<i>IsAcceptable</i>	Output variable to be set, indicates message acceptability (must not be null)

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

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_MSG_BAD_ARGUMENT</a>	Error - bad argument.

## 10.34 cFE Pipe Management APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_SB\\_CreatePipe \(CFE\\_SB\\_Pipeld\\_t \\*PipeldPtr, uint16 Depth, const char \\*PipeName\)](#)  
*Creates a new software bus pipe.*
- [CFE\\_Status\\_t CFE\\_SB\\_DeletePipe \(CFE\\_SB\\_Pipeld\\_t Pipeld\)](#)  
*Delete a software bus pipe.*
- [CFE\\_Status\\_t CFE\\_SB\\_Pipeld\\_ToIndex \(CFE\\_SB\\_Pipeld\\_t PipelD, uint32 \\*Idx\)](#)  
*Obtain an index value correlating to an SB Pipe ID.*
- [CFE\\_Status\\_t CFE\\_SB\\_SetPipeOpts \(CFE\\_SB\\_Pipeld\\_t Pipeld, uint8 OptS\)](#)  
*Set options on a pipe.*
- [CFE\\_Status\\_t CFE\\_SB\\_GetPipeOpts \(CFE\\_SB\\_Pipeld\\_t Pipeld, uint8 \\*OptSPtr\)](#)  
*Get options on a pipe.*
- [CFE\\_Status\\_t CFE\\_SB\\_GetPipeName \(char \\*PipeNameBuf, size\\_t PipeNameSize, CFE\\_SB\\_Pipeld\\_t Pipeld\)](#)  
*Get the pipe name for a given id.*
- [CFE\\_Status\\_t CFE\\_SB\\_GetPipeldByName \(CFE\\_SB\\_Pipeld\\_t \\*PipeldPtr, const char \\*PipeName\)](#)  
*Get pipe id by pipe name.*

### 10.34.1 Detailed Description

### 10.34.2 Function Documentation

**10.34.2.1 CFE\_SB\_CreatePipe()** [CFE\\_Status\\_t](#) CFE\_SB\_CreatePipe (

```
    CFE_SB_Pipeld_t * PipeIdPtr,
    uint16 Depth,
    const char * PipeName )
```

Creates a new software bus pipe.

**Description**

This routine creates and initializes an input pipe that the calling application can use to receive software bus messages. By default, no messages are routed to the new pipe. So, the application must use [CFE\\_SB\\_Subscribe](#) to specify which messages it wants to receive on this pipe.

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>out</i>	<i>PipeIdPtr</i>	A pointer to a variable of type <a href="#">CFE_SB_PipeId_t</a> (must not be null), which will be filled in with the pipe ID information by the <a href="#">CFE_SB_CreatePipe</a> routine. *PipeIdPtr is the identifier for the created pipe.
<i>in</i>	<i>Depth</i>	The maximum number of messages that will be allowed on this pipe at one time.
<i>in</i>	<i>PipeName</i>	A string (must not be null) to be used to identify this pipe in error messages and routing information telemetry. The string must be no longer than <a href="#">OS_MAX_API_NAME</a> (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 [CF\\_AppInit\(\)](#), and [CF\\_CFDP\\_InitEngine\(\)](#).

### 10.34.2.2 CFE\_SB\_DeletePipe()

```
CFE_Status_t CFE_SB_DeletePipe (
    CFE_SB_PipeId_t PipeId )
```

Delete a software bus pipe.

**Description**

This routine deletes an input pipe and cleans up all data structures associated with the pipe. All subscriptions made for this pipe by calls to [CFE\\_SB\\_Subscribe](#) will be automatically removed from the SB routing tables. Any messages in the pipe will be discarded.

Applications should not call this routine for all of their SB pipes as part of their orderly shutdown process, as the pipe will be deleted by the support framework at the appropriate time.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>PipeId</i>	The pipe ID (obtained previously from <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](#)

Referenced by [CF\\_CFDP\\_DisableEngine\(\)](#).

**10.34.2.3 CFE\_SB\_GetPipeIdByName()** [CFE\\_Status\\_t](#) CFE\_SB\_GetPipeIdByName (   
*CFE\_SB\_PipeId\_t* \* *PipeIdPtr*,  
*const char* \* *PipeName* )

Get pipe id by pipe name.

**Description**

This routine finds the pipe id for a pipe name.

**Parameters**

in	<i>PipeName</i>	The name of the pipe (must not be null).
out	<i>PipeldPtr</i>	The Pipeld for that name (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\\_PIPEOPTS\\_IGNOREMINE](#)

**10.34.2.4 CFE\_SB\_GetPipeName()** [CFE\\_Status\\_t](#) CFE\_SB\_GetPipeName (   
*char* \* *PipeNameBuf*,

```
size_t PipeNameSize,
CFE_SB_PipeId_t PipeId )
```

Get the pipe name for a given id.

#### Description

This routine finds the pipe name for a pipe id.

#### Parameters

out	<i>PipeNameBuf</i>	The buffer to receive the pipe name (must not be null).
in	<i>PipeNameSize</i>	The size (in chars) of the PipeName buffer (must not be zero).
in	<i>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\\_GetPipeldByName](#)

```
10.34.2.5 CFE_SB_GetPipeOpts() CFE_Status_t CFE_SB_GetPipeOpts (
    CFE_SB_PipeId_t PipeId,
    uint8 * OptsPtr )
```

Get options on a pipe.

#### Description

This routine gets the current options on a pipe.

#### Parameters

in	<i>Pipeld</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](#)

**10.34.2.6 CFE\_SB\_PipeId\_ToIndex()** `CFE_Status_t CFE_SB_PipeId_ToIndex (`

```
    CFE_SB_PipeId_t PipeID,  
    uint32 * Idx )
```

Obtain an index value correlating to an SB Pipe ID.

This calculates a zero based integer value that may be used for indexing into a local resource table/array.

Index values are only guaranteed to be unique for resources of the same type. For instance, the indices corresponding to two [valid] application IDs will never overlap, but the index of a pipe ID and an app ID may be the same. Furthermore, indices may be reused if a resource is deleted and re-created.

**Note**

There is no inverse of this function - indices cannot be converted back to the original PipeID value. The caller should retain the original ID for future use.

**Parameters**

in	<i>PipeID</i>	Pipe ID to convert
out	<i>Idx</i>	Buffer where the calculated index will be stored (must not be null)

**Returns**

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

**Return values**

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

**10.34.2.7 CFE\_SB\_SetPipeOpts()** `CFE_Status_t CFE_SB_SetPipeOpts (`

```
    CFE_SB_PipeId_t PipeId,  
    uint8 Opts )
```

Set options on a pipe.

**Description**

This routine sets (or clears) options to alter the pipe's behavior. Options are (re)set every call to this routine.

**Parameters**

in	<i>PipeId</i>	The pipe ID of the pipe to set options on.
in	<i>Opts</i>	A bit field of options: <a href="#">cFE SB Pipe options</a>

**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\\_GetPipeOpts](#) [CFE\\_SB\\_GetPipeIdByName](#) [CFE\\_SB\\_PIPEOPTS\\_IGNOREMIN](#)

## 10.35 cFE Message Subscription Control APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_SB\\_SubscribeEx](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId, [CFE\\_SB\\_PipeId\\_t](#) PipeId, [CFE\\_SB\\_Qos\\_t](#) Quality, [uint16](#) MsgLim)
   
*Subscribe to a message on the software bus.*
- [CFE\\_Status\\_t CFE\\_SB\\_Subscribe](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId, [CFE\\_SB\\_PipeId\\_t](#) PipeId)
   
*Subscribe to a message on the software bus with default parameters.*
- [CFE\\_Status\\_t CFE\\_SB\\_SubscribeLocal](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId, [CFE\\_SB\\_PipeId\\_t](#) PipeId, [uint16](#) MsgLim)
   
*Subscribe to a message while keeping the request local to a cpu.*
- [CFE\\_Status\\_t CFE\\_SB\\_Unsubscribe](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId, [CFE\\_SB\\_PipeId\\_t](#) PipeId)
   
*Remove a subscription to a message on the software bus.*
- [CFE\\_Status\\_t CFE\\_SB\\_UnsubscribeLocal](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId, [CFE\\_SB\\_PipeId\\_t](#) PipeId)
   
*Remove a subscription to a message on the software bus on the current CPU.*

### 10.35.1 Detailed Description

### 10.35.2 Function Documentation

#### 10.35.2.1 [CFE\\_SB\\_Subscribe\(\)](#) [CFE\\_Status\\_t CFE\\_SB\\_Subscribe](#) (

```
CFE_SB_MsgId_t MsgId,
CFE_SB_PipeId_t PipeId )
```

Subscribe to a message on the software bus with default parameters.

**Description**

This routine adds the specified pipe to the destination list for the specified message ID. This is the same as [CFE\\_SB\\_SubscribeEx](#) with the Quality field set to [CFE\\_SB\\_DEFAULT\\_QOS](#) and MsgLim set to [CFE\\_PLATFORM\\_SB\\_DEFAULT\\_MSG\\_LIMIT](#) (4).

**Assumptions, External Events, and Notes:**

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

**Parameters**

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

**Returns**

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

**Return values**

<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\\_SubscribeEx](#), [CFE\\_SB\\_SubscribeLocal](#), [CFE\\_SB\\_Unsubscribe](#), [CFE\\_SB\\_UnsubscribeLocal](#)

Referenced by [CF\\_ApplInit\(\)](#).

```
10.35.2.2 CFE_SB_SubscribeEx() CFE\_Status\_t CFE_SB_SubscribeEx (
    CFE\_SB\_MsgId\_t MsgId,
    CFE\_SB\_PipeId\_t PipeId,
    CFE\_SB\_Qos\_t Quality,
    uint16 MsgLim )
```

Subscribe to a message on the software bus.

**Description**

This routine adds the specified pipe to the destination list associated with the specified message ID.

**Assumptions, External Events, and Notes:**

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

**Parameters**

in	<i>MsgId</i>	The message ID of the message to be subscribed to.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should be sent to.
in	<i>Quality</i>	The requested Quality of Service (QoS) required of the messages. Most callers will use <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**

<a href="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_SB_MAX_MSGS_MET</i></a>	(return value only verified in coverage test) Max Messages Met.
<a href="#"><i>CFE_SB_MAX_DESTS_MET</i></a>	Max Destinations Met.
<a href="#"><i>CFE_SB_BAD_ARGUMENT</i></a>	Bad Argument.
<a href="#"><i>CFE_SB_BUF_ALOC_ERR</i></a>	(return value only verified in coverage test) Buffer Allocation Error.

**See also**

[CFE\\_SB\\_Subscribe](#), [CFE\\_SB\\_SubscribeLocal](#), [CFE\\_SB\\_Unsubscribe](#), [CFE\\_SB\\_UnsubscribeLocal](#)

**10.35.2.3 CFE\_SB\_SubscribeLocal()** [\*CFE\\_Status\\_t\*](#) [\*CFE\\_SB\\_SubscribeLocal\*](#) (

```
CFE\_SB\_MsgId\_t MsgId,
CFE\_SB\_PipeId\_t PipeId,
uint16 MsgLim )
```

Subscribe to a message while keeping the request local to a cpu.

**Description**

This routine adds the specified pipe to the destination list for the specified message ID. This is similar to [CFE\\_SB\\_SubscribeEx](#) with the Quality field set to [\*CFE\\_SB\\_DEFAULT\\_QOS\*](#) and *MsgLim* set to [\*CFE\\_PLATFORM\\_SB\\_DEFAULT\\_MSG\\_LIMIT\*](#), but will not report the subscription.

Software Bus Network (SBN) application is an example use case, where local subscriptions should not be reported to peers.

**Assumptions, External Events, and Notes:**

- This API is typically only used by Software Bus Network (SBN) Application

**Parameters**

<i>in</i>	<i>MsgId</i>	The message ID of the message to be subscribed to.
<i>in</i>	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should be sent to.
<i>in</i>	<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="#"><i>CFE_SUCCESS</i></a>	Successful execution.
<a href="#"><i>CFE_SB_MAX_MSGS_MET</i></a>	(return value only verified in coverage test) Max Messages Met.
<a href="#"><i>CFE_SB_MAX_DESTS_MET</i></a>	Max Destinations Met.

#### Return values

<a href="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.
<a href="#">CFE_SB_BUF_ALOC_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](#)

Referenced by [CF\\_CFDP\\_InitEngine\(\)](#).

**10.35.2.4 CFE\_SB\_Unsubscribe()** [CFE\\_Status\\_t](#) CFE\_SB\_Unsubscribe (   
     [CFE\\_SB\\_MsgId\\_t](#) MsgId,  
     [CFE\\_SB\\_PipeId\\_t](#) PipeId )

Remove a subscription to a message on the software bus.

#### Description

This routine removes the specified pipe from the destination list for the specified message ID.

#### Assumptions, External Events, and Notes:

If the Pipe is not subscribed to MsgId, the [CFE\\_SB\\_UNSUB\\_NO\\_SUBS\\_EID](#) event will be generated and [CFE\\_SUCCESS](#) will be returned

#### Parameters

in	<i>MsgId</i>	The message ID of the message to be unsubscribed.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should no longer be sent to.

#### Returns

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

#### Return values

<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](#)

**10.35.2.5 CFE\_SB\_UnsubscribeLocal()** [CFE\\_Status\\_t](#) CFE\_SB\_UnsubscribeLocal (   
     [CFE\\_SB\\_MsgId\\_t](#) MsgId,  
     [CFE\\_SB\\_PipeId\\_t](#) PipeId )

Remove a subscription to a message on the software bus on the current CPU.

### Description

This routine removes the specified pipe from the destination list for the specified message ID on the current CPU.

### Assumptions, External Events, and Notes:

This API is typically only used by Software Bus Network (SBN) Application. If the Pipe is not subscribed to MsgId, the CFE\_SB\_UNSUB\_NO\_SUBS\_EID event will be generated and [CFE\\_SUCCESS](#) will be returned

### Parameters

in	<i>MsgId</i>	The message ID of the message to be unsubscribed.
in	<i>PipeId</i>	The pipe ID of the pipe the subscribed message should no longer be sent to.

### Returns

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

### Return values

<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\\_Unsubscribe](#)

## 10.36 cFE Send/Receive Message APIs

### Functions

- [CFE\\_Status\\_t CFE\\_SB\\_TransmitMsg](#) (`const CFE_MSG_Message_t *MsgPtr, bool IsOrigination`)  
*Transmit a message.*
- [CFE\\_Status\\_t CFE\\_SB\\_ReceiveBuffer](#) (`CFE_SB_Buffer_t **BufPtr, CFE_SB_PipeId_t PipeId, int32 TimeOut`)  
*Receive a message from a software bus pipe.*

### 10.36.1 Detailed Description

### 10.36.2 Function Documentation

**10.36.2.1 CFE\_SB\_ReceiveBuffer()** `CFE_Status_t CFE_SB_ReceiveBuffer (`  
`CFE_SB_Buffer_t ** BufPtr,`  
`CFE_SB_PipeId_t PipeId,`  
`int32 TimeOut )`

Receive a message from a software bus pipe.

**Description**

This routine retrieves the next message from the specified pipe. If the pipe is empty, this routine will block until either a new message comes in or the timeout value is reached.

**Assumptions, External Events, and Notes:**

Note - If an error occurs in this API, the \*BufPtr value may be NULL or random. Therefore, it is recommended that the return code be tested for CFE\_SUCCESS before processing the message.

**Parameters**

in, out	<i>BufPtr</i>	A pointer to the software bus buffer to receive to (must not be null). Typically a caller declares a ptr of type CFE_SB_Buffer_t (i.e. CFE_SB_Buffer_t *Ptr) then gives the address of that pointer (&Ptr) as this parameter. After a successful receipt of a message, *BufPtr will point to the first byte of the software bus buffer. This should be used as a read-only pointer (in systems with an MMU, writes to this pointer may cause a memory protection fault). The *BufPtr is valid only until the next call to CFE_SB_ReceiveBuffer for the same pipe.
in	<i>PipeId</i>	The pipe ID of the pipe containing the message to be obtained.
in	<i>TimeOut</i>	The number of milliseconds to wait for a new message if the pipe is empty at the time of the call. This can also be set to <a href="#">CFE_SB_POLL</a> for a non-blocking receive or <a href="#">CFE_SB_PEND_FOREVER</a> to wait forever for a message to arrive.

**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_TIME_OUT</a>	Time Out.
<a href="#">CFE_SB_PIPE_RD_ERR</a>	(return value only verified in coverage test) Pipe Read Error.
<a href="#">CFE_SB_NO_MESSAGE</a>	No Message.

Referenced by CF\_AppMain(), and CF\_CFDP\_ReceiveMessage().

```
10.36.2.2 CFE_SB_TransmitMsg() CFE_Status_t CFE_SB_TransmitMsg (
    const CFE_MSG_Message_t * MsgPtr,
    bool IsOrigination )
```

Transmit a message.

**Description**

This routine copies the specified message into a software bus buffer which is then transmitted to all subscribers. The software bus will read the message ID from the message header to determine which pipes should receive the message.

The IsOrigination parameter should be passed as "true" if the message was newly constructed by the sender and is being sent for the first time. This enables the message origination actions as determined by the CFE MSG module, which may include (but not limited to):

- Updating sequence number
- Updating timestamp
- Calculating a CRC, checksum, or other message error control field

Conversely, when forwarding a message that originated from an external entity (e.g. messages passing through CI or SBN), the parameter should be passed as "false" to not overwrite existing data.

#### Assumptions, External Events, and Notes:

- This routine will not normally wait for the receiver tasks to process the message before returning control to the caller's task.
- However, if a higher priority task is pending and subscribed to this message, that task may get to run before returning control to the caller.
- In previous versions of CFE, the boolean parameter referred to the sequence number header of telemetry messages only. This has been extended to apply more generically to any headers, as determined by the CFE MSG implementation.

#### Parameters

in	<i>MsgPtr</i>	A pointer to the message to be sent (must not be null). This must point to the first byte of the message header.
in	<i>IsOrigination</i>	Update the headers of the message

#### Returns

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

#### Return values

<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.
<a href="#">CFE_SB_BUF_ALOC_ERR</a>	(return value only verified in coverage test) Buffer Allocation Error.

Referenced by CF\_SendHkCmd().

## 10.37 cFE Zero Copy APIs

### Functions

- [CFE\\_SB\\_Buffer\\_t \\* CFE\\_SB\\_AllocateMessageBuffer](#) (*size\_t* *MsgSize*)
 

*Get a buffer pointer to use for "zero copy" SB sends.*
- [CFE\\_Status\\_t CFE\\_SB\\_ReleaseMessageBuffer](#) (*CFE\_SB\_Buffer\_t* \**BufPtr*)
 

*Release an unused "zero copy" buffer pointer.*
- [CFE\\_Status\\_t CFE\\_SB\\_TransmitBuffer](#) (*CFE\_SB\_Buffer\_t* \**BufPtr*, *bool* *IsOrigination*)
 

*Transmit a buffer.*

### 10.37.1 Detailed Description

### 10.37.2 Function Documentation

#### 10.37.2.1 CFE\_SB\_AllocateMessageBuffer() `CFE_SB_Buffer_t* CFE_SB_AllocateMessageBuffer (`

`size_t MsgSize )`

Get a buffer pointer to use for "zero copy" SB sends.

##### Description

This routine can be used to get a pointer to one of the software bus' internal memory buffers that are used for sending messages. The caller can use this memory buffer to build an SB message, then send it using the [CFE\\_SB\\_TransmitBuffer\(\)](#) function. This interface avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer.

##### Assumptions, External Events, and Notes:

1. The pointer returned by [CFE\\_SB\\_AllocateMessageBuffer\(\)](#) is only good for one call to [CFE\\_SB\\_TransmitBuffer\(\)](#).
2. Once a buffer has been successfully transmitted (as indicated by a successful return from [CFE\\_SB\\_TransmitBuffer\(\)](#)) the buffer becomes owned by the SB application. It will automatically be freed by SB once all recipients have finished reading it.
3. Applications must not de-reference the message pointer (for reading or writing) after the call to [CFE\\_SB\\_TransmitBuffer\(\)](#).
4. If [CFE\\_SB\\_ReleaseMessageBuffer](#) should be used only if a message is not transmitted

##### Parameters

<code>in</code>	<code>MsgSize</code>	The size of the SB message buffer the caller wants (including the SB message header).
-----------------	----------------------	---

##### Returns

A pointer to a memory buffer that message data can be written to for use with [CFE\\_SB\\_TransmitBuffer\(\)](#).

Referenced by CF\_CFDP\_MsgOutGet(), and CF\_CFDP\_SendEotPkt().

#### 10.37.2.2 CFE\_SB\_ReleaseMessageBuffer() `CFE_Status_t CFE_SB_ReleaseMessageBuffer (`

`CFE_SB_Buffer_t * BufPtr )`

Release an unused "zero copy" buffer pointer.

##### Description

This routine can be used to release a pointer to one of the software bus' internal memory buffers.

##### Assumptions, External Events, and Notes:

1. This function is not needed for normal "zero copy" transfers. It is needed only for cleanup when an application gets a pointer using [CFE\\_SB\\_AllocateMessageBuffer\(\)](#), but (due to some error condition) never uses that pointer in a call to [CFE\\_SB\\_TransmitBuffer\(\)](#).

**Parameters**

in	<i>BufPtr</i>	A pointer to the SB internal buffer (must not be null). This must be a pointer returned by a call to <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.

Referenced by CF\_CFDP\_MsgOutGet().

**10.37.2.3 CFE\_SB\_TransmitBuffer()** [CFE\\_Status\\_t](#) CFE\_SB\_TransmitBuffer (

```
CFE_SB_Buffer_t * BufPtr,
    bool IsOrigination )
```

Transmit a buffer.

**Description**

This routine sends a message that has been created directly in an internal SB message buffer by an application (after a call to [CFE\\_SB\\_AllocateMessageBuffer](#)). This interface is more complicated than the normal [CFE\\_SB\\_TransmitMsg](#) interface, but it avoids an extra copy of the message from the user's memory buffer to the software bus internal buffer. The "zero copy" interface can be used to improve performance in high-rate, high-volume software bus traffic.

The IsOrigination parameter should be passed as "true" if the message was newly constructed by the sender and is being sent for the first time. This enables the message origination actions as determined by the CFE MSG module, which may include (but not limited to):

- Updating sequence number
- Updating timestamp
- Calculating a CRC, checksum, or other message error control field

Conversely, when forwarding a message that originated from an external entity (e.g. messages passing through CI or SBN), the parameter should be passed as "false" to not overwrite existing data.

**Assumptions, External Events, and Notes:**

1. A handle returned by [CFE\\_SB\\_AllocateMessageBuffer](#) is "consumed" by a *successful* call to [CFE\\_SB\\_TransmitBuffer](#).
2. If this function returns [CFE\\_SUCCESS](#), this indicates the zero copy handle is now owned by software bus, and is no longer owned by the calling application, and should not be re-used.
3. However if this function fails (returns any error status) it does not change the state of the buffer at all, meaning the calling application still owns it. (a failure means the buffer is left in the same state it was before the call).
4. Applications should be written as if [CFE\\_SB\\_AllocateMessageBuffer](#) is equivalent to a `malloc()` and a successful call to [CFE\\_SB\\_TransmitBuffer](#) is equivalent to a `free()`.
5. Applications must not de-reference the message pointer (for reading or writing) after a successful call to [CFE\\_SB\\_TransmitBuffer](#).
6. This function will increment and apply the internally tracked sequence counter if set to do so.

**Parameters**

in	<i>BufPtr</i>	A pointer to the buffer to be sent (must not be null).
in	<i>IsOrigination</i>	Update applicable header field(s) of a newly constructed message

**Returns**

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

**Return values**

<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.

Referenced by CF\_CFDP\_Send(), and CF\_CFDP\_SendEotPkt().

## 10.38 cFE Message Characteristics APIs

**Functions**

- void [CFE\\_SB\\_SetUserDataLength](#) ([CFE\\_MSG\\_Message\\_t](#) \*MsgPtr, size\_t DataLength)  
*Sets the length of user data in a software bus message.*
- void [CFE\\_SB\\_TimeStampMsg](#) ([CFE\\_MSG\\_Message\\_t](#) \*MsgPtr)  
*Sets the time field in a software bus message with the current spacecraft time.*
- int32 [CFE\\_SB\\_MessageStringSet](#) (char \*DestStringPtr, const char \*SourceStringPtr, size\_t DestMaxSize, size\_t SourceMaxSize)  
*Copies a string into a software bus message.*
- void \* [CFE\\_SB\\_GetUserData](#) ([CFE\\_MSG\\_Message\\_t](#) \*MsgPtr)  
*Get a pointer to the user data portion of a software bus message.*
- size\_t [CFE\\_SB\\_GetUserDataLength](#) (const [CFE\\_MSG\\_Message\\_t](#) \*MsgPtr)  
*Gets the length of user data in a software bus message.*
- int32 [CFE\\_SB\\_MessageStringGet](#) (char \*DestStringPtr, const char \*SourceStringPtr, const char \*DefaultString, size\_t DestMaxSize, size\_t SourceMaxSize)  
*Copies a string out of a software bus message.*

### 10.38.1 Detailed Description

### 10.38.2 Function Documentation

#### 10.38.2.1 [CFE\\_SB\\_GetUserData\(\)](#) void\* [CFE\\_SB\\_GetUserData](#) ( [CFE\\_MSG\\_Message\\_t](#) \* *MsgPtr* )

Get a pointer to the user data portion of a software bus message.

**Description**

This routine returns a pointer to the user data portion of a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function and avoid hard coding offsets into their SB message buffers.

**Assumptions, External Events, and Notes:**

None

**Parameters**

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

**Returns**

A pointer to the first byte of user data within the software bus message.

**10.38.2.2 CFE\_SB\_GetUserDataLength()** `size_t CFE_SB_GetUserDataLength (`  
 `const CFE_MSG_Message_t * MsgPtr )`

Gets the length of user data in a software bus message.

**Description**

This routine returns the size of the user data in a software bus message.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
----	---------------	---

**Returns**

The size (in bytes) of the user data in the software bus message.

**Return values**

0	if an error occurs, such as if the <i>MsgPtr</i> argument is not valid.
---	---

**10.38.2.3 CFE\_SB\_MessageStringGet()** `int32 CFE_SB_MessageStringGet (`  
 `char * DestStringPtr,`  
 `const char * SourceStringPtr,`  
 `const char * DefaultString,`  
 `size_t DestMaxSize,`  
 `size_t SourceMaxSize )`

Copies a string out of a software bus message.

## Description

Strings within software bus messages have a defined/fixed maximum length, and may not necessarily be null terminated within the message. This presents a possible issue when using the C library functions to copy strings out of a message.

This function should replace use of C library functions such as strcpy/strncpy when copying strings out of software bus messages to local storage buffers.

Up to [SourceMaxSize] or [DestMaxSize-1] (whichever is smaller) characters will be copied from the source buffer to the destination buffer, and a NUL termination character will be written to the destination buffer as the last character.

If the DefaultString pointer is non-NULL, it will be used in place of the source string if the source is an empty string. This is typically a string constant that comes from the platform configuration, allowing default values to be assumed for fields that are unspecified.

**IMPORTANT** - the default string, if specified, must be null terminated. This will be the case if a string literal is passed in (the typical/expected use case).

If the default is NULL, then only the source string will be copied, and the result will be an empty string if the source was empty.

If the destination buffer is too small to store the entire string, it will be truncated, but it will still be null terminated.

## Parameters

out	<i>DestStringPtr</i>	Pointer to destination buffer (must not be null)
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="#">CFE_SB_BAD_ARGUMENT</a>	Bad Argument.
-------------------------------------	---------------

```
10.38.2.4 CFE_SB_MessageStringSet() int32 CFE_SB_MessageStringSet (
    char * DestStringPtr,
    const char * SourceStringPtr,
    size_t DestMaxSize,
    size_t SourceMaxSize )
```

Copies a string into a software bus message.

## Description

Strings within software bus messages have a defined/fixed maximum length, and may not necessarily be null terminated within the message. This presents a possible issue when using the C library functions to copy strings out of a message.

This performs a very similar function to "strncpy()" except that the sizes of *both* buffers are passed in. Neither buffer is required to be null-terminated, but copying will stop after the first termination character is encountered.

If the destination buffer is not completely filled by the source data (such as if the supplied string was shorter than the allotted length) the destination buffer will be padded with NUL characters up to the size of the buffer, similar to what

`strncpy()` does. This ensures that the entire destination buffer is set.

#### Note

If the source string buffer is already guaranteed to be null terminated, then there is no difference between the C library "strncpy()" function and this implementation. It is only necessary to use this when termination of the source buffer is not guaranteed.

#### Parameters

out	<i>DestStringPtr</i>	Pointer to destination buffer (component of SB message definition) (must not be null)
in	<i>SourceStringPtr</i>	Pointer to source buffer (must not be null)
in	<i>DestMaxSize</i>	Size of destination buffer as defined by the message definition
in	<i>SourceMaxSize</i>	Size of source buffer

#### Returns

Number of characters copied or error code, see [cFE Return Code Defines](#)

#### Return values

<a href="#"><i>CFE_SB_BAD_ARGUMENT</i></a>	Bad Argument.
--	---------------

#### 10.38.2.5 [\*CFE\\_SB\\_SetUserDataLength\(\)\*](#) void *CFE\_SB\_SetUserDataLength* (

```
    CFE_MSG_Message_t * MsgPtr,
    size_t DataLength )
```

Sets the length of user data in a software bus message.

#### Description

This routine sets the field in the SB message header that determines the size of the user data in a software bus message. SB message header formats can be different for each deployment of the cFE. So, applications should use this function rather than trying to poke a length value directly into their SB message buffers.

#### Assumptions, External Events, and Notes:

- You must set a valid message ID in the SB message header before calling this function.

#### Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
in	<i>DataLength</i>	The length to set (size of the user data, in bytes).

#### 10.38.2.6 [\*CFE\\_SB\\_TimeStampMsg\(\)\*](#) void *CFE\_SB\_TimeStampMsg* (

```
    CFE_MSG_Message_t * MsgPtr )
```

Sets the time field in a software bus message with the current spacecraft time.

### Description

This routine sets the time of a software bus message with the current spacecraft time. This will be the same time that is returned by the function [CFE\\_TIME\\_GetTime](#).

### Assumptions, External Events, and Notes:

- If the underlying implementation of software bus messages does not include a time field, then this routine will do nothing.

### Parameters

in	<i>MsgPtr</i>	A pointer to the buffer that contains the software bus message (must not be null). This must point to the first byte of the message header.
----	---------------	---

Referenced by CF\_CFDP\_SendEotPkt(), and CF\_SendHkCmd().

## 10.39 cFE Message ID APIs

### Functions

- bool [CFE\\_SB\\_IsValidMsgId](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId)  
*Identifies whether a given [CFE\\_SB\\_MsgId\\_t](#) is valid.*
- static bool [CFE\\_SB\\_MsgId\\_Equal](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId1, [CFE\\_SB\\_MsgId\\_t](#) MsgId2)  
*Identifies whether two [CFE\\_SB\\_MsgId\\_t](#) values are equal.*
- static [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_MsgIdToValue](#) ([CFE\\_SB\\_MsgId\\_t](#) MsgId)  
*Converts a [CFE\\_SB\\_MsgId\\_t](#) to a normal integer.*
- static [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_ValueToMsgId](#) ([CFE\\_SB\\_MsgId\\_Atom\\_t](#) MsgIdValue)  
*Converts a normal integer into a [CFE\\_SB\\_MsgId\\_t](#).*
- [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_CmdTopicIdToMsgId](#) ([uint16](#) TopicId, [uint16](#) InstanceNum)  
*Converts a topic ID and instance number combination into a MsgID value integer.*
- [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_TlmTopicIdToMsgId](#) ([uint16](#) TopicId, [uint16](#) InstanceNum)  
*Converts a topic ID and instance number combination into a MsgID value integer.*
- [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_GlobalCmdTopicIdToMsgId](#) ([uint16](#) TopicId)  
*Converts a topic ID to a MsgID value integer for Global commands.*
- [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_GlobalTlmTopicIdToMsgId](#) ([uint16](#) TopicId)  
*Converts a topic ID to a MsgID value integer for Global telemetry.*
- [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_LocalCmdTopicIdToMsgId](#) ([uint16](#) TopicId)  
*Converts a topic ID to a MsgID value integer for local commands.*
- [CFE\\_SB\\_MsgId\\_Atom\\_t](#) [CFE\\_SB\\_LocalTlmTopicIdToMsgId](#) ([uint16](#) TopicId)  
*Converts a topic ID to a MsgID value integer for local telemetry.*

### 10.39.1 Detailed Description

### 10.39.2 Function Documentation

```
10.39.2.1 CFE_SB_CmdTopicIdToMsgId() CFE_SB_MsgId_Atom_t CFE_SB_CmdTopicIdToMsgId (
    uint16 TopicId,
    uint16 InstanceNum )
```

Converts a topic ID and instance number combination into a MsgID value integer.

#### Description

This function accepts a data pair of topic ID + instance number and returns the corresponding MsgID Value (integer) for commands.

#### Assumptions and Notes:

A topic ID identifies a certain data stream from an application, for example the CFE Software bus ground commands (CFE\_MISSION\_SB\_CMD\_TOPICID). In contrast to MsgID, the topic ID is consistent across all CPUs in a system, whereas each CPU instance will have a unique MsgID.

CPU instance numbers are 1-based. The instance number of 0 is reserved for "global" MsgID values that are the same on all CPUs. The PSP function may be used to obtain the current CPU number for the host processor.

#### See also

[CFE\\_SB\\_TlmTopicIdToMsgId\(\)](#), [CFE\\_PSP\\_GetProcessorId\(\)](#)

#### Returns

Integer representation of the [CFE\\_SB\\_MsgId\\_t](#)

```
10.39.2.2 CFE_SB_GlobalCmdTopicIdToMsgId() CFE_SB_MsgId_Atom_t CFE_SB_GlobalCmdTopicIdToMsgId (
    uint16 TopicId )
```

Converts a topic ID to a MsgID value integer for Global commands.

#### Description

This is a wrapper around [CFE\\_SB\\_CmdTopicIdToMsgId\(\)](#) for topic IDs which are the same on all CPUs within a system (i.e. not specific to a certain processor)

#### Assumptions and Notes:

Global MsgIDs may be used when only a single instance of a service exists within the system. The CFE framework does not use this feature for commands, but is defined for future use.

#### See also

[CFE\\_SB\\_CmdTopicIdToMsgId\(\)](#), [CFE\\_SB\\_LocalCmdTopicIdToMsgId\(\)](#)

#### Returns

Integer representation of the [CFE\\_SB\\_MsgId\\_t](#)

**10.39.2.3 CFE\_SB\_GlobalTlmTopicIdToMsgId()** `CFE_SB_MsgId_Atom_t` `CFE_SB_GlobalTlmTopicIdToMsgId (`  
`uint16 TopicId )`

Converts a topic ID to a MsgID value integer for Global telemetry.

#### Description

This is a wrapper around [CFE\\_SB\\_TlmTopicIdToMsgId\(\)](#) for topic IDs which are the same on all CPUs within a system (i.e. not specific to a certain processor)

#### Assumptions and Notes:

Global MsgIDs may be used when only a single instance of a service exists within the system. An example for such telemetry is the time synchronization service published by CFE\_TIME.

#### See also

[CFE\\_SB\\_TlmTopicIdToMsgId\(\)](#), [CFE\\_SB\\_LocalTlmTopicIdToMsgId\(\)](#)

#### Returns

Integer representation of the [CFE\\_SB\\_MsgId\\_t](#)

**10.39.2.4 CFE\_SB\_IsValidMsgId()** `bool` `CFE_SB_IsValidMsgId (`  
`CFE_SB_MsgId_t MsgId )`

Identifies whether a given [CFE\\_SB\\_MsgId\\_t](#) is valid.

#### Description

Implements a basic sanity check on the value provided

#### Returns

Boolean message ID validity indicator

#### Return values

<code>true</code>	Message ID is within the valid range
<code>false</code>	Message ID is not within the valid range

Referenced by CF\_AppPipe().

**10.39.2.5 CFE\_SB\_LocalCmdTopicIdToMsgId()** `CFE_SB_MsgId_Atom_t` `CFE_SB_LocalCmdTopicIdToMsgId (`  
`uint16 TopicId )`

Converts a topic ID to a MsgID value integer for local commands.

#### Description

This is a wrapper around [CFE\\_SB\\_CmdTopicIdToMsgId\(\)](#) for topic IDs which are unique on all CPUs within a system (i.e. specific to a certain processor)

**Assumptions and Notes:**

This assumes the caller is referring to a service running on the same processor instance as itself.

**See also**

[CFE\\_SB\\_CmdTopicIdToMsgId\(\)](#), [CFE\\_SB\\_LocalTlmTopicIdToMsgId\(\)](#)

**Returns**

Integer representation of the [CFE\\_SB\\_MsgId\\_t](#)

**10.39.2.6 CFE\_SB\_LocalTlmTopicIdToMsgId()** [CFE\\_SB\\_MsgId\\_Atom\\_t](#) CFE\_SB\_LocalTlmTopicIdToMsgId (

`uint16 TopicId )`

Converts a topic ID to a MsgID value integer for local telemetry.

**Description**

This is a wrapper around [CFE\\_SB\\_TlmTopicIdToMsgId\(\)](#) for topic IDs which are unique on all CPUs within a system (i.e. specific to a certain processor)

**Assumptions and Notes:**

This assumes the caller is referring to a service running on the same processor instance as itself.

**See also**

[CFE\\_SB\\_TlmTopicIdToMsgId\(\)](#), [CFE\\_SB\\_LocalCmdTopicIdToMsgId\(\)](#)

**Returns**

Integer representation of the [CFE\\_SB\\_MsgId\\_t](#)

**10.39.2.7 CFE\_SB\_MsgId\_Equal()** static bool CFE\_SB\_MsgId\_Equal (

`CFE_SB_MsgId_t MsgId1,`

`CFE_SB_MsgId_t MsgId2 ) [inline], [static]`

Identifies whether two [CFE\\_SB\\_MsgId\\_t](#) values are equal.

**Description**

In cases where the [CFE\\_SB\\_MsgId\\_t](#) type is not a simple integer type, it may not be possible to do a direct equality check. This inline function provides an abstraction for the equality check between two [CFE\\_SB\\_MsgId\\_t](#) values.

Applications should transition to using this function to compare MsgId values for equality to remain compatible with future versions of cFE.

**Returns**

Boolean message ID equality indicator

**Return values**

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

Definition at line 791 of file cfe\_sb.h.  
References CFE\_SB\_MSGID\_UNWRAP\_VALUE.  
Referenced by CF\_AppPipe().

**10.39.2.8 CFE\_SB\_MsgIdToValue()** static [CFE\\_SB\\_MsgId\\_Atom\\_t](#) CFE\_SB\_MsgIdToValue ( [CFE\\_SB\\_MsgId\\_t](#) MsgId ) [inline], [static]

Converts a [CFE\\_SB\\_MsgId\\_t](#) to a normal integer.

#### Description

In cases where the [CFE\\_SB\\_MsgId\\_t](#) type is not a simple integer type, it is not possible to directly display the value in a printf-style statement, use it in a switch() statement, or other similar use cases.

This inline function provides the ability to map a [CFE\\_SB\\_MsgId\\_t](#) type back into a simple integer value. Applications should transition to using this function wherever a [CFE\\_SB\\_MsgId\\_t](#) type needs to be used as an integer.

#### Assumptions and Notes:

This negates the type safety that was gained by using a non-integer type for the [CFE\\_SB\\_MsgId\\_t](#) value. This should only be used in specific cases such as UI display (printf, events, etc) where the value is being sent externally. Any internal API calls should be updated to use the [CFE\\_SB\\_MsgId\\_t](#) type directly, rather than an integer type.

#### Returns

Integer representation of the [CFE\\_SB\\_MsgId\\_t](#)

Definition at line 822 of file cfe\_sb.h.  
References CFE\_SB\_MSGID\_UNWRAP\_VALUE.  
Referenced by CF\_AppPipe().

**10.39.2.9 CFE\_SB\_TlmTopicIdToMsgId()** [CFE\\_SB\\_MsgId\\_Atom\\_t](#) CFE\_SB\_TlmTopicIdToMsgId ( [uint16](#) TopicId, [uint16](#) InstanceNum )

Converts a topic ID and instance number combination into a MsgID value integer.

#### Description

This function accepts a data pair of topic ID + instance number and returns the corresponding MsgID Value (integer) for telemetry.

#### Assumptions and Notes:

A topic ID identifies a certain data stream from an application, for example the CFE Software bus housekeeping telemetry (CFE\_MISSION\_SB\_HK\_TLM\_TOPICID). In contrast to MsgID, the topic ID is consistent across all CPUs in a system, whereas each CPU instance will have a unique MsgID.

CPU instance numbers are 1-based. The instance number of 0 is reserved for "global" MsgID values that are the same on all CPUs. The PSP function may be used to obtain the current CPU number for the host processor.

#### See also

[CFE\\_SB\\_CmdTopicIdToMsgId\(\)](#), [CFE\\_PSP\\_GetProcessorId\(\)](#)

#### Returns

Integer representation of the [CFE\\_SB\\_MsgId\\_t](#)

**10.39.2.10 CFE\_SB\_ValueToMsgId()** static [CFE\\_SB\\_MsgId\\_t](#) CFE\_SB\_ValueToMsgId ( [CFE\\_SB\\_MsgId\\_Atom\\_t](#) MsgIdValue ) [inline], [static]  
Converts a normal integer into a [CFE\\_SB\\_MsgId\\_t](#).

#### Description

In cases where the [CFE\\_SB\\_MsgId\\_t](#) type is not a simple integer type, it is not possible to directly use an integer value supplied via a define or similar method.

This inline function provides the ability to map an integer value into a corresponding [CFE\\_SB\\_MsgId\\_t](#) value. Applications should transition to using this function wherever an integer needs to be used for a [CFE\\_SB\\_MsgId\\_t](#).

#### Assumptions and Notes:

This negates the type safety that was gained by using a non-integer type for the [CFE\\_SB\\_MsgId\\_t](#) value. This should only be used in specific cases where the value is coming from an external source. Any internal API calls should be updated to return the [CFE\\_SB\\_MsgId\\_t](#) type directly, rather than an integer type.

#### Returns

[CFE\\_SB\\_MsgId\\_t](#) representation of the integer

Definition at line 851 of file cfe\_sb.h.

References [CFE\\_SB\\_MSGID\\_WRAP\\_VALUE](#).

Referenced by [CF\\_AppInit\(\)](#), [CF\\_AppPipe\(\)](#), [CF\\_CFDP\\_InitEngine\(\)](#), [CF\\_CFDP\\_MsgOutGet\(\)](#), and [CF\\_CFDP\\_SendEotPkt\(\)](#).

## 10.40 cFE SB Pipe options

### Macros

- #define [CFE\\_SB\\_PIPEOPTS\\_IGNOREMINE](#) 0x00000001

*Messages sent by the app that owns this pipe will not be sent to this pipe.*

#### 10.40.1 Detailed Description

#### 10.40.2 Macro Definition Documentation

##### 10.40.2.1 CFE\_SB\_PIPEOPTS\_IGNOREMINE #define CFE\_SB\_PIPEOPTS\_IGNOREMINE 0x00000001

Messages sent by the app that owns this pipe will not be sent to this pipe.

Definition at line 132 of file cfe\_sb\_api\_typedefs.h.

## 10.41 cFE Registration APIs

### Functions

- [CFE\\_Status\\_t CFE\\_TBL\\_Register](#) ([CFE\\_TBL\\_Handle\\_t](#) \*TblHandlePtr, const char \*Name, size\_t Size, uint16 TblOptionFlags, [CFE\\_TBL\\_CallbackFuncPtr\\_t](#) TblValidationFuncPtr)  
*Register a table with cFE to obtain Table Management Services.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Share](#) ([CFE\\_TBL\\_Handle\\_t](#) \*TblHandlePtr, const char \*TblName)  
*Obtain handle of table registered by another application.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Unregister](#) ([CFE\\_TBL\\_Handle\\_t](#) TblHandle)  
*Unregister a table.*

### 10.41.1 Detailed Description

### 10.41.2 Function Documentation

**10.41.2.1 CFE\_TBL\_Register()** `CFE_Status_t CFE_TBL_Register (`  
 `CFE_TBL_Handle_t * TblHandlePtr,`  
 `const char * Name,`  
 `size_t Size,`  
 `uint16 TblOptionFlags,`  
 `CFE_TBL_CallbackFuncPtr_t TblValidationFuncPtr )`

Register a table with cFE to obtain Table Management Services.

#### Description

When an application is created and initialized, it is responsible for creating its table images via the TBL API. The application must inform the Table Service of the table name, table size and selection of optional table features.

#### Assumptions, External Events, and Notes:

Note: This function call can block. Therefore, interrupt service routines should NOT create their own tables. An application should create any table(s) and provide the handle(s) to the interrupt service routine.

#### Parameters

out	<i>TblHandlePtr</i>	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	<i>Name</i>	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	<i>Size</i>	The size, in bytes, of the table to be created (must not be zero). This is the size that will be allocated as a shared memory resource between the Table Management Service and the calling application.

## Parameters

in	<i>TblOptionFlags</i>	<p>Flag bits indicating selected options for table. A bitwise OR of the following option flags:</p> <ul style="list-style-type: none"> <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</li> </ul>
Generated by Doxygen		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

**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 [CF\\_TableInit\(\)](#).

**10.41.2.2 CFE\_TBL\_Share()** [CFE\\_Status\\_t](#) CFE\_TBL\_Share (

```
CFE_TBL_Handle_t * TblHandlePtr,
const char * TblName )
```

Obtain handle of table registered by another application.

### Description

After a table has been created, other applications can gain access to that table via the table handle. In order for two or more applications to share a table, the applications that do not create the table must obtain the handle using this function.

### Assumptions, External Events, and Notes:

None

### Parameters

<i>out</i>	<i>TblHandlePtr</i>	A pointer to a <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](#)

### 10.41.2.3 [CFE\\_TBL\\_Unregister\(\)](#)

```
CFE_Status_t CFE_TBL_Unregister (
    CFE_TBL_Handle_t TblHandle )
```

Unregister a table.

### Description

When an application is being removed from the system, ES will clean up/free all the application related resources including tables so apps are not required to call this function.

A valid use-case for this API is to unregister a shared table if access is no longer needed or the owning application was removed from the system (CS app is an example).

Typically apps should only register tables during initialization and registration/unregistration by the owning application during operation should be avoided. If unavoidable, special care needs to be taken (especially for shared tables) to avoid race conditions due to competing requests from multiple tasks.

Note the table will not be removed from memory until all table access links have been removed (registration and all shared access).

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <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](#)

## 10.42 cFE Manage Table Content APIs

**Functions**

- [CFE\\_Status\\_t CFE\\_TBL\\_Load \(CFE\\_TBL\\_Handle\\_t TblHandle, CFE\\_TBL\\_SrcEnum\\_t SrcType, const void \\*SrcDataPtr\)](#)  
*Load a specified table with data from specified source.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Update \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Update contents of a specified table, if an update is pending.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Validate \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Perform steps to validate the contents of a table image.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Manage \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Perform standard operations to maintain a table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_DumpToBuffer \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Copies the contents of a Dump Only Table to a shared buffer.*
- [CFE\\_Status\\_t CFE\\_TBL\\_Modified \(CFE\\_TBL\\_Handle\\_t TblHandle\)](#)  
*Notify cFE Table Services that table contents have been modified by the Application.*

### 10.42.1 Detailed Description

### 10.42.2 Function Documentation

**10.42.2.1 CFE\_TBL\_DumpToBuffer()** `CFE_Status_t CFE_TBL_DumpToBuffer (`  
`CFE_TBL_Handle_t TblHandle )`

Copies the contents of a Dump Only Table to a shared buffer.

**Description**

Typically, apps should just call [CFE\\_TBL\\_Manage](#) as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just a dump should be performed.

**Assumptions, External Events, and Notes:**

If the table does not have a dump pending status, nothing will occur (no error, no dump)

**Parameters**

in	<i>TblHandle</i>	Handle of Table to be dumped.
----	------------------	-------------------------------

**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.
<a href="#">CFE_TBL_INFO_DUMP_PENDING</a>	Dump Pending.

**See also**

[CFE\\_TBL\\_Manage](#)

**10.42.2.2 CFE\_TBL\_Load()** `CFE_Status_t CFE_TBL_Load (`

```
    CFE_TBL_Handle_t TblHandle,  
    CFE_TBL_SrcEnum_t SrcType,  
    const void * SrcDataPtr )
```

Load a specified table with data from specified source.

**Description**

Once an application has created a table ([CFE\\_TBL\\_Register](#)), it must provide the values that initialize the contents of that table. The application accomplishes this with one of two different TBL API calls. This function call initializes the table with values that are held in a data structure.

**Assumptions, External Events, and Notes:**

This function call can block. Therefore, interrupt service routines should NOT initialize their own tables. An application should initialize any table(s) prior to providing the handle(s) to the interrupt service routine.

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <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.
<a href="#">CFE_TBL_ERR_INVALID_HANDLE</a>	Invalid Handle.
<a href="#">CFE_TBL_ERR_DUMP_ONLY</a>	Dump Only Error.
<a href="#">CFE_TBL_ERR_ILLEGAL_SRC_TYPE</a>	Illegal Source Type.
<a href="#">CFE_TBL_ERR_LOAD_IN_PROGRESS</a>	Load In Progress.
<a href="#">CFE_TBL_ERR_LOAD_INCOMPLETE</a>	Load Incomplete.
<a href="#">CFE_TBL_ERR_NO_BUFFER_AVAIL</a>	No Buffer Available.
<a href="#">CFE_TBL_ERR_ACCESS</a>	
<a href="#">CFE_TBL_ERR_FILE_TOO_LARGE</a>	File Too Large.
<a href="#">CFE_TBL_ERR_BAD_CONTENT_ID</a>	Bad Content ID.
<a href="#">CFE_TBL_ERR_BAD_SUBTYPE_ID</a>	Bad Subtype ID.
<a href="#">CFE_TBL_ERR_NO_STD_HEADER</a>	No Standard Header.
<a href="#">CFE_TBL_ERR_NO_TBL_HEADER</a>	No Table Header.
<a href="#">CFE_TBL_ERR_PARTIAL_LOAD</a>	Partial Load Error.
<a href="#">CFE_TBL_BAD_ARGUMENT</a>	Bad Argument.

**See also**

[CFE\\_TBL\\_Update](#), [CFE\\_TBL\\_Validate](#), [CFE\\_TBL\\_Manage](#)

Referenced by CF\_TableInit().

**10.42.2.3 CFE\_TBL\_Manage()** `CFE_Status_t CFE_TBL_Manage (`  
`CFE_TBL_Handle_t TblHandle )`

Perform standard operations to maintain a table.

**Description**

Applications should call this API periodically to process pending requests for update, validation, or dump to buffer. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <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_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_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 CF\_CheckTables(), and CF\_TableInit().

**10.42.2.4 CFE\_TBL\_Modified()** `CFE_Status_t CFE_TBL_Modified (`  
`CFE_TBL_Handle_t TblHandle )`

Notify cFE Table Services that table contents have been modified by the Application.

**Description**

This API notifies Table Services that the contents of the specified table has been modified by the Application. This notification is important when a table has been registered as "Critical" because Table Services can then update the contents of the table kept in the Critical Data Store.

**Assumptions, External Events, and Notes:**

None

**Parameters**

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

**10.42.2.5 CFE\_TBL\_Update()** [CFE\\_Status\\_t](#) CFE\_TBL\_Update ( [CFE\\_TBL\\_Handle\\_t](#) *TblHandle* )

Update contents of a specified table, if an update is pending.

**Description**

Typically, apps should just call [CFE\\_TBL\\_Manage](#) as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just an update should be performed.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <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**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_NO_UPDATE_PENDING</i>	No Update Pending.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.

**See also**

[CFE\\_TBL\\_Load](#), [CFE\\_TBL\\_Validate](#), [CFE\\_TBL\\_Manage](#)

#### 10.42.2.6 CFE\_TBL\_Validate()

```
CFE_Status_t CFE_TBL_Validate (
    CFE_TBL_Handle_t TblHandle )
```

Perform steps to validate the contents of a table image.

**Description**

Typically, apps should just call [CFE\\_TBL\\_Manage](#) as part of routine processing which will perform validation, update, or dump if pending. This API is provided for the case where just a validation should be performed.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>TblHandle</i>	Handle, previously obtained from <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_NO_VALIDATION_PENDING</i>	
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.

**See also**

[CFE\\_TBL\\_Update](#), [CFE\\_TBL\\_Manage](#), [CFE\\_TBL\\_Load](#)

## 10.43 cFE Access Table Content APIs

### Functions

- [CFE\\_Status\\_t CFE\\_TBL\\_GetAddress](#) (void \*\*TblPtr, [CFE\\_TBL\\_Handle\\_t](#) TblHandle)  
*Obtain the current address of the contents of the specified table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_ReleaseAddress](#) ([CFE\\_TBL\\_Handle\\_t](#) TblHandle)  
*Release previously obtained pointer to the contents of the specified table.*
- [CFE\\_Status\\_t CFE\\_TBL\\_GetAddresses](#) (void \*\*TblPtrs[], uint16 NumTables, const [CFE\\_TBL\\_Handle\\_t](#) TblHandles[])  
*Obtain the current addresses of an array of specified tables.*
- [CFE\\_Status\\_t CFE\\_TBL\\_ReleaseAddresses](#) (uint16 NumTables, const [CFE\\_TBL\\_Handle\\_t](#) TblHandles[])  
*Release the addresses of an array of specified tables.*

### 10.43.1 Detailed Description

### 10.43.2 Function Documentation

**10.43.2.1 CFE\_TBL\_GetAddress()** [CFE\\_Status\\_t](#) CFE\_TBL\_GetAddress ( void \*\* TblPtr,  
                                 [CFE\\_TBL\\_Handle\\_t](#) TblHandle )

Obtain the current address of the contents of the specified table.

#### Description

When a table has been created and initialized, it is available to any application that can identify it with its unique handle. In order to view the data contained in the table, an application must call this function or [CFE\\_TBL\\_GetAddresses](#).

#### Assumptions, External Events, and Notes:

1. This call can be a blocking call when the table is not double buffered and is shared with another application of lower priority that just happens to be in the middle of a table update of the specific table. If this occurs, the application performing the table update will automatically have its priority elevated in order to release the resource as soon as possible.
2. An application must always release the returned table address using the [CFE\\_TBL\\_ReleaseAddress](#) or [CFE\\_TBL\\_ReleaseAddresses](#) function prior to either a [CFE\\_TBL\\_Update](#) call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.
3. [CFE\\_TBL\\_ERR\\_NEVER\\_LOADED](#) will be returned if the table has never been loaded (either from file or from a block of memory), but the function will still return a valid table pointer to a table with all zero content. This pointer must be released with the [CFE\\_TBL\\_ReleaseAddress](#) API before the table can be loaded with data.

**Parameters**

<b>out</b>	<i>TblPtr</i>	The address of a pointer (must not be null) that will be loaded with the address of the first byte of the table. This pointer can then be typecast by the calling application to the appropriate table data structure. *TblPtr is the address of the first byte of data associated with the specified table.
<b>in</b>	<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 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 CF\_CheckTables(), and CF\_TableInit().

**10.43.2.2 CFE\_TBL\_GetAddresses()** [CFE\\_Status\\_t](#) CFE\_TBL\_GetAddresses (

```
void ** TblPtrs[],
uint16 NumTables,
const CFE_TBL_Handle_t TblHandles[] )
```

Obtain the current addresses of an array of specified tables.

**Description**

When a table has been created and initialized, it is available to any application that can identify it with its unique handle. In order to view the data contained in the table, an application must call this function or [CFE\\_TBL\\_GetAddress](#).

**Assumptions, External Events, and Notes:**

1. This call can be a blocking call when the table is not double buffered and is shared with another application of lower priority that just happens to be in the middle of a table update of the specific table. If this occurs, the application performing the table update will automatically have its priority elevated in order to release the resource as soon as possible.
2. An application must always release the returned table address using the [CFE\\_TBL\\_ReleaseAddress](#) or [CFE\\_TBL\\_ReleaseAddresses](#) function prior to either a [CFE\\_TBL\\_Update](#) call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

3. `CFE_TBL_ERR_NEVER_LOADED` will be returned if the table has never been loaded (either from file or from a block of memory), but the function will still return a valid table pointer to a table with all zero content. This pointer must be released with the `CFE_TBL_ReleaseAddress` API before the table can be loaded with data.

#### Parameters

<code>out</code>	<code>TblPtrs</code>	Array of Pointers (must not be null) to variables that calling Application wishes to hold the start addresses of the Tables. <code>*TblPtrs</code> is an array of addresses of the first byte of data associated with the specified tables.
<code>in</code>	<code>NumTables</code>	Size of <code>TblPtrs</code> and <code>TblHandles</code> arrays.
<code>in</code>	<code>TblHandles</code>	Array of Table Handles, previously obtained from <code>CFE_TBL_Register</code> or <code>CFE_TBL_Share</code> , of those tables whose start addresses are to be obtained.

#### Returns

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

#### Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_TBL_INFO_UPDATED</code>	Updated.
<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_ERR_UNREGISTERED</code>	Unregistered.
<code>CFE_TBL_ERR_NEVER_LOADED</code>	Never Loaded.
<code>CFE_TBL_BAD_ARGUMENT</code>	Bad Argument.

#### See also

[CFE\\_TBL\\_GetAddress](#), [CFE\\_TBL\\_ReleaseAddress](#), [CFE\\_TBL\\_ReleaseAddresses](#)

#### 10.43.2.3 CFE\_TBL\_ReleaseAddress()

```
CFE_Status_t CFE_TBL_ReleaseAddress (
    CFE_TBL_Handle_t TblHandle )
```

Release previously obtained pointer to the contents of the specified table.

#### Description

Each application is **required** to release a table address obtained through the `CFE_TBL_GetAddress` function.

#### Assumptions, External Events, and Notes:

An application must always release the returned table address using the `CFE_TBL_ReleaseAddress` function prior to either a `CFE_TBL_Update` call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

**Parameters**

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 whose address is to be released.
----	------------------	---

**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_NEVER_LOADED</a>	Never Loaded.

**See also**

[CFE\\_TBL\\_GetAddress](#), [CFE\\_TBL\\_GetAddresses](#), [CFE\\_TBL\\_ReleaseAddresses](#)

Referenced by CF\_CheckTables().

**10.43.2.4 CFE\_TBL\_ReleaseAddresses()** [CFE\\_Status\\_t](#) CFE\_TBL\_ReleaseAddresses (   
  [uint16](#) NumTables,  
  const [CFE\\_TBL\\_Handle\\_t](#) TblHandles[] )

Release the addresses of an array of specified tables.

**Description**

Each application is **required** to release a table address obtained through the [CFE\\_TBL\\_GetAddress](#) function.

**Assumptions, External Events, and Notes:**

An application must always release the returned table address using the [CFE\\_TBL\\_ReleaseAddress](#) function prior to either a [CFE\\_TBL\\_Update](#) call or any blocking call (e.g. - pending on software bus message, etc). Table updates cannot occur while table addresses have not been released.

**Parameters**

in	<i>NumTables</i>	Size of TblHandles array.
in	<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.
<i>CFE_TBL_ERR_NEVER_LOADED</i>	Never Loaded.
<i>CFE_TBL_BAD_ARGUMENT</i>	Bad Argument.

### See also

[CFE\\_TBL\\_GetAddress](#), [CFE\\_TBL\\_ReleaseAddress](#), [CFE\\_TBL\\_GetAddresses](#)

## 10.44 cFE Get Table Information APIs

### Functions

- ***CFE\_Status\_t CFE\_TBL\_GetStatus (CFE\_TBL\_Handle\_t TblHandle)***  
*Obtain current status of pending actions for a table.*
- ***CFE\_Status\_t CFE\_TBL\_GetInfo (CFE\_TBL\_Info\_t \*TblInfoPtr, const char \*TblName)***  
*Obtain characteristics/information of/about a specified table.*
- ***CFE\_Status\_t CFE\_TBL\_NotifyByMessage (CFE\_TBL\_Handle\_t TblHandle, CFE\_SB\_MsgId\_t MsgId, CFE\_MSG\_FcnCode\_t CommandCode, uint32 Parameter)***  
*Instruct cFE Table Services to notify Application via message when table requires management.*

### 10.44.1 Detailed Description

### 10.44.2 Function Documentation

**10.44.2.1 CFE\_TBL\_GetInfo()** *CFE\_Status\_t CFE\_TBL\_GetInfo (*  
*CFE\_TBL\_Info\_t \* TblInfoPtr,*  
*const char \* TblName )*

Obtain characteristics/information of/about a specified table.

#### Description

This API provides the registry information associated with the specified table. The function fills the given data structure with the data found in the Table Registry.

#### Assumptions, External Events, and Notes:

None

#### Parameters

<i>out</i>	<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.
<i>in</i>	<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**

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

**See also**

[CFE\\_TBL\\_GetStatus](#)

**10.44.2.2 CFE\_TBL\_GetStatus()** [\*CFE\\_Status\\_t\*](#) *CFE\_TBL\_GetStatus* ( [\*CFE\\_TBL\\_Handle\\_t\*](#) *TblHandle* )

Obtain current status of pending actions for a table.

**Description**

An application is **required** to perform a periodic check for an update or a validation request for all the tables that it creates. Typically, the application that created the table would call this function at the start or conclusion of any routine processing cycle. If a table update or validation request is pending, the Application should follow up with a call to [CFE\\_TBL\\_Update](#) or [CFE\\_TBL\\_Validate](#) respectively.

**Assumptions, External Events, and Notes:**

None

**Parameters**

<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 to be managed.
-----------	------------------	---

**Returns**

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

**Return values**

<i>CFE_SUCCESS</i>	Successful execution.
<i>CFE_TBL_INFO_UPDATE_PENDING</i>	Update Pending.
<i>CFE_TBL_INFO_VALIDATION_PENDING</i>	
<i>CFE_TBL_INFO_DUMP_PENDING</i>	Dump Pending.
<i>CFE_ES_ERR_RESOURCEID_NOT_VALID</i>	Resource ID is not valid.
<i>CFE_TBL_ERR_NO_ACCESS</i>	No Access.
<i>CFE_TBL_ERR_INVALID_HANDLE</i>	Invalid Handle.

**Note**

Some status return codes are "success" while being non-zero. This behavior will change in the future.

**See also**

[CFE\\_TBL\\_Manage](#), [CFE\\_TBL\\_Update](#), [CFE\\_TBL\\_Validate](#), [CFE\\_TBL\\_GetInfo](#)

**10.44.2.3 CFE\_TBL\_NotifyByMessage()** [CFE\\_Status\\_t](#) CFE\_TBL\_NotifyByMessage (

```
    CFE_TBL_Handle_t TblHandle,  
    CFE_SB_MsgId_t MsgId,  
    CFE_MSG_FcnCode_t CommandCode,  
    uint32 Parameter )
```

Instruct cFE Table Services to notify Application via message when table requires management.

**Description**

This API instructs Table Services to send a message to the calling Application whenever the specified table requires management by the application. This feature allows applications to avoid polling table services via the [CFE\\_TBL\\_Manage](#) call to determine whether a table requires updates, validation, etc. This API should be called following the [CFE\\_TBL\\_Register](#) API whenever the owning application requires this feature.

**Assumptions, External Events, and Notes:**

- Only the application that owns the table is allowed to register a notification message
- Recommend **NOT** using the ground command MID which typically impacts command counters. The typical approach is to use a unique MID for inter-task communications similar to how schedulers typically trigger application housekeeping messages.

**Parameters**

in	<i>TblHandle</i>	Handle of Table with which the message should be associated.
in	<i>MsgId</i>	Message ID to be used in notification message sent by Table Services.
in	<i>CommandCode</i>	Command Code value to be placed in secondary header of message sent by Table Services.
in	<i>Parameter</i>	Application defined value to be passed as a parameter in the message sent by Table Services. Suggested use includes an application's table index that allows the same MsgId and Command Code to be used for all table management notifications.

**Returns**

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

**Return values**

<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](#)

## 10.45 cFE Table Type Defines

### Macros

- `#define CFE_TBL_OPT_BUFFER_MSK (0x0001)`  
*Table buffer mask.*
- `#define CFE_TBL_OPT_SNGL_BUFFER (0x0000)`  
*Single buffer table.*
- `#define CFE_TBL_OPT_DBL_BUFFER (0x0001)`  
*Double buffer table.*
- `#define CFE_TBL_OPT_LD_DMP_MSK (0x0002)`  
*Table load/dump mask.*
- `#define CFE_TBL_OPT_LOAD_DUMP (0x0000)`  
*Load/Dump table.*
- `#define CFE_TBL_OPT_DUMP_ONLY (0x0002)`  
*Dump only table.*
- `#define CFE_TBL_OPT_USR_DEF_MSK (0x0004)`  
*Table user defined mask.*
- `#define CFE_TBL_OPT_NOT_USR_DEF (0x0000)`  
*Not user defined table.*
- `#define CFE_TBL_OPT_USR_DEF_ADDR (0x0006)`  
*User Defined table.,*
- `#define CFE_TBL_OPT_CRITICAL_MSK (0x0008)`  
*Table critical mask.*
- `#define CFE_TBL_OPT_NOT_CRITICAL (0x0000)`  
*Not critical table.*
- `#define CFE_TBL_OPT_CRITICAL (0x0008)`  
*Critical table.*
- `#define CFE_TBL_OPT_DEFAULT (CFE_TBL_OPT_SNGL_BUFFER | CFE_TBL_OPT_LOAD_DUMP)`  
*Default table options.*

### 10.45.1 Detailed Description

### 10.45.2 Macro Definition Documentation

#### 10.45.2.1 CFE\_TBL\_OPT\_BUFFER\_MSK `#define CFE_TBL_OPT_BUFFER_MSK (0x0001)`

Table buffer mask.

Definition at line 49 of file `cfe_tbl_api_typedefs.h`.

#### 10.45.2.2 CFE\_TBL\_OPT\_CRITICAL `#define CFE_TBL_OPT_CRITICAL (0x0008)`

Critical table.

Definition at line 64 of file `cfe_tbl_api_typedefs.h`.

**10.45.2.3 CFE\_TBL\_OPT\_CRITICAL\_MSK** #define CFE\_TBL\_OPT\_CRITICAL\_MSK (0x0008)

Table critical mask.

Definition at line 62 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.4 CFE\_TBL\_OPT\_DBL\_BUFFER** #define CFE\_TBL\_OPT\_DBL\_BUFFER (0x0001)

Double buffer table.

Definition at line 51 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.5 CFE\_TBL\_OPT\_DEFAULT** #define CFE\_TBL\_OPT\_DEFAULT (CFE\_TBL\_OPT\_SNGL\_BUFFER | CFE\_TBL\_OPT\_LOAD\_DUMP)

Default table options.

Definition at line 67 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.6 CFE\_TBL\_OPT\_DUMP\_ONLY** #define CFE\_TBL\_OPT\_DUMP\_ONLY (0x0002)

Dump only table.

Definition at line 55 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.7 CFE\_TBL\_OPT\_LD\_DMP\_MSK** #define CFE\_TBL\_OPT\_LD\_DMP\_MSK (0x0002)

Table load/dump mask.

Definition at line 53 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.8 CFE\_TBL\_OPT\_LOAD\_DUMP** #define CFE\_TBL\_OPT\_LOAD\_DUMP (0x0000)

Load/Dump table.

Definition at line 54 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.9 CFE\_TBL\_OPT\_NOT\_CRITICAL** #define CFE\_TBL\_OPT\_NOT\_CRITICAL (0x0000)

Not critical table.

Definition at line 63 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.10 CFE\_TBL\_OPT\_NOT\_USR\_DEF** #define CFE\_TBL\_OPT\_NOT\_USR\_DEF (0x0000)

Not user defined table.

Definition at line 58 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.11 CFE\_TBL\_OPT\_SNGL\_BUFFER** #define CFE\_TBL\_OPT\_SNGL\_BUFFER (0x0000)

Single buffer table.

Definition at line 50 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.12 CFE\_TBL\_OPT\_USR\_DEF\_ADDR** #define CFE\_TBL\_OPT\_USR\_DEF\_ADDR (0x0006)

User Defined table.,

**Note**

Automatically includes [CFE\\_TBL\\_OPT\\_DUMP\\_ONLY](#) option

Definition at line 60 of file cfe\_tbl\_api\_typedefs.h.

**10.45.2.13 CFE\_TBL\_OPT\_USR\_DEF\_MSK** #define CFE\_TBL\_OPT\_USR\_DEF\_MSK (0x0004)

Table user defined mask.

Definition at line 57 of file cfe\_tbl\_api\_typedefs.h.

## 10.46 cFE Get Current Time APIs

### Functions

- [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_GetTime \(void\)](#)  
*Get the current spacecraft time.*
- [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_GetTAI \(void\)](#)  
*Get the current TAI (MET + SCTF) time.*
- [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_GetUTC \(void\)](#)  
*Get the current UTC (MET + SCTF - Leap Seconds) time.*
- [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_GetMET \(void\)](#)  
*Get the current value of the Mission Elapsed Time (MET).*
- [uint32 CFE\\_TIME\\_GetMETseconds \(void\)](#)  
*Get the current seconds count of the mission-elapsed time.*
- [uint32 CFE\\_TIME\\_GetMETsubsecs \(void\)](#)  
*Get the current sub-seconds count of the mission-elapsed time.*

### 10.46.1 Detailed Description

### 10.46.2 Function Documentation

**10.46.2.1 CFE\_TIME\_GetMET()** [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_GetMET \(void \)](#)

Get the current value of the Mission Elapsed Time (MET).

#### Description

This routine returns the current mission-elapsed time (MET). MET is usually derived from a hardware-based clock that is not adjusted during normal operations. Callers of this routine should not assume that the MET return value has any specific relationship to any ground-based time standard.

#### Assumptions, External Events, and Notes:

None

#### Returns

The current MET

#### See also

[CFE\\_TIME\\_GetTime](#), [CFE\\_TIME\\_GetTAI](#), [CFE\\_TIME\\_GetUTC](#), [CFE\\_TIME\\_GetMETseconds](#), [CFE\\_TIME\\_GetMETsubsecs](#),  
[CFE\\_TIME\\_MET2SCTime](#)

**10.46.2.2 CFE\_TIME\_GetMETseconds()** `uint32 CFE_TIME_GetMETseconds (`  
    `void )`

Get the current seconds count of the mission-elapsed time.

#### Description

This routine is the same as [CFE\\_TIME\\_GetMET](#), except that it returns only the integer seconds portion of the MET time.

#### Assumptions, External Events, and Notes:

None

#### Returns

The current MET seconds

#### See also

[CFE\\_TIME\\_GetTime](#), [CFE\\_TIME\\_GetTAI](#), [CFE\\_TIME\\_GetUTC](#), [CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETsubsecs](#),  
[CFE\\_TIME\\_MET2SCTime](#)

**10.46.2.3 CFE\_TIME\_GetMETsubsecs()** `uint32 CFE_TIME_GetMETsubsecs (`  
    `void )`

Get the current sub-seconds count of the mission-elapsed time.

#### Description

This routine is the same as [CFE\\_TIME\\_GetMET](#), except that it returns only the integer sub-seconds portion of the MET time. Each count is equal to  $2^{-32}$  seconds.

#### Assumptions, External Events, and Notes:

None

#### Returns

The current MET sub-seconds

#### See also

[CFE\\_TIME\\_GetTime](#), [CFE\\_TIME\\_GetTAI](#), [CFE\\_TIME\\_GetUTC](#), [CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETseconds](#),  
[CFE\\_TIME\\_MET2SCTime](#)

**10.46.2.4 CFE\_TIME\_GetTAI()** `CFE_TIME_SysTime_t CFE_TIME_GetTAI (`  
    `void )`

Get the current TAI (MET + SCTF) time.

#### Description

This routine returns the current TAI time to the caller. TAI is an international time standard that does not include leap seconds. This routine should only be used in situations where TAI is absolutely required. Applications that call [CFE\\_TIME\\_GetTAI](#) may not be portable to all missions. Maintenance of correct TAI in flight is not guaranteed under all mission operations scenarios. To maintain re-usability across missions, most applications should be using [CFE\\_TIME\\_GetTime](#), rather than the specific routines for getting UTC/TAI directly.

**Assumptions, External Events, and Notes:**

1. The "TAI" time returned is referenced to the mission-defined time epoch, which may or may not be the same as the standard TAI epoch.
2. Even though TAI does not include leap seconds, the time returned by this function can still jump forward or backward without warning when the spacecraft clock is set or adjusted by operators. Applications using this function must be able to handle these time discontinuities gracefully.

**Returns**

The current spacecraft time in TAI

**See also**

[CFE\\_TIME\\_GetTime](#), [CFE\\_TIME\\_GetUTC](#), [CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETseconds](#), [CFE\\_TIME\\_GetMETsubsecs](#)

**10.46.2.5 CFE\_TIME\_GetTime()** `CFE_TIME_SysTime_t CFE_TIME_GetTime ( void )`

Get the current spacecraft time.

**Description**

This routine returns the current spacecraft time, which is the amount of time elapsed since the epoch as set in mission configuration. The time returned is either TAI (no leap seconds) or UTC (including leap seconds). This choice is made in the mission configuration file by defining either [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_TAI](#) or [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_UTC](#) as true at compile time. To maintain re-usability across missions, most applications should be using this function rather than the specific routines for getting UTC/TAI directly.

**Assumptions, External Events, and Notes:**

None

**Returns**

The current spacecraft time in default format

**See also**

[CFE\\_TIME\\_GetTAI](#), [CFE\\_TIME\\_GetUTC](#), [CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETseconds](#), [CFE\\_TIME\\_GetMETsubsecs](#)

Referenced by CF\_CFDP\_Send().

**10.46.2.6 CFE\_TIME\_GetUTC()** `CFE_TIME_SysTime_t CFE_TIME_GetUTC ( void )`

Get the current UTC (MET + SCTF - Leap Seconds) time.

**Description**

This routine returns the current UTC time to the caller. This routine should only be used in situations where UTC is absolutely required. Applications that call [CFE\\_TIME\\_GetUTC](#) may not be portable to all missions. Maintenance of correct UTC in flight is not guaranteed under all mission operations scenarios. If UTC is maintained in flight, it will jump backwards occasionally due to leap second adjustments. To maintain re-usability across missions, most applications should be using [CFE\\_TIME\\_GetTime](#), rather than the specific routines for getting UTC/TAI directly.

**Assumptions, External Events, and Notes:**

Note: The "UTC" time returned is referenced to the mission-defined time epoch, which may or may not be the same as the standard UTC epoch.

**Returns**

The current spacecraft time in UTC

**See also**

[CFE\\_TIME\\_GetTime](#), [CFE\\_TIME\\_GetTAI](#), [CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETseconds](#), [CFE\\_TIME\\_GetMETsubsecs](#)

## 10.47 cFE Get Time Information APIs

**Functions**

- [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_GetSTCF \(void\)](#)  
*Get the current value of the spacecraft time correction factor (STCF).*
- [int16 CFE\\_TIME\\_GetLeapSeconds \(void\)](#)  
*Get the current value of the leap seconds counter.*
- [CFE\\_TIME\\_ClockState\\_Enum\\_t CFE\\_TIME\\_GetClockState \(void\)](#)  
*Get the current state of the spacecraft clock.*
- [uint16 CFE\\_TIME\\_GetClockInfo \(void\)](#)  
*Provides information about the spacecraft clock.*

### 10.47.1 Detailed Description

### 10.47.2 Function Documentation

#### 10.47.2.1 CFE\_TIME\_GetClockInfo() `uint16 CFE_TIME_GetClockInfo ( void )`

Provides information about the spacecraft clock.

**Description**

This routine returns information on the spacecraft clock in a bit mask.

**Assumptions, External Events, and Notes:**

None

**Returns**

Spacecraft clock information, [cFE Clock State Flag Defines](#). To extract the information from the returned value, the flags can be used as in the following:

```
if ((ReturnValue & CFE_TIME_FLAG_xxxxxxx) == CFE_TIME_FLAG_xxxxxxx) then the following definition of the CFE_TIME_FLAG_xxxxxxx is true.
```

**See also**

[CFE\\_TIME\\_GetSTCF](#), [CFE\\_TIME\\_GetLeapSeconds](#), [CFE\\_TIME\\_GetClockState](#)

**10.47.2.2 CFE\_TIME\_GetClockState()** `CFE_TIME_ClockState_Enum_t CFE_TIME_GetClockState ( void )`

Get the current state of the spacecraft clock.

#### Description

This routine returns the spacecraft clock state. Applications that are highly dependent on valid time may want to call this routine before taking actions based on the times returned by the various clock routines

#### Assumptions, External Events, and Notes:

None

#### Returns

The current spacecraft clock state

#### See also

[CFE\\_TIME\\_GetSTCF](#), [CFE\\_TIME\\_GetLeapSeconds](#), [CFE\\_TIME\\_GetClockInfo](#)

**10.47.2.3 CFE\_TIME\_GetLeapSeconds()** `int16 CFE_TIME_GetLeapSeconds ( void )`

Get the current value of the leap seconds counter.

#### Description

This routine returns the current value of the leap seconds counter. This is the delta seconds between international atomic time (TAI) and universal coordinated time (UTC). There is no API provided to set or adjust leap seconds or SCTF, those actions should be done by command only. This API is provided for applications to be able to include leap seconds in their data products to aid in time correlation during downstream science data processing. Note that some mission operations teams do not maintain the leap seconds count, preferring to adjust the STCF instead. Users of this function should check with their mission ops team to see how they are planning to handle leap seconds.

#### Assumptions, External Events, and Notes:

None

#### Returns

The current spacecraft leap seconds.

#### See also

[CFE\\_TIME\\_GetSTCF](#), [CFE\\_TIME\\_GetClockState](#), [CFE\\_TIME\\_GetClockInfo](#)

**10.47.2.4 CFE\_TIME\_GetSTCF()** `CFE_TIME_SysTime_t CFE_TIME_GetSTCF ( void )`

Get the current value of the spacecraft time correction factor (STCF).

**Description**

This routine returns the current value of the spacecraft time correction factor. This is the delta time between the MET and the TAI time. There is no API provided to set or adjust leap seconds or SCTF, those actions should be done by command only. This API is provided for applications to be able to include STCF in their data products to aid in time correlation during downstream science data processing.

**Assumptions, External Events, and Notes:**

Does not include leap seconds

**Returns**

The current SCTF

**See also**

[CFE\\_TIME\\_GetLeapSeconds](#), [CFE\\_TIME\\_GetClockState](#), [CFE\\_TIME\\_GetClockInfo](#)

## 10.48 cFE Time Arithmetic APIs

**Functions**

- [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_Add \(CFE\\_TIME\\_SysTime\\_t Time1, CFE\\_TIME\\_SysTime\\_t Time2\)](#)  
*Adds two time values.*
- [CFE\\_TIME\\_SysTime\\_t CFE\\_TIME\\_Subtract \(CFE\\_TIME\\_SysTime\\_t Time1, CFE\\_TIME\\_SysTime\\_t Time2\)](#)  
*Subtracts two time values.*
- [CFE\\_TIME\\_Compare\\_t CFE\\_TIME\\_Compare \(CFE\\_TIME\\_SysTime\\_t TimeA, CFE\\_TIME\\_SysTime\\_t TimeB\)](#)  
*Compares two time values.*

### 10.48.1 Detailed Description

### 10.48.2 Function Documentation

#### 10.48.2.1 CFE\_TIME\_Add()

```
CFE_TIME_SysTime_t CFE_TIME_Add (
    CFE_TIME_SysTime_t Time1,
    CFE_TIME_SysTime_t Time2 )
```

Adds two time values.

**Description**

This routine adds the two specified times and returns the result. Normally, at least one of the input times should be a value representing a delta time. Adding two absolute times together will not cause an error, but the result will probably be meaningless.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>Time1</i>	The first time to be added.
in	<i>Time2</i>	The second time to be added.

### Returns

The sum of the two times. If the sum is greater than the maximum value that can be stored in a `CFE_TIME_SysTime_t`, the result will roll over (this is not considered an error).

### See also

[CFE\\_TIME\\_Subtract](#), [CFE\\_TIME\\_Compare](#)

**10.48.2.2 CFE\_TIME\_Compare()** `CFE_TIME_Compare_t CFE_TIME_Compare (`  
 `CFE_TIME_SysTime_t TimeA,`  
 `CFE_TIME_SysTime_t TimeB )`

Compares two time values.

### Description

This routine compares two time values to see which is "greater". It is important that applications use this function rather than trying to directly compare the component pieces of times. This function will handle roll-over cases seamlessly, which may not be intuitively obvious. The cFE's internal representation of time "rolls over" when the 32 bit seconds count reaches 0xFFFFFFFF. Also, subtracting a delta time from an absolute time close to the epoch could result in "roll under". The strange cases that result from these situations can be handled by defining the comparison function for times as follows: Plot the two times on the circumference of a circle where 0 is at the top and 0x80000000 is at the bottom. If the shortest arc from time A to time B runs clockwise around the circle, then time A is less than time B. If the shortest arc from A to B runs counter-clockwise, then time A is greater than time B.

### Assumptions, External Events, and Notes:

None

### Parameters

in	<i>TimeA</i>	The first time to compare.
in	<i>TimeB</i>	The second time to compare.

### Returns

The result of comparing the two times.

### Return values

<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.
<code>CFE_TIME_A_LT_B</code>	The first specified time is considered to be before the second specified time.

### See also

[CFE\\_TIME\\_Add](#), [CFE\\_TIME\\_Subtract](#)

**10.48.2.3 CFE\_TIME\_Subtract()** `CFE_TIME_SysTime_t CFE_TIME_Subtract (`  
    `CFE_TIME_SysTime_t Time1,`  
    `CFE_TIME_SysTime_t Time2 )`

Subtracts two time values.

#### Description

This routine subtracts time2 from time1 and returns the result. The time values can represent either absolute or delta times, but not all combinations make sense.

- AbsTime - AbsTime = DeltaTime
- AbsTime - DeltaTime = AbsTime
- DeltaTime - DeltaTime = DeltaTime
- DeltaTime - AbsTime = garbage

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>Time1</i>	The base time.
in	<i>Time2</i>	The time to be subtracted from the base time.

#### Returns

The result of subtracting the two times. If the subtraction results in an underflow, the result will roll over (this is not considered an error).

#### See also

[CFE\\_TIME\\_Add](#), [CFE\\_TIME\\_Compare](#)

## 10.49 cFE Time Conversion APIs

#### Functions

- [`CFE\_TIME\_SysTime\_t CFE\_TIME\_MET2SCTime \(CFE\_TIME\_SysTime\_t METTime\)`](#)  
*Convert specified MET into Spacecraft Time.*
- [`uint32 CFE\_TIME\_Sub2MicroSecs \(uint32 SubSeconds\)`](#)  
*Converts a sub-seconds count to an equivalent number of microseconds.*
- [`uint32 CFE\_TIME\_Micro2SubSecs \(uint32 MicroSeconds\)`](#)  
*Converts a number of microseconds to an equivalent sub-seconds count.*

#### 10.49.1 Detailed Description

#### 10.49.2 Function Documentation

**10.49.2.1 CFE\_TIME\_MET2SCTime()** `CFE_TIME_SysTime_t CFE_TIME_MET2SCTime (`  
    `CFE_TIME_SysTime_t METTime )`

Convert specified MET into Spacecraft Time.

**Description**

This function returns Spacecraft Time given MET. Note that Spacecraft Time is returned as either UTC or TAI depending on whether the mission configuration parameter [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_UTC](#) or [CFE\\_MISSION\\_TIME\\_CFG\\_DEFAULT\\_TAI](#) was set to true at compile time.

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>METTime</i>	The MET to be converted.
----	----------------	--------------------------

**Returns**

Spacecraft Time (UTC or TAI) corresponding to the specified MET

**See also**

[CFE\\_TIME\\_GetMET](#), [CFE\\_TIME\\_GetMETseconds](#), [CFE\\_TIME\\_GetMETsubsecs](#), [CFE\\_TIME\\_Sub2MicroSecs](#), [CFE\\_TIME\\_Micro2SubSecs](#)

**10.49.2.2 CFE\_TIME\_Micro2SubSecs()** `uint32 CFE_TIME_Micro2SubSecs (`  
`uint32 MicroSeconds )`

Converts a number of microseconds to an equivalent sub-seconds count.

**Description**

This routine converts from microseconds (each tick is 1e-06 seconds) to a subseconds count (each tick is 1 / 2^32 seconds).

**Assumptions, External Events, and Notes:**

None

**Parameters**

in	<i>MicroSeconds</i>	The sub-seconds count to convert.
----	---------------------	-----------------------------------

**Returns**

The equivalent number of subseconds. If the number of microseconds passed in is greater than one second, (i.e. > 999,999), the return value is equal to 0xffffffff.

**See also**

[CFE\\_TIME\\_MET2SCTime](#), [CFE\\_TIME\\_Sub2MicroSecs](#),

**10.49.2.3 CFE\_TIME\_Sub2MicroSecs()** `uint32 CFE_TIME_Sub2MicroSecs (`  
`uint32 SubSeconds )`

Converts a sub-seconds count to an equivalent number of microseconds.

#### Description

This routine converts from a sub-seconds count (each tick is  $1 / 2^{32}$  seconds) to microseconds (each tick is  $1e-06$  seconds).

#### Assumptions, External Events, and Notes:

None

#### Parameters

in	<i>SubSeconds</i>	The sub-seconds count to convert.
----	-------------------	-----------------------------------

#### Returns

The equivalent number of microseconds.

#### See also

[CFE\\_TIME\\_MET2SCTime](#), [CFE\\_TIME\\_Micro2SubSecs](#),

## 10.50 cFE External Time Source APIs

### Functions

- void [CFE\\_TIME\\_ExternalTone](#) (void)  
*Provides the 1 Hz signal from an external source.*
- void [CFE\\_TIME\\_ExternalMET](#) ([CFE\\_TIME\\_SysTime\\_t](#) NewMET)  
*Provides the Mission Elapsed Time from an external source.*
- void [CFE\\_TIME\\_ExternalGPS](#) ([CFE\\_TIME\\_SysTime\\_t](#) NewTime, [int16](#) NewLeaps)  
*Provide the time from an external source that has data common to GPS receivers.*
- void [CFE\\_TIME\\_ExternalTime](#) ([CFE\\_TIME\\_SysTime\\_t](#) NewTime)  
*Provide the time from an external source that measures time relative to a known epoch.*
- [CFE\\_Status\\_t CFE\\_TIME\\_RegisterSyncCallback](#) ([CFE\\_TIME\\_SynchCallbackPtr\\_t](#) CallbackFuncPtr)  
*Registers a callback function that is called whenever time synchronization occurs.*
- [CFE\\_Status\\_t CFE\\_TIME\\_UnregisterSyncCallback](#) ([CFE\\_TIME\\_SynchCallbackPtr\\_t](#) CallbackFuncPtr)  
*Unregisters a callback function that is called whenever time synchronization occurs.*

### 10.50.1 Detailed Description

### 10.50.2 Function Documentation

#### 10.50.2.1 [CFE\\_TIME\\_ExternalGPS\(\)](#) void CFE\_TIME\_ExternalGPS (

```
    CFE\_TIME\_SysTime\_t NewTime,
    int16 NewLeaps )
```

Provide the time from an external source that has data common to GPS receivers.

### Description

This routine provides a method to provide cFE TIME with current time data acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration parameter specified window for tone signal and data packet verification.

Internally, cFE TIME will calculate a new STCF as the difference between this new time value and the spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

### Assumptions, External Events, and Notes:

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_GPS](#), which indicates that the external time data consists of a time value relative to a known epoch, plus a leap seconds value.

### Parameters

in	<i>NewTime</i>	The MET value at the next (or previous) 1 Hz tone signal.
in	<i>NewLeaps</i>	The Leap Seconds value used to calculate time as UTC.

### See also

[CFE\\_TIME\\_ExternalTone](#), [CFE\\_TIME\\_ExternalMET](#), [CFE\\_TIME\\_ExternalTime](#)

### 10.50.2.2 CFE\_TIME\_ExternalMET()

```
void CFE_TIME_ExternalMET (
    CFE_TIME_SysTime_t NewMET )
```

Provides the Mission Elapsed Time from an external source.

### Description

This routine provides a method to provide cFE TIME with MET acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration parameter specified window for tone signal and data packet verification.

The MET value at the tone "should" have zero subseconds. Although the interface accepts non-zero values for sub-seconds, it may be harmful to other applications that expect zero subseconds at the moment of the tone. Any decision to use non-zero subseconds should be carefully considered.

**Assumptions, External Events, and Notes:**

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_MET](#), which indicates that the external time data consists of MET.

**Parameters**

in	<a href="#">NewMET</a>	The MET value at the next (or previous) 1 Hz tone signal.
----	------------------------	---

**See also**

[CFE\\_TIME\\_ExternalTone](#), [CFE\\_TIME\\_ExternalGPS](#), [CFE\\_TIME\\_ExternalTime](#)

**10.50.2.3 CFE\_TIME\_ExternalTime()** `void CFE_TIME_ExternalTime (`  
`CFE\_TIME\_SysTime\_t NewTime )`

Provide the time from an external source that measures time relative to a known epoch.

**Description**

This routine provides a method to provide cFE TIME with current time data acquired from an external source. There is a presumption that this function will be called at the appropriate time (relative to the tone) such that this call may be used by cFE TIME as the signal to generate the "time at the tone" data command. The "time at the tone" data command must arrive within the configuration specified window for tone signal and data packet verification.

Internally, cFE TIME will calculate a new STCF as the difference between this new time value and the spacecraft MET value at the tone. This allows cFE TIME to always calculate time as the sum of MET and STCF. The value of STCF will change only as much as the drift factor between spacecraft MET and the external time source.

**Assumptions, External Events, and Notes:**

- This routine is included in the API only when 3 specific configuration parameters are set to true. The first is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) which defines this instantiation of cFE TIME as a time server (not a client). The second required configuration parameter is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) which enables time source selection commands to the cFE TIME task, and further enables configuration definitions for the selected type of external time data. The third configuration parameter required for this routine is [CFE\\_PLATFORM\\_TIME\\_CFG\\_SRC\\_TIME](#), which indicates that the external time data consists of a time value relative to a known epoch.

**Parameters**

in	<a href="#">NewTime</a>	The MET value at the next (or previous) 1 Hz tone signal.
----	-------------------------	---

**See also**

[CFE\\_TIME\\_ExternalTone](#), [CFE\\_TIME\\_ExternalMET](#), [CFE\\_TIME\\_ExternalGPS](#)

```
10.50.2.4 CFE_TIME_ExternalTone() void CFE_TIME_ExternalTone (
    void )
```

Provides the 1 Hz signal from an external source.

#### Description

This routine provides a method for cFE TIME software to be notified of the occurrence of the 1Hz tone signal without knowledge of the specific hardware design. Regardless of the source of the tone, this routine should be called as soon as possible after detection to allow cFE TIME software the opportunity to latch the local clock as close as possible to the instant of the tone.

#### Assumptions, External Events, and Notes:

- This routine may be called directly from within the context of an interrupt handler.

#### See also

[CFE\\_TIME\\_ExternalMET](#), [CFE\\_TIME\\_ExternalGPS](#), [CFE\\_TIME\\_ExternalTime](#)

```
10.50.2.5 CFE_TIME_RegisterSynchCallback() CFE_Status_t CFE_TIME_RegisterSynchCallback (
    CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr )
```

Registers a callback function that is called whenever time synchronization occurs.

#### Description

This routine passes a callback function pointer for an Application that wishes to be notified whenever a legitimate time synchronization signal (typically a 1 Hz) is received.

#### Assumptions, External Events, and Notes:

Only a single callback per application is supported, and this function should only be called from a single thread within each application (typically the apps main thread). If an application requires triggering multiple child tasks at 1Hz, it should distribute the timing signal internally, rather than registering for multiple callbacks.

#### Parameters

in	<i>CallbackFuncPtr</i>	Function to call at synchronization interval (must not be null)
----	------------------------	---

#### Returns

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

#### Return values

<a href="#">CFE_SUCCESS</a>	Successful execution.
<a href="#">CFE_TIME_TOO_MANY_SYNCH_CALLBACKS</a>	Too Many Sync Callbacks.
<a href="#">CFE_TIME_BAD_ARGUMENT</a>	Bad Argument.

See also

[CFE\\_TIME\\_UnregisterSynchCallback](#)

**10.50.2.6 CFE\_TIME\_UnregisterSynchCallback()** `CFE_Status_t CFE_TIME_UnregisterSynchCallback ( CFE_TIME_SynchCallbackPtr_t CallbackFuncPtr )`

Unregisters a callback function that is called whenever time synchronization occurs.

#### Description

This routine removes the specified callback function pointer from the list of Callback functions that are called whenever a time synchronization (typically the 1Hz signal) is received.

#### Assumptions, External Events, and Notes:

Only a single callback per application is supported, and this function should only be called from a single thread within each application (typically the apps main thread).

#### Parameters

in	<code>CallbackFuncPtr</code>	Function to remove from synchronization call list (must not be null)
----	------------------------------	--

#### Returns

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

#### Return values

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_TIME_CALLBACK_NOT_REGISTERED</code>	Callback Not Registered.
<code>CFE_TIME_BAD_ARGUMENT</code>	Bad Argument.

#### See also

[CFE\\_TIME\\_RegisterSynchCallback](#)

## 10.51 cFE Miscellaneous Time APIs

#### Functions

- `CFE_Status_t CFE_TIME_Print (char *PrintBuffer, CFE_TIME_SysTime_t TimeToPrint)`  
*Print a time value as a string.*
- `void CFE_TIME_Local1HzISR (void)`

*This function is called via a timer callback set up at initialization of the TIME service.*

### 10.51.1 Detailed Description

### 10.51.2 Function Documentation

---

```
10.51.2.1 CFE_TIME_Local1HzISR() void CFE_TIME_Local1HzISR (
    void )
```

This function is called via a timer callback set up at initialization of the TIME service.

#### Description

Drives the time processing logic from the system PSP layer. This must be called once per second based on a hardware interrupt or OS kernel signal.

#### Assumptions, External Events, and Notes:

This will update the global data structures accordingly, incrementing each by the 1Hz amount.

---

```
10.51.2.2 CFE_TIME_Print() CFE_Status_t 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.

**Returns**

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

**Return values**

<code>CFE_SUCCESS</code>	Successful execution.
<code>CFE_TIME_BAD_ARGUMENT</code>	Bad Argument.

## 10.52 cFE Resource ID base values

**Enumerations**

- enum {
   
`CFE_RESOURCEID_ES_TASKID_BASE_OFFSET` = `OS_OBJECT_TYPE_OS_TASK` , `CFE_RESOURCEID_ES_APPID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 1` , `CFE_RESOURCEID_ES_LIBID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 2` , `CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 3` ,
 `CFE_RESOURCEID_ES_POOLID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 4` , `CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 5` , `CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 6` ,
 `CFE_RESOURCEID_CONFIGID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 7` ,
 `CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 8` , `CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 9` ,
 `CFE_RESOURCEID_TBL_LOADBUFFID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 10` , `CFE_RESOURCEID_TBL_ACCESSID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 11` ,
 `CFE_RESOURCEID_TBL_REGID_BASE_OFFSET` = `OS_OBJECT_TYPE_USER + 12` }
- enum {
   
`CFE_ES_TASKID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_TASKID_BASE_OFFSET)` , `CFE_ES_APPID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_APPID_BASE_OFFSET)` , `CFE_ES_LIBID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_LIBID_BASE_OFFSET)` , `CFE_ES_COUNTID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET)` ,
 `CFE_ES_POOLID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_POOLID_BASE_OFFSET)` , `CFE_ES_CDSBLOCKID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET)` , `CFE_SB_PIPEID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET)` , `CFE_CONFIGID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_CONFIGID_BASE_OFFSET)` ,
 `CFE_TBL_VALRESULTID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET)` , `CFE_TBL_DUMPCTRLID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET)` , `CFE_TBL_LOADBUFFID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_LOADBUFFID_BASE_OFFSET)` , `CFE_TBL_HANDLE_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_ACCESSID_BASE_OFFSET)` ,
 `CFE_TBL_REGID_BASE` = `CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_REGID_BASE_OFFSET)` }

### 10.52.1 Detailed Description

### 10.52.2 Enumeration Type Documentation

#### 10.52.2.1 anonymous enum anonymous enum

**Enumerator**

CFE_RESOURCEID_ES_TASKID_BASE_OFFSET	
CFE_RESOURCEID_ES_APPID_BASE_OFFSET	
CFE_RESOURCEID_ES_LIBID_BASE_OFFSET	
CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET	
CFE_RESOURCEID_ES_POOLID_BASE_OFFSET	
CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET	
CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET	
CFE_RESOURCEID_CONFIGID_BASE_OFFSET	
CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET	
CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET	
CFE_RESOURCEID_TBL_LOADBUFFID_BASE_OFFSET	
CFE_RESOURCEID_TBL_ACCESSID_BASE_OFFSET	
CFE_RESOURCEID_TBL_REGID_BASE_OFFSET	

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

**10.52.2.2 anonymous enum anonymous enum****Enumerator**

CFE_ES_TASKID_BASE	
CFE_ES_APPID_BASE	
CFE_ES_LIBID_BASE	
CFE_ES_COUNTID_BASE	
CFE_ES_POOLID_BASE	
CFE_ES_CDSBLOCKID_BASE	
CFE_SB_PIPEID_BASE	
CFE_CONFIGID_BASE	
CFE_TBL_VALRESULTID_BASE	
CFE_TBL_DUMPCTRLID_BASE	
CFE_TBL_LOADBUFFID_BASE	
CFE_TBL_HANDLE_BASE	
CFE_TBL_REGID_BASE	

Definition at line 87 of file `cfe_core_resourceid_basevalues.h`.

## 10.53 cFE Clock State Flag Defines

**Macros**

- #define `CFE_TIME_FLAG_CLKSET` 0x8000

*The spacecraft time has been set.*

- #define `CFE_TIME_FLAG_FLYING` 0x4000

*This instance of Time Services is flywheeling.*

- #define `CFE_TIME_FLAG_SRCINT` 0x2000

*The clock source is set to "internal".*

- #define CFE\_TIME\_FLAG\_SIGPRI 0x1000  
*The clock signal is set to "primary".*
- #define CFE\_TIME\_FLAG\_SRVFLY 0x0800  
*The Time Server is in flywheel mode.*
- #define CFE\_TIME\_FLAG\_CMDFLY 0x0400  
*This instance of Time Services was commanded into flywheel mode.*
- #define CFE\_TIME\_FLAG\_ADDADJ 0x0200  
*One time STCF Adjustment is to be done in positive direction.*
- #define CFE\_TIME\_FLAG\_ADD1HZ 0x0100  
*1 Hz STCF Adjustment is to be done in a positive direction*
- #define CFE\_TIME\_FLAG\_ADDTCL 0x0080  
*Time Client Latency is applied in a positive direction.*
- #define CFE\_TIME\_FLAG\_SERVER 0x0040  
*This instance of Time Services is a Time Server.*
- #define CFE\_TIME\_FLAG\_GDTONE 0x0020  
*The tone received is good compared to the last tone received.*
- #define CFE\_TIME\_FLAG\_REFERR 0x0010  
*GetReference read error, will be set if unable to get a consistent ref value.*
- #define CFE\_TIME\_FLAG\_UNUSED 0x000F  
*Reserved flags - should be zero.*

### 10.53.1 Detailed Description

### 10.53.2 Macro Definition Documentation

**10.53.2.1 CFE\_TIME\_FLAG\_ADD1HZ** #define CFE\_TIME\_FLAG\_ADD1HZ 0x0100  
1 Hz STCF Adjustment is to be done in a positive direction  
Definition at line 42 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.2 CFE\_TIME\_FLAG\_ADDADJ** #define CFE\_TIME\_FLAG\_ADDADJ 0x0200  
One time STCF Adjustment is to be done in positive direction.  
Definition at line 41 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.3 CFE\_TIME\_FLAG\_ADDTCL** #define CFE\_TIME\_FLAG\_ADDTCL 0x0080  
Time Client Latency is applied in a positive direction.  
Definition at line 43 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.4 CFE\_TIME\_FLAG\_CLKSET** #define CFE\_TIME\_FLAG\_CLKSET 0x8000  
The spacecraft time has been set.  
Definition at line 35 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.5 CFE\_TIME\_FLAG\_CMDFLY** #define CFE\_TIME\_FLAG\_CMDFLY 0x0400  
This instance of Time Services was commanded into flywheel mode.  
Definition at line 40 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.6 CFE\_TIME\_FLAG\_FLYING** #define CFE\_TIME\_FLAG\_FLYING 0x4000

This instance of Time Services is flywheeling.

Definition at line 36 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.7 CFE\_TIME\_FLAG\_GDTONE** #define CFE\_TIME\_FLAG\_GDTONE 0x0020

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

Definition at line 45 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.8 CFE\_TIME\_FLAG\_REFERR** #define CFE\_TIME\_FLAG\_REFERR 0x0010

GetReference read error, will be set if unable to get a consistent ref value.

Definition at line 47 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.9 CFE\_TIME\_FLAG\_SERVER** #define CFE\_TIME\_FLAG\_SERVER 0x0040

This instance of Time Services is a Time Server.

Definition at line 44 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.10 CFE\_TIME\_FLAG\_SIGPRI** #define CFE\_TIME\_FLAG\_SIGPRI 0x1000

The clock signal is set to "primary".

Definition at line 38 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.11 CFE\_TIME\_FLAG\_SRCINT** #define CFE\_TIME\_FLAG\_SRCINT 0x2000

The clock source is set to "internal".

Definition at line 37 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.12 CFE\_TIME\_FLAG\_SRVFLY** #define CFE\_TIME\_FLAG\_SRVFLY 0x0800

The Time Server is in flywheel mode.

Definition at line 39 of file default\_cfe\_time\_msgdefs.h.

**10.53.2.13 CFE\_TIME\_FLAG\_UNUSED** #define CFE\_TIME\_FLAG\_UNUSED 0x000F

Reserved flags - should be zero.

Definition at line 48 of file default\_cfe\_time\_msgdefs.h.

## 10.54 OSAL Semaphore State Defines

### Macros

- #define OS\_SEM\_FULL 1  
*Semaphore full state.*
- #define OS\_SEM\_EMPTY 0  
*Semaphore empty state.*

### 10.54.1 Detailed Description

### 10.54.2 Macro Definition Documentation

**10.54.2.1 OS\_SEM\_EMPTY** #define OS\_SEM\_EMPTY 0  
Semaphore empty state.  
Definition at line 35 of file osapi-binsem.h.

**10.54.2.2 OS\_SEM\_FULL** #define OS\_SEM\_FULL 1  
Semaphore full state.  
Definition at line 34 of file osapi-binsem.h.

## 10.55 OSAL Binary Semaphore APIs

### Functions

- int32 **OS\_BinSemCreate** (osal\_id\_t \*sem\_id, const char \*sem\_name, uint32 sem\_initial\_value, uint32 options)  
*Creates a binary semaphore.*
- int32 **OS\_BinSemFlush** (osal\_id\_t sem\_id)  
*Unblock all tasks pending on the specified semaphore.*
- int32 **OS\_BinSemGive** (osal\_id\_t sem\_id)  
*Increment the semaphore value.*
- int32 **OS\_BinSemTake** (osal\_id\_t sem\_id)  
*Decrement the semaphore value.*
- int32 **OS\_BinSemTimedWait** (osal\_id\_t sem\_id, uint32 msecs)  
*Decrement the semaphore value with a timeout.*
- int32 **OS\_BinSemDelete** (osal\_id\_t sem\_id)  
*Deletes the specified Binary Semaphore.*
- int32 **OS\_BinSemGetIdByName** (osal\_id\_t \*sem\_id, const char \*sem\_name)  
*Find an existing semaphore ID by name.*
- int32 **OS\_BinSemGetInfo** (osal\_id\_t sem\_id, OS\_bin\_sem\_prop\_t \*bin\_prop)  
*Fill a property object buffer with details regarding the resource.*

### 10.55.1 Detailed Description

### 10.55.2 Function Documentation

**10.55.2.1 OS\_BinSemCreate()** int32 OS\_BinSemCreate (osal\_id\_t \* sem\_id, const char \* sem\_name, uint32 sem\_initial\_value, uint32 options )

Creates a binary semaphore.

Creates a binary semaphore with initial value specified by `sem_initial_value` and name specified by `sem_name`. `sem_id` will be returned to the caller

#### Parameters

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 binary 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 sen name or sem_id are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NO_FREE_IDS</i>	if all of the semaphore ids are taken
<i>OS_ERR_NAME_TAKEN</i>	if this is already the name of a binary semaphore
<i>OS_SEM_FAILURE</i>	if the OS call failed (return value only verified in coverage test)

**10.55.2.2 OS\_BinSemDelete() `int32 OS_BinSemDelete(`**

`osal_id_t sem_id )`

Deletes the specified Binary Semaphore.

This is the function used to delete a binary semaphore in the operating system. This also frees the respective sem\_id to be used again when another semaphore is created.

**Parameters**

in	<i>sem_id</i>	The object ID to delete
----	---------------	-------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid binary semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified failure occurs (return value only verified in coverage test)

**10.55.2.3 OS\_BinSemFlush() `int32 OS_BinSemFlush(`**

`osal_id_t sem_id )`

Unblock all tasks pending on the specified semaphore.

The function unblocks all tasks pending on the specified semaphore. However, this function does not change the state of the semaphore.

**Parameters**

in	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a binary semaphore
<code>OS_SEM_FAILURE</code>	if an unspecified failure occurs (return value only verified in coverage test)

**10.55.2.4 OS\_BinSemGetIdByName()** `int32 OS_BinSemGetIdByName( osal_id_t * sem_id, const char * sem_name )`

Find an existing semaphore ID by name.

This function tries to find a binary sem Id given the name of a bin\_sem. The id is returned through sem\_id

**Parameters**

<code>out</code>	<code>sem_id</code>	will be set to the ID of the existing resource
<code>in</code>	<code>sem_name</code>	the name of the existing resource to find (must not be null)

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	is semid or sem_name are NULL pointers
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name was not found in the table

**10.55.2.5 OS\_BinSemGetInfo()** `int32 OS_BinSemGetInfo( osal_id_t sem_id, OS_bin_sem_prop_t * bin_prop )`

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info( name and creator) about the specified binary semaphore.

**Parameters**

<code>in</code>	<code>sem_id</code>	The object ID to operate on
<code>out</code>	<code>bin_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 in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the bin_prop pointer is null
<i>OS_ERR_NOT_IMPLEMENTED</i>	Not implemented.

**10.55.2.6 OS\_BinSemGive()** `int32 OS_BinSemGive ( osal_id_t sem_id )`

Increment the semaphore value.

The function unlocks the semaphore referenced by `sem_id` by performing a semaphore unlock operation on that semaphore. If the semaphore value resulting from this operation is positive, then no threads were blocked waiting for the semaphore to become unlocked; the semaphore value is simply incremented for this semaphore.

**Parameters**

in	<i>sem-&gt;_id</i>	The object ID to operate on
----	--------------------	-----------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a binary semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified failure occurs (return value only verified in coverage test)

**10.55.2.7 OS\_BinSemTake()** `int32 OS_BinSemTake ( osal_id_t sem_id )`

Decrement the semaphore value.

The locks the semaphore referenced by `sem_id` by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then the calling thread shall not return from the call until it either locks the semaphore or the call is interrupted.

**Parameters**

in	<i>sem-&gt;_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>	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)

```
10.55.2.8 OS_BinSemTimedWait() int32 OS_BinSemTimedWait (
    osal_id_t sem_id,
    uint32 msecs )
```

Decrement the semaphore value with a timeout.

The function locks the semaphore referenced by `sem_id`. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore, this wait shall be terminated when the specified timeout, `msecs`, expires.

**Parameters**

in	<code>sem_id</code>	The object ID to operate on
in	<code>msecs</code>	The maximum amount of time to block, in milliseconds

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_SEM_TIMEOUT</code>	if semaphore was not relinquished in time
<code>OS_ERR_INVALID_ID</code>	if the ID passed in is not a valid semaphore ID
<code>OS_SEM_FAILURE</code>	if an unspecified failure occurs (return value only verified in coverage test)

## 10.56 OSAL BSP low level access APIs

**Functions**

- void `OS_BSP_SetResourceTypeConfig (uint32 ResourceType, uint32 ConfigOptionValue)`
- `uint32 OS_BSP_GetResourceTypeConfig (uint32 ResourceType)`
- `uint32 OS_BSP_GetArgC (void)`
- `char *const * OS_BSP_GetArgV (void)`
- void `OS_BSP_SetExitCode (int32 code)`

### 10.56.1 Detailed Description

These are for OSAL internal BSP information access to pass any BSP-specific boot/command line/startup arguments through to the application, and return a status code back to the OS after exit.

Not intended for user application use

### 10.56.2 Function Documentation

**10.56.2.1 OS\_BSP\_GetArgC()** `uint32 OS_BSP_GetArgC ( void )`

**10.56.2.2 OS\_BSP\_GetArgV()** `char* const* OS_BSP_GetArgV ( void )`

**10.56.2.3 OS\_BSP\_GetResourceTypeConfig()** `uint32 OS_BSP_GetResourceTypeConfig ( uint32 ResourceType )`

**10.56.2.4 OS\_BSP\_SetExitCode()** `void OS_BSP_SetExitCode ( int32 code )`

**10.56.2.5 OS\_BSP\_SetResourceTypeConfig()** `void OS_BSP_SetResourceTypeConfig ( uint32 ResourceType, uint32 ConfigOptionValue )`

## 10.57 OSAL Real Time Clock APIs

### Functions

- **`int32 OS_GetMonotonicTime (OS_time_t *time_struct)`**  
*Get the monotonic time.*
- **`int32 OS_GetLocalTime (OS_time_t *time_struct)`**  
*Get the local time.*
- **`int32 OS_SetLocalTime (const OS_time_t *time_struct)`**  
*Set the local time.*
- **`OS_time_t OS_TimeFromRelativeMilliseconds (int32 relative_msec)`**  
*Gets an absolute time value relative to the current time.*
- **`int32 OS_TimeToRelativeMilliseconds (OS_time_t time)`**  
*Gets a relative time value from an absolute time.*
- **`static int64 OS_TimeGetTotalSeconds (OS_time_t tm)`**  
*Get interval from an `OS_time_t` object normalized to whole number of seconds.*
- **`static OS_time_t OS_TimeFromTotalSeconds (int64 tm)`**  
*Get an `OS_time_t` interval object from an integer number of seconds.*
- **`static int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`**  
*Get interval from an `OS_time_t` object normalized to millisecond units.*
- **`static OS_time_t OS_TimeFromTotalMilliseconds (int64 tm)`**  
*Get an `OS_time_t` interval object from a integer number of milliseconds.*
- **`static int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`**  
*Get interval from an `OS_time_t` object normalized to microsecond units.*

- static `OS_time_t OS_TimeFromTotalMicroseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of microseconds.*
- static `int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to nanosecond units.*
- static `OS_time_t OS_TimeFromTotalNanoseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of nanoseconds.*
- static `int64 OS_TimeGetFractionalPart (OS_time_t tm)`  
*Get subseconds portion (fractional part only) from an `OS_time_t` object.*
- static `uint32 OS_TimeGetSubsecondsPart (OS_time_t tm)`  
*Get 32-bit normalized subseconds (fractional part only) from an `OS_time_t` object.*
- static `uint32 OS_TimeGetMillisecondsPart (OS_time_t tm)`  
*Get milliseconds portion (fractional part only) from an `OS_time_t` object.*
- static `uint32 OS_TimeGetMicrosecondsPart (OS_time_t tm)`  
*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.*
- static `bool OS_TimeEqual (OS_time_t time1, OS_time_t time2)`  
*Checks if two time values are equal.*
- static `int8_t OS_TimeGetSign (OS_time_t time)`  
*Checks the sign of the time value.*
- static `int8_t OS_TimeCompare (OS_time_t time1, OS_time_t time2)`  
*Compares two time values.*

### 10.57.1 Detailed Description

### 10.57.2 Function Documentation

#### 10.57.2.1 `OS_GetLocalTime()` `int32 OS_GetLocalTime (` `OS_time_t * time_struct )`

Get the local time.

This function gets the local time from the underlying OS.

##### Note

Mission time management typically uses the cFE Time Service

**Parameters**

<code>out</code>	<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 <code>time_struct</code> is null

**10.57.2.2 OS\_GetMonotonicTime()** `int32 OS_GetMonotonicTime ( OS_time_t * time_struct )`

Get the monotonic time.

This function gets the monotonic time from the underlying OS.

Monotonic time differs from local or real (wall-clock) time in that it cannot be set. As a result, the time is always guaranteed to be increasing, and never moves backwards or has discontinuities (with some exceptions - see POSIX documentation).

The Epoch is undefined- In many implementations, the epoch is the system boot time, and the clock increases indefinitely so long as the system remains powered on.

**Note**

Sometimes a monotonic clock is implemented in the operating system and sometimes it is implemented in platform-specific hardware.

If the operating system does not provide a monotonic clock, then this function may return [OS\\_ERR\\_NOT\\_IMPLEMENTED](#). However, a platform-specific monotonic clock may still exist, in the form of a CPU register or external oscillator. This API only accesses a monotonic clock if one is provided by the operating system.  
To read a platform-specific monotonic clock in CFE, see [CFE\\_PSP\\_GetTime\(\)](#).

**Parameters**

<code>out</code>	<code>time_struct</code>	An <a href="#">OS_time_t</a> that will be set to the monotonic time (must not be null)
------------------	--------------------------	--

**Returns**

Get monotonic time status, see [OSAL Return Code Defines](#)

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if <code>time_struct</code> is null
<a href="#">OS_ERR_NOT_IMPLEMENTED</a>	if operating system does not implement a monotonic clock

**10.57.2.3 OS\_SetLocalTime()** `int32 OS_SetLocalTime (`

```
const OS_time_t * time_struct )
```

Set the local time.

This function sets the local time on the underlying OS.

#### Note

Mission time management typically uses the cFE Time Services

#### Parameters

in	time_struct	An <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

**10.57.2.4 OS\_TimeAdd()** static OS\_time\_t OS\_TimeAdd (

```
    OS_time_t time1,
    OS_time_t time2 ) [inline], [static]
```

Computes the sum of two time intervals.

#### Parameters

in	time1	The first interval
in	time2	The second interval

#### Returns

The sum of the two intervals (time1 + time2)

Definition at line 562 of file osapi-clock.h.

References OS\_time\_t::ticks.

**10.57.2.5 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 497 of file osapi-clock.h.

References [OS\\_TIME\\_TICKS\\_PER\\_SECOND](#), [OS\\_TIME\\_TICKS\\_PER\\_USEC](#), and [OS\\_time\\_t::ticks](#).

**10.57.2.6 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 521 of file osapi-clock.h.

References [OS\\_TIME\\_TICKS\\_PER\\_MSEC](#), [OS\\_TIME\\_TICKS\\_PER\\_SECOND](#), and [OS\\_time\\_t::ticks](#).

**10.57.2.7 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 473 of file osapi-clock.h.

References OS\_TIME\_TICK\_RESOLUTION\_NS, OS\_TIME\_TICKS\_PER\_SECOND, and OS\_time\_t::ticks.

**10.57.2.8 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 544 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, and OS\_time\_t::ticks.

**10.57.2.9 OS\_TimeCompare()** static int8\_t OS\_TimeCompare (

```
    OS\_time\_t time1,
    OS\_time\_t time2 ) [inline], [static]
```

Compares two time values.

**Parameters**

in	<i>time1</i>	The first time
in	<i>time2</i>	The second time

**Return values**

-1	if the time1 < time2
----	----------------------

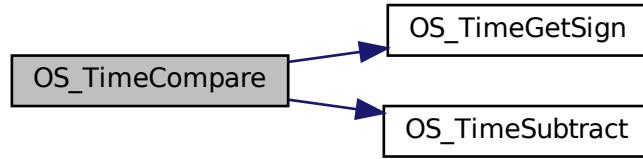
#### Return values

<i>0</i>	if the times are equal
<i>1</i>	if the time1 > time2

Definition at line 624 of file osapi-clock.h.

References OS\_TimeGetSign(), and OS\_TimeSubtract().

Here is the call graph for this function:



**10.57.2.10 OS\_TimeEqual()** static bool OS\_TimeEqual (   
*OS\_time\_t* time1,   
*OS\_time\_t* time2 ) [inline], [static]

Checks if two time values are equal.

#### Parameters

<i>in</i>	<i>time1</i>	The first time value
<i>in</i>	<i>time2</i>	The second time value

#### Return values

<i>true</i>	if the two values are equal
<i>false</i>	if the two values are not equal

Definition at line 593 of file osapi-clock.h.

References OS\_time\_t::ticks.

**10.57.2.11 OS\_TimeFromRelativeMilliseconds()** *OS\_time\_t* OS\_TimeFromRelativeMilliseconds (   
*int32* relative\_msec )

Gets an absolute time value relative to the current time.

This function adds the given interval, expressed in milliseconds, to the current clock and returns the result.

#### Note

This is intended to ease transitioning from a relative timeout value to an absolute timeout value. The result can be passed to any function that accepts an absolute timeout, to mimic the behavior of a relative timeout.

**Parameters**

in	<i>relative_msec</i>	A relative time interval, in milliseconds
----	----------------------	---

**Returns**

Absolute time value after adding interval

**10.57.2.12 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 310 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_USEC.

**10.57.2.13 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 276 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_MSEC.

**10.57.2.14 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	<code>tm</code>	Time interval value, in nanoseconds
----	-----------------	-------------------------------------

Returns

`OS_time_t` value representing the interval

Definition at line 349 of file osapi-clock.h.

References OS\_TIME\_TICK\_RESOLUTION\_NS.

**10.57.2.15 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	<code>tm</code>	Time interval value, in seconds
----	-----------------	---------------------------------

Returns

`OS_time_t` value representing the interval

Definition at line 242 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND.

**10.57.2.16 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 365 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, and OS\_time\_t::ticks.

Referenced by OS\_TimeGetMicrosecondsPart(), OS\_TimeGetMillisecondsPart(), OS\_TimeGetNanosecondsPart(), and OS\_TimeGetSubsecondsPart().

**10.57.2.17 OS\_TimeGetMicrosecondsPart()** static uint32 OS\_TimeGetMicrosecondsPart (

    OS\_time\_t tm ) [inline], [static]

Get microseconds portion (fractional part only) from an [OS\\_time\\_t](#) object.

Extracts the fractional part from a given [OS\\_time\\_t](#) object normalized to units of microseconds.

This function may be used to adapt applications initially implemented using an older OSAL version where [OS\\_time\\_t](#) was a structure containing a "seconds" and "microsecs" field.

This function will obtain a value that is compatible with the "microsecs" field of [OS\\_time\\_t](#) as it was defined in previous versions of OSAL, as well as the "tv\_usec" field of POSIX-style "struct timeval" values.

**See also**

[OS\\_TimeGetTotalSeconds\(\)](#)

**Parameters**

in	tm	Time interval value
----	----	---------------------

**Returns**

Number of microseconds in time interval

Definition at line 433 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_USEC, and OS\_TimeGetFractionalPart().

Here is the call graph for this function:



**10.57.2.18 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 408 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_MSEC, and OS\_TimeGetFractionalPart().

Here is the call graph for this function:



**10.57.2.19 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
----	-----------	---------------------

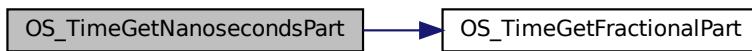
**Returns**

Number of nanoseconds in time interval

Definition at line 452 of file osapi-clock.h.

References OS\_TIME\_TICK\_RESOLUTION\_NS, and OS\_TimeGetFractionalPart().

Here is the call graph for this function:



**10.57.2.20 OS\_TimeGetSign()** static int8\_t OS\_TimeGetSign (   
     OS\_time\_t time ) [inline], [static]

Checks the sign of the time value.

#### Parameters

in	time	The time to check
----	------	-------------------

#### Return values

-1	if the time value is negative / below 0
0	if the time value is 0
1	if the time value is positive / above 0

Definition at line 608 of file osapi-clock.h.

References OS\_time\_t::ticks.

Referenced by OS\_TimeCompare().

**10.57.2.21 OS\_TimeGetSubsecondsPart()** static uint32 OS\_TimeGetSubsecondsPart (   
     OS\_time\_t tm ) [inline], [static]

Get 32-bit normalized subseconds (fractional part only) from an [OS\\_time\\_t](#) object.

Extracts the fractional part from a given [OS\\_time\\_t](#) object in maximum precision, with units of  $2^{-32}$  sec. This is a base-2 fixed-point fractional value with the point left-justified in the 32-bit value (i.e. left of MSB).

This is (mostly) compatible with the CFE "subseconds" value, where 0x80000000 represents exactly one half second, and 0 represents a full second.

#### Parameters

in	tm	Time interval value
----	----	---------------------

#### Returns

Fractional/subsecond portion of time interval as 32-bit fixed point value

Definition at line 384 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, and [OS\\_TimeGetFractionalPart\(\)](#).

Here is the call graph for this function:



**10.57.2.22 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	<i>tm</i>	Time interval value
----	-----------	---------------------

#### Returns

Whole number of microseconds in time interval

Definition at line 293 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_USEC, and OS\_time\_t::ticks.

**10.57.2.23 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	<i>tm</i>	Time interval value
----	-----------	---------------------

#### Returns

Whole number of milliseconds in time interval

Definition at line 259 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_MSEC, and OS\_time\_t::ticks.

**10.57.2.24 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	<i>tm</i>	Time interval value
----	-----------	---------------------

**Returns**

Whole number of microseconds in time interval

Definition at line 332 of file osapi-clock.h.

References OS\_TIME\_TICK\_RESOLUTION\_NS, and OS\_time\_t::ticks.

**10.57.2.25 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 225 of file osapi-clock.h.

References OS\_TIME\_TICKS\_PER\_SECOND, and OS\_time\_t::ticks.

**10.57.2.26 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 577 of file osapi-clock.h.

References OS\_time\_t::ticks.

Referenced by OS\_TimeCompare().

**10.57.2.27 OS\_TimeToRelativeMilliseconds()** `int32 OS_TimeToRelativeMilliseconds ( OS_time_t time )`

Gets a relative time value from an absolute time.

This function computes the number of milliseconds until the given absolute time value is reached in the system clock.

#### Note

This is intended to ease transitioning from a relative timeout value to an absolute timeout value. The result can be passed to any function that accepts a relative timeout, to mimic the behavior of an absolute timeout.

The return value of this function is intended to be compatible with the relative timeout parameter of various OSAL APIs e.g. [OS\\_TimedRead\(\)](#) / [OS\\_TimedWrite\(\)](#)

#### Parameters

in	<i>time</i>	An absolute time value
----	-------------	------------------------

#### Returns

Milliseconds until time value will be reached

#### Return values

<code>OS_CHECK</code>	(0) if time is the current time or is in the past
<code>OS_PEND</code>	(-1) if time is far in the future (not expressable as an int32)

## 10.58 OSAL Core Operation APIs

### Functions

- void [OS\\_Application\\_Startup](#) (void)  
*Application startup.*
- void [OS\\_Application\\_Run](#) (void)  
*Application run.*
- `int32 OS_API_Init` (void)  
*Initialization of API.*
- void [OS\\_API\\_Teardown](#) (void)  
*Teardown/de-initialization of OSAL API.*
- void [OS\\_IdleLoop](#) (void)  
*Background thread implementation - waits forever for events to occur.*
- void [OS\\_DeleteAllObjects](#) (void)  
*delete all resources created in OSAL.*
- void [OS\\_ApplicationShutdown](#) (`uint8` flag)  
*Initiate orderly shutdown.*
- void [OS\\_ApplicationExit](#) (`int32` Status)  
*Exit/Abort the application.*
- `int32 OS_RegisterEventHandler` (`OS_EventHandler_t` handler)  
*Callback routine registration.*
- `size_t OS_strlen` (`const char *s`, `size_t maxlen`)  
*get string length*

### 10.58.1 Detailed Description

These are for OSAL core operations for startup/initialization, running, and shutdown. Typically only used in bsp's, unit tests, psp's, etc.

Not intended for user application use

### 10.58.2 Function Documentation

#### 10.58.2.1 OS\_API\_Init() `int32 OS_API_Init (`     `void )`

Initialization of API.

This function initializes the internal data structures of the OS Abstraction Layer. It must be called in the application startup code before calling any other OS routines.

##### Returns

Execution status, see [OSAL Return Code Defines](#). Any error code (negative) means the OSAL can not be initialized. Typical platform specific response is to abort since additional OSAL calls will have undefined behavior.

##### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERROR</code>	Failed execution. (return value only verified in coverage test)

#### 10.58.2.2 OS\_API\_Teardown() `void OS_API_Teardown (`     `void )`

Teardown/de-initialization of OSAL API.

This is the inverse of [OS\\_API\\_Init\(\)](#). It will release all OS resources and return the system to a state similar to what it was prior to invoking [OS\\_API\\_Init\(\)](#) initially.

Normally for embedded applications, the OSAL is initialized after boot and will remain initialized in memory until the processor is rebooted. However for testing and development purposes, it is potentially useful to reset back to initial conditions.

For testing purposes, this API is designed/intended to be compatible with the `UtTest_AddTeardown()` routine provided by the UT-Assert subsystem.

##### Note

This is a "best-effort" routine and it may not always be possible/guaranteed to recover all resources, particularly in the case of off-nominal conditions, or if a resource is used outside of OSAL.

For example, while this will attempt to unload all dynamically-loaded modules, doing so may not be possible and/or may induce undefined behavior if resources are in use by tasks/functions outside of OSAL.

#### 10.58.2.3 OS\_Application\_Run() `void OS_Application_Run (`     `void )`

Application run.

Run abstraction such that the same BSP can be used for operations and testing.

#### 10.58.2.4 OS\_Application\_Startup() `void OS_Application_Startup (`     `void )`

Application startup.

Startup abstraction such that the same BSP can be used for operations and testing.

**10.58.2.5 OS\_ApplicationExit()** `void OS_ApplicationExit (`  
 `int32 Status )`

Exit/Abort the application.

Indicates that the OSAL application should exit and return control to the OS. This is intended for e.g. scripted unit testing where the test needs to end without user intervention.

This function does not return. Production code typically should not ever call this.

#### Note

This exits the entire process including tasks that have been created.

**10.58.2.6 OS\_ApplicationShutdown()** `void OS_ApplicationShutdown (`  
 `uint8 flag )`

Initiate orderly shutdown.

Indicates that the OSAL application should perform an orderly shutdown of ALL tasks, clean up all resources, and exit the application.

This allows the task currently blocked in [OS\\_IdleLoop\(\)](#) to wake up, and for that function to return to its caller.

This is preferred over e.g. [OS\\_ApplicationExit\(\)](#) which exits immediately and does not provide for any means to clean up first.

#### Parameters

in	<i>flag</i>	set to true to initiate shutdown, false to cancel
----	-------------	---

**10.58.2.7 OS\_DeleteAllObjects()** `void OS_DeleteAllObjects (`  
 `void )`

delete all resources created in OSAL.

provides a means to clean up all resources allocated by this instance of OSAL. It would typically be used during an orderly shutdown but may also be helpful for testing purposes.

**10.58.2.8 OS\_IdleLoop()** `void OS_IdleLoop (`  
 `void )`

Background thread implementation - waits forever for events to occur.

This should be called from the BSP main routine or initial thread after all other board and application initialization has taken place and all other tasks are running.

Typically just waits forever until "OS\_shutdown" flag becomes true.

**10.58.2.9 OS\_RegisterEventHandler()** `int32 OS_RegisterEventHandler (`  
 `OS_EventHandler_t handler )`

Callback routine registration.

This hook enables the application code to perform extra platform-specific operations on various system events such as resource creation/deletion.

#### Note

Some events are invoked while the resource is "locked" and therefore application-defined handlers for these events should not block or attempt to access other OSAL resources.

**Parameters**

in	<i>handler</i>	The application-provided event handler (must not be null)
----	----------------	---

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if handler is NULL

**10.58.2.10 OS\_strlen()** size\_t OS\_strlen (   
     const char \* *s*,  
     size\_t *maxlen* )

get string length

Provides an OSAL routine to get the functionality of the (non-C99) "strnlen()" function, via the C89/C99 standard "memchr()" function instead.

**Parameters**

in	<i>s</i>	The input string
in	<i>maxlen</i>	Maximum length to check

**Return values**

<i>Length</i>	of the string or maxlen, whichever is smaller.
---------------	--

Referenced by CF\_CFDP\_SendMd(), and CF\_WriteHistoryEntryToFile().

## 10.59 OSAL Condition Variable APIs

**Functions**

- [int32 OS\\_CondVarCreate \(osal\\_id\\_t \\*var\\_id, const char \\*var\\_name, uint32 options\)](#)  
*Creates a condition variable resource.*
- [int32 OS\\_CondVarLock \(osal\\_id\\_t var\\_id\)](#)  
*Locks/Acquires the underlying mutex associated with a condition variable.*
- [int32 OS\\_CondVarUnlock \(osal\\_id\\_t var\\_id\)](#)  
*Unlocks/Releases the underlying mutex associated with a condition variable.*
- [int32 OS\\_CondVarSignal \(osal\\_id\\_t var\\_id\)](#)  
*Signals the condition variable resource referenced by var\_id.*
- [int32 OS\\_CondVarBroadcast \(osal\\_id\\_t var\\_id\)](#)  
*Broadcasts the condition variable resource referenced by var\_id.*
- [int32 OS\\_CondVarWait \(osal\\_id\\_t var\\_id\)](#)  
*Waits on the condition variable object referenced by var\_id.*

- `int32 OS_CondVarTimedWait (osal_id_t var_id, const OS_time_t *abs_wakeup_time)`  
*Time-limited wait on the condition variable object referenced by var\_id.*
- `int32 OS_CondVarDelete (osal_id_t var_id)`  
*Deletes the specified condition variable.*
- `int32 OS_CondVarGetIdByName (osal_id_t *var_id, const char *var_name)`  
*Find an existing condition variable ID by name.*
- `int32 OS_CondVarGetInfo (osal_id_t var_id, OS_condvar_prop_t *condvar_prop)`  
*Fill a property object buffer with details regarding the resource.*

### 10.59.1 Detailed Description

### 10.59.2 Function Documentation

#### 10.59.2.1 OS\_CondVarBroadcast() `int32 OS_CondVarBroadcast (` `osal_id_t var_id )`

Broadcasts the condition variable resource referenced by var\_id.

This function may be used to indicate when the state of a data object has been changed.

If there are threads blocked on the condition variable object referenced by var\_id when this function is called, all threads will be unblocked.

Note that although all threads are unblocked, because the mutex is re-acquired before the wait function returns, only a single task will be testing the condition at a given time. The order with which each blocked task runs is determined by the scheduling policy.

#### Parameters

in	<code>var_id</code>	The object ID to operate on
----	---------------------	-----------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid condition variable

#### 10.59.2.2 OS\_CondVarCreate() `int32 OS_CondVarCreate (` `osal_id_t * var_id,` `const char * var_name,` `uint32 options )`

Creates a condition variable resource.

A condition variable adds a more sophisticated synchronization option for mutexes, such that it can operate on arbitrary user-defined conditions rather than simply a counter or boolean (as in the case of simple semaphores).

Creating a condition variable resource in OSAL will in turn create both a basic mutex as well as a synchronization overlay. The underlying mutex is similar to the mutex functionality provided by the OSAL mutex subsystem, and can be locked and unlocked normally.

This mutex is intended to protect access to any arbitrary user-defined data object that serves as the condition being tested.

A task that needs a particular state of the object should follow this general flow:

- Lock the underlying mutex
- Test for the condition being waited for (a user-defined check on user-defined data)
- If condition IS NOT met, then call [OS\\_CondVarWait\(\)](#) to wait, then repeat test
- If condition IS met, then unlock the underlying mutex and continue

A task that changes the state of the object should follow this general flow:

- Lock the underlying mutex
- Change the state as necessary
- Call either [OS\\_CondVarSignal\(\)](#) or [OS\\_CondVarBroadcast\(\)](#)
- Unlock the underlying mutex

#### Parameters

out	<i>var_id</i>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<i>var_name</i>	the name of the new resource to create (must not be null)
in	<i>options</i>	reserved for future use. Should be passed as 0.

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if var_id or var_name are NULL
<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_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

### 10.59.2.3 OS\_CondVarDelete() `int32 OS_CondVarDelete ( osal_id_t var_id )`

Deletes the specified condition variable.

Delete the condition variable and releases any related system resources.

#### Parameters

in	<i>var_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 condvar

**10.59.2.4 OS\_CondVarGetIdByName()** `int32 OS_CondVarGetIdByName (`

```
    osal_id_t * var_id,
    const char * var_name )
```

Find an existing condition variable ID by name.

This function tries to find an existing condition variable ID given the name. The id is returned through var\_id.

**Parameters**

<i>out</i>	<i>var_id</i>	will be set to the ID of the existing resource
<i>in</i>	<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**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	is var_id or var_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

**10.59.2.5 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**

<i>in</i>	<i>var_id</i>	The object ID to operate on
<i>out</i>	<i>condvar_prop</i>	The property object buffer to fill (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 id passed in is not a valid semaphore
<a href="#">OS_INVALID_POINTER</a>	if the mut_prop pointer is null

**10.59.2.6 OS\_CondVarLock()** `int32 OS_CondVarLock ( osal_id_t var_id )`

Locks/Acquires the underlying mutex associated with a condition variable.

The mutex should always be locked by a task before reading or modifying the data object associated with a condition variable.

**Note**

This lock must be acquired by a task before invoking [OS\\_CondVarWait\(\)](#) or [OS\\_CondVarTimedWait\(\)](#) on the same condition variable.

**Parameters**

in	<i>var←_id</i>	The object ID to operate on
----	----------------	-----------------------------

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the id passed in is not a valid condition variable

**10.59.2.7 OS\_CondVarSignal()** `int32 OS_CondVarSignal ( osal_id_t var_id )`

Signals the condition variable resource referenced by var\_id.

This function may be used to indicate when the state of a data object has been changed.

If there are threads blocked on the condition variable object referenced by var\_id when this function is called, one of those threads will be unblocked, as determined by the scheduling policy.

**Parameters**

in	<i>var←_id</i>	The object ID to operate on
----	----------------	-----------------------------

**Returns**

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid condition variable

```
10.59.2.8 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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	the id passed in is not a valid condvar

```
10.59.2.9 OS_CondVarUnlock() int32 OS_CondVarUnlock (
    osal_id_t var_id )
```

Unlocks/Releases the underlying mutex associated with a condition variable.

The mutex should be unlocked by a task once reading or modifying the data object associated with a condition variable is complete.

## Parameters

in	<i>var_id</i>	The object ID to operate on
----	---------------	-----------------------------

## Returns

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid condition variable

**10.59.2.10 OS\_CondVarWait()** `int32 OS_CondVarWait (`  
    `osal_id_t var_id )`

Waits on the condition variable object referenced by var\_id.

The calling task will be blocked until another task calls the function [OS\\_CondVarSignal\(\)](#) or [OS\\_CondVarBroadcast\(\)](#) on the same condition variable.

The underlying mutex associated with the condition variable must be locked and owned by the calling task at the time this function is invoked. As part of this call, the mutex will be unlocked as the task blocks. This is done in such a way that there is no possibility that another task could acquire the mutex before the calling task has actually blocked.

This atomicity with respect to blocking the task and unlocking the mutex is a critical difference between condition variables and other synchronization primitives. It avoids a window of opportunity where inherent in the simpler synchronization resource types where the state of the data could change between the time that the calling task tested the state and the time that the task actually blocks on the sync resource.

#### Parameters

in	<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>	the id passed in is not a valid condvar

## 10.60 OSAL Counting Semaphore APIs

#### Functions

- `int32 OS_CountSemCreate (osal_id_t *sem_id, const char *sem_name, uint32 sem_initial_value, uint32 options)`  
*Creates a counting semaphore.*
- `int32 OS_CountSemGive (osal_id_t sem_id)`  
*Increment the semaphore value.*
- `int32 OS_CountSemTake (osal_id_t sem_id)`  
*Decrement the semaphore value.*
- `int32 OS_CountSemTimedWait (osal_id_t sem_id, uint32 msecs)`  
*Decrement the semaphore value with timeout.*
- `int32 OS_CountSemDelete (osal_id_t sem_id)`  
*Deletes the specified counting Semaphore.*
- `int32 OS_CountSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`  
*Find an existing semaphore ID by name.*
- `int32 OS_CountSemGetInfo (osal_id_t sem_id, OS_count_sem_prop_t *count_prop)`  
*Fill a property object buffer with details regarding the resource.*

### 10.60.1 Detailed Description

### 10.60.2 Function Documentation

```
10.60.2.1 OS_CountSemCreate() int32 OS_CountSemCreate (
    osal_id_t * sem_id,
    const char * sem_name,
    uint32 sem_initial_value,
    uint32 options )
```

Creates a counting semaphore.

Creates a counting semaphore with initial value specified by `sem_initial_value` and name specified by `sem_name`. `sem_id` will be returned to the caller.

#### Note

Underlying RTOS implementations may or may not impose a specific upper limit to the value of a counting semaphore. If the OS has a specific limit and the `sem_initial_value` exceeds this limit, then `OS_INVALID_SEM_VALUE` is returned. On other implementations, any 32-bit integer value may be acceptable. For maximum portability, it is recommended to keep counting semaphore values within the range of a "short int" (i.e. between 0 and 32767). Many platforms do accept larger values, but may not be guaranteed.

#### Parameters

out	<code>sem_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<code>sem_name</code>	the name of the new resource to create (must not be null)
in	<code>sem_initial_value</code>	the initial value of the counting semaphore
in	<code>options</code>	Reserved for future use, should be passed as 0.

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if <code>sem_name</code> or <code>sem_id</code> 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 all of the semaphore ids are taken
<code>OS_ERR_NAME_TAKEN</code>	if this is already the name of a counting semaphore
<code>OS_INVALID_SEM_VALUE</code>	if the semaphore value is too high (return value only verified in coverage test)
<code>OS_SEM_FAILURE</code>	if an unspecified implementation error occurs (return value only verified in coverage test)

```
10.60.2.2 OS_CountSemDelete() int32 OS_CountSemDelete (
    osal_id_t sem_id )
```

Deletes the specified counting Semaphore.

**Parameters**

in	<i>sem_id</i>	The object ID to delete
----	---------------	-------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

**10.60.2.3 OS\_CountSemGetIdByName()** `int32 OS_CountSemGetIdByName (`

```
    osal_id_t * sem_id,
    const char * sem_name )
```

Find an existing semaphore ID by name.

This function tries to find a counting sem Id given the name of a count\_sem. The id is returned through sem\_id

**Parameters**

out	<i>sem_id</i>	will be set to the ID of the existing resource
in	<i>sem_name</i>	the name of the existing resource to find (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	is semid or sem_name are NULL pointers
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <a href="#">OS_MAX_API_NAME</a>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name was not found in the table

Referenced by CF\_CFDP\_InitEngine().

**10.60.2.4 OS\_CountSemGetInfo()** `int32 OS_CountSemGetInfo (`

```
    osal_id_t sem_id,
    OS_count_sem_prop_t * count_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info( name and creator) about the specified counting semaphore.

**Parameters**

in	<i>sem_id</i>	The object ID to operate on
out	<i>count_prop</i>	The property object buffer to fill (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the count_prop pointer is null
<i>OS_ERR_NOT_IMPLEMENTED</i>	Not implemented.

**10.60.2.5 OS\_CountSemGive()** `int32 OS_CountSemGive (`

`osal_id_t sem_id )`

Increment the semaphore value.

The function unlocks the semaphore referenced by *sem\_id* by performing a semaphore unlock operation on that semaphore. If the semaphore value resulting from this operation is positive, then no threads were blocked waiting for the semaphore to become unlocked; the semaphore value is simply incremented for this semaphore.

**Parameters**

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 counting semaphore
<i>OS_SEM_FAILURE</i>	if an unspecified implementation error occurs (return value only verified in coverage test)

**10.60.2.6 OS\_CountSemTake()** `int32 OS_CountSemTake (`

`osal_id_t sem_id )`

Decrement the semaphore value.

The locks the semaphore referenced by *sem\_id* by performing a semaphore lock operation on that semaphore. If the semaphore value is currently zero, then the calling thread shall not return from the call until it either locks the semaphore or the call is interrupted.

**Parameters**

in	<i>sem-&gt;_id</i>	The object 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>	the Id passed in is not a valid counting semaphore
<a href="#">OS_SEM_FAILURE</a>	if an unspecified implementation error occurs (return value only verified in coverage test)

**10.60.2.7 OS\_CountSemTimedWait()** `int32 OS_CountSemTimedWait ( osal_id_t sem_id, uint32 msecs )`

Decrement the semaphore value with timeout.

The function locks the semaphore referenced by `sem_id`. However, if the semaphore cannot be locked without waiting for another process or thread to unlock the semaphore, this wait shall be terminated when the specified timeout, `msecs`, expires.

**Parameters**

in	<i>sem-&gt;_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**

<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 implementation error occurs (return value only verified in coverage test)

Referenced by CF\_CFDP\_MsgOutGet().

## 10.61 OSAL Directory APIs

**Functions**

- `int32 OS_DirectoryOpen (osal_id_t *dir_id, const char *path)`

*Opens a directory.*

- `int32 OS_DirectoryClose (osal_id_t dir_id)`  
*Closes an open directory.*
- `int32 OS_DirectoryRewind (osal_id_t dir_id)`  
*Rewinds an open directory.*
- `int32 OS_DirectoryRead (osal_id_t dir_id, os_dirent_t *dirent)`  
*Reads the next name in the directory.*
- `int32 OS_mkdir (const char *path, uint32 access)`  
*Makes a new directory.*
- `int32 OS_rmdir (const char *path)`  
*Removes a directory from the file system.*

### 10.61.1 Detailed Description

### 10.61.2 Function Documentation

#### 10.61.2.1 OS\_DirectoryClose() `int32 OS_DirectoryClose (` `osal_id_t dir_id )`

Closes an open directory.

The directory referred to by `dir_id` will be closed

##### Parameters

in	<code>dir←_id</code>	The handle ID of the directory
----	----------------------	--------------------------------

##### Returns

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

##### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the directory handle is invalid

Referenced by CF\_CFDP\_DisableEngine(), and CF\_CFDP\_ProcessPlaybackDirectory().

#### 10.61.2.2 OS\_DirectoryOpen() `int32 OS_DirectoryOpen (` `osal_id_t * dir_id,` `const char * path )`

Opens a directory.

Prepares for reading the files within a directory

##### Parameters

out	<code>dir←_id</code>	Location to store handle ID of the directory (must not be null)
in	<code>path</code>	The directory to open (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if dir_id or path is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the path argument exceeds the maximum length
<i>OS_FS_ERR_PATH_INVALID</i>	if the path argument is not valid
<i>OS_ERROR</i>	if the directory could not be opened

Referenced by CF\_CFDP\_PlaybackDir\_Initiate().

```
10.61.2.3 OS_DirectoryRead() int32 OS_DirectoryRead (
    osal_id_t dir_id,
    os_dirent_t * dirent )
```

Reads the next name in the directory.

Obtains directory entry data for the next file from an open directory

**Parameters**

in	<i>dir←_id</i>	The handle ID of the directory
out	<i>dirent</i>	Buffer to store directory entry information (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if dirent argument is NULL
<i>OS_ERR_INVALID_ID</i>	if the directory handle is invalid
<i>OS_ERROR</i>	at the end of the directory or if the OS call otherwise fails

Referenced by CF\_CFDP\_ProcessPlaybackDirectory().

```
10.61.2.4 OS_DirectoryRewind() int32 OS_DirectoryRewind (
    osal_id_t dir_id )
```

Rewinds an open directory.

Resets a directory read handle back to the first file.

**Parameters**

in	<i>dir←_id</i>	The handle ID of the directory
----	----------------	--------------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the directory handle is invalid

```
10.61.2.5 OS_mkdir() int32 OS_mkdir (
    const char * path,
    uint32 access )
```

Makes a new directory.

Makes a directory specified by path.

**Parameters**

in	<i>path</i>	The new directory name (must not be null)
in	<i>access</i>	The permissions for the directory (reserved for future use)

**Note**

Current implementations do not utilize the "access" parameter. Applications should still pass the intended value ([OS\\_READ\\_WRITE](#) or [OS\\_READ\\_ONLY](#)) to be compatible with future implementations.

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if path is NULL
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the path is too long to be stored locally
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_ERROR</i>	if the OS call fails (return value only verified in coverage test)

Referenced by CF\_CFDP\_InitEngine().

```
10.61.2.6 OS_rmdir() int32 OS_rmdir (
    const char * path )
```

Removes a directory from the file system.

Removes a directory from the structure. The directory must be empty prior to this operation.

**Parameters**

in	<i>path</i>	The directory to remove
----	-------------	-------------------------

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if path is NULL
<code>OS_FS_ERR_PATH_INVALID</code>	if path cannot be parsed
<code>OS_FS_ERR_PATH_TOO_LONG</code>	
<code>OS_ERROR</code>	if the directory remove operation failed (return value only verified in coverage test)

## 10.62 OSAL Return Code Defines

**Macros**

- `#define OS_SUCCESS (0)`  
*Successful execution.*
- `#define OS_ERROR (-1)`  
*Failed execution.*
- `#define OS_INVALID_POINTER (-2)`  
*Invalid pointer.*
- `#define OS_ERROR_ADDRESS_MISALIGNED (-3)`  
*Address misalignment.*
- `#define OS_ERROR_TIMEOUT (-4)`  
*Error timeout.*
- `#define OS_INVALID_INT_NUM (-5)`  
*Invalid Interrupt number.*
- `#define OS_SEM_FAILURE (-6)`  
*Semaphore failure.*
- `#define OS_SEM_TIMEOUT (-7)`  
*Semaphore timeout.*
- `#define OS_QUEUE_EMPTY (-8)`  
*Queue empty.*
- `#define OS_QUEUE_FULL (-9)`  
*Queue full.*
- `#define OS_QUEUE_TIMEOUT (-10)`  
*Queue timeout.*
- `#define OS_QUEUE_INVALID_SIZE (-11)`  
*Queue invalid size.*
- `#define OS_QUEUE_ID_ERROR (-12)`  
*Queue ID error.*
- `#define OS_ERR_NAME_TOO_LONG (-13)`  
*name length including null terminator greater than `OS_MAX_API_NAME`*
- `#define OS_ERR_NO_FREE_IDS (-14)`  
*No free IDs.*
- `#define OS_ERR_NAME_TAKEN (-15)`  
*Name taken.*

- #define **OS\_ERR\_INVALID\_ID** (-16)  
*Invalid ID.*
- #define **OS\_ERR\_NAME\_NOT\_FOUND** (-17)  
*Name not found.*
- #define **OS\_ERR\_SEM\_NOT\_FULL** (-18)  
*Semaphore not full.*
- #define **OS\_ERR\_INVALID\_PRIORITY** (-19)  
*Invalid priority.*
- #define **OS\_INVALID\_SEM\_VALUE** (-20)  
*Invalid semaphore value.*
- #define **OS\_ERR\_FILE** (-27)  
*File error.*
- #define **OS\_ERR\_NOT\_IMPLEMENTED** (-28)  
*Not implemented.*
- #define **OS\_TIMER\_ERR\_INVALID\_ARGS** (-29)  
*Timer invalid arguments.*
- #define **OS\_TIMER\_ERR\_TIMER\_ID** (-30)  
*Timer ID error.*
- #define **OS\_TIMER\_ERR\_UNAVAILABLE** (-31)  
*Timer unavailable.*
- #define **OS\_TIMER\_ERR\_INTERNAL** (-32)  
*Timer internal error.*
- #define **OS\_ERR\_OBJECT\_IN\_USE** (-33)  
*Object in use.*
- #define **OS\_ERR\_BAD\_ADDRESS** (-34)  
*Bad address.*
- #define **OS\_ERR\_INCORRECT\_OBJ\_STATE** (-35)  
*Incorrect object state.*
- #define **OS\_ERR\_INCORRECT\_OBJ\_TYPE** (-36)  
*Incorrect object type.*
- #define **OS\_ERR\_STREAM\_DISCONNECTED** (-37)  
*Stream disconnected.*
- #define **OS\_ERR\_OPERATION\_NOT\_SUPPORTED** (-38)  
*Requested operation not support on supplied object(s)*
- #define **OS\_ERR\_INVALID\_SIZE** (-40)  
*Invalid Size.*
- #define **OS\_ERR\_OUTPUT\_TOO\_LARGE** (-41)  
*Size of output exceeds limit*
- #define **OS\_ERR\_INVALID\_ARGUMENT** (-42)  
*Invalid argument value (other than ID or size)*
- #define **OS\_ERR\_TRY AGAIN** (-43)  
*Failure is temporary in nature, call may be repeated.*
- #define **OS\_ERR\_EMPTY\_SET** (-44)  
*Address or name lookup returned no results.*
- #define **OS\_FS\_ERR\_PATH\_TOO\_LONG** (-103)  
*FS path too long.*

- #define OS\_FS\_ERR\_NAME\_TOO\_LONG (-104)  
*FS name too long.*
- #define OS\_FS\_ERR\_DRIVE\_NOT\_CREATED (-106)  
*FS drive not created.*
- #define OS\_FS\_ERR\_DEVICE\_NOT\_FREE (-107)  
*FS device not free.*
- #define OS\_FS\_ERR\_PATH\_INVALID (-108)  
*FS path invalid.*

### 10.62.1 Detailed Description

The specific status/return code definitions listed in this section may be extended or refined in future versions of OSAL.

#### Note

Application developers should assume that any OSAL API may return any status value listed here. While the documentation of each OSAL API function indicates the return/status values that function may directly generate, functions may also pass through other status codes from related functions, so that list should not be considered absolute/exhaustive.

The int32 data type should be used to store an OSAL status code. Negative values will always represent errors, while non-negative values indicate success. Most APIs specifically return OS\_SUCCESS (0) upon successful execution, but some return a nonzero value, such as data size.

Ideally, in order to more easily adapt to future OSAL versions and status code extensions/refinements, applications should typically check for errors as follows:

```
int32 status;
status = OS_TaskCreate(...);  (or any other API)
if (status < OS_SUCCESS)
{
    handle or report error....
    may also check for specific codes here.
}
else
{
    handle normal/successful status...
}
```

### 10.62.2 Macro Definition Documentation

#### 10.62.2.1 OS\_ERR\_BAD\_ADDRESS #define OS\_ERR\_BAD\_ADDRESS (-34)

Bad address.

Definition at line 124 of file osapi-error.h.

#### 10.62.2.2 OS\_ERR\_EMPTY\_SET #define OS\_ERR\_EMPTY\_SET (-44)

Address or name lookup returned no results.

Definition at line 133 of file osapi-error.h.

#### 10.62.2.3 OS\_ERR\_FILE #define OS\_ERR\_FILE (-27)

File error.

Definition at line 117 of file osapi-error.h.

**10.62.2.4 OS\_ERR\_INCORRECT\_OBJ\_STATE** #define OS\_ERR\_INCORRECT\_OBJ\_STATE (-35)  
Incorrect object state.  
Definition at line 125 of file osapi-error.h.

**10.62.2.5 OS\_ERR\_INCORRECT\_OBJ\_TYPE** #define OS\_ERR\_INCORRECT\_OBJ\_TYPE (-36)  
Incorrect object type.  
Definition at line 126 of file osapi-error.h.

**10.62.2.6 OS\_ERR\_INVALID\_ARGUMENT** #define OS\_ERR\_INVALID\_ARGUMENT (-42)  
Invalid argument value (other than ID or size)  
Definition at line 131 of file osapi-error.h.

**10.62.2.7 OS\_ERR\_INVALID\_ID** #define OS\_ERR\_INVALID\_ID (-16)  
Invalid ID.  
Definition at line 112 of file osapi-error.h.

**10.62.2.8 OS\_ERR\_INVALID\_PRIORITY** #define OS\_ERR\_INVALID\_PRIORITY (-19)  
Invalid priority.  
Definition at line 115 of file osapi-error.h.

**10.62.2.9 OS\_ERR\_INVALID\_SIZE** #define OS\_ERR\_INVALID\_SIZE (-40)  
Invalid Size.  
Definition at line 129 of file osapi-error.h.

**10.62.2.10 OS\_ERR\_NAME\_NOT\_FOUND** #define OS\_ERR\_NAME\_NOT\_FOUND (-17)  
Name not found.  
Definition at line 113 of file osapi-error.h.

**10.62.2.11 OS\_ERR\_NAME\_TAKEN** #define OS\_ERR\_NAME\_TAKEN (-15)  
Name taken.  
Definition at line 111 of file osapi-error.h.

**10.62.2.12 OS\_ERR\_NAME\_TOO\_LONG** #define OS\_ERR\_NAME\_TOO\_LONG (-13)  
name length including null terminator greater than [OS\\_MAX\\_API\\_NAME](#)  
Definition at line 109 of file osapi-error.h.

**10.62.2.13 OS\_ERR\_NO\_FREE\_IDS** #define OS\_ERR\_NO\_FREE\_IDS (-14)  
No free IDs.  
Definition at line 110 of file osapi-error.h.

**10.62.2.14 OS\_ERR\_NOT\_IMPLEMENTED** #define OS\_ERR\_NOT\_IMPLEMENTED (-28)

Not implemented.

Definition at line 118 of file osapi-error.h.

**10.62.2.15 OS\_ERR\_OBJECT\_IN\_USE** #define OS\_ERR\_OBJECT\_IN\_USE (-33)

Object in use.

Definition at line 123 of file osapi-error.h.

**10.62.2.16 OS\_ERR\_OPERATION\_NOT\_SUPPORTED** #define OS\_ERR\_OPERATION\_NOT\_SUPPORTED (-38)

Requested operation not support on supplied object(s)

Definition at line 128 of file osapi-error.h.

**10.62.2.17 OS\_ERR\_OUTPUT\_TOO\_LARGE** #define OS\_ERR\_OUTPUT\_TOO\_LARGE (-41)

Size of output exceeds limit

Definition at line 130 of file osapi-error.h.

**10.62.2.18 OS\_ERR\_SEM\_NOT\_FULL** #define OS\_ERR\_SEM\_NOT\_FULL (-18)

Semaphore not full.

Definition at line 114 of file osapi-error.h.

**10.62.2.19 OS\_ERR\_STREAM\_DISCONNECTED** #define OS\_ERR\_STREAM\_DISCONNECTED (-37)

Stream disconnected.

Definition at line 127 of file osapi-error.h.

**10.62.2.20 OS\_ERR\_TRY AGAIN** #define OS\_ERR\_TRY AGAIN (-43)

Failure is temporary in nature, call may be repeated.

Definition at line 132 of file osapi-error.h.

**10.62.2.21 OS\_ERROR** #define OS\_ERROR (-1)

Failed execution.

Definition at line 97 of file osapi-error.h.

**10.62.2.22 OS\_ERROR\_ADDRESS\_MISALIGNED** #define OS\_ERROR\_ADDRESS\_MISALIGNED (-3)

Address misalignment.

Definition at line 99 of file osapi-error.h.

**10.62.2.23 OS\_ERROR\_TIMEOUT** #define OS\_ERROR\_TIMEOUT (-4)

Error timeout.

Definition at line 100 of file osapi-error.h.

**10.62.2.24 OS\_FS\_ERR\_DEVICE\_NOT\_FREE** #define OS\_FS\_ERR\_DEVICE\_NOT\_FREE (-107)  
FS device not free.  
Definition at line 146 of file osapi-error.h.

**10.62.2.25 OS\_FS\_ERR\_DRIVE\_NOT\_CREATED** #define OS\_FS\_ERR\_DRIVE\_NOT\_CREATED (-106)  
FS drive not created.  
Definition at line 145 of file osapi-error.h.

**10.62.2.26 OS\_FS\_ERR\_NAME\_TOO\_LONG** #define OS\_FS\_ERR\_NAME\_TOO\_LONG (-104)  
FS name too long.  
Definition at line 144 of file osapi-error.h.

**10.62.2.27 OS\_FS\_ERR\_PATH\_INVALID** #define OS\_FS\_ERR\_PATH\_INVALID (-108)  
FS path invalid.  
Definition at line 147 of file osapi-error.h.

**10.62.2.28 OS\_FS\_ERR\_PATH\_TOO\_LONG** #define OS\_FS\_ERR\_PATH\_TOO\_LONG (-103)  
FS path too long.  
Definition at line 143 of file osapi-error.h.

**10.62.2.29 OS\_INVALID\_INT\_NUM** #define OS\_INVALID\_INT\_NUM (-5)  
Invalid Interrupt number.  
Definition at line 101 of file osapi-error.h.

**10.62.2.30 OS\_INVALID\_POINTER** #define OS\_INVALID\_POINTER (-2)  
Invalid pointer.  
Definition at line 98 of file osapi-error.h.

**10.62.2.31 OS\_INVALID\_SEM\_VALUE** #define OS\_INVALID\_SEM\_VALUE (-20)  
Invalid semaphore value.  
Definition at line 116 of file osapi-error.h.

**10.62.2.32 OS\_QUEUE\_EMPTY** #define OS\_QUEUE\_EMPTY (-8)  
Queue empty.  
Definition at line 104 of file osapi-error.h.

**10.62.2.33 OS\_QUEUE\_FULL** #define OS\_QUEUE\_FULL (-9)  
Queue full.  
Definition at line 105 of file osapi-error.h.

**10.62.2.34 OS\_QUEUE\_ID\_ERROR** #define OS\_QUEUE\_ID\_ERROR (-12)

Queue ID error.

Definition at line 108 of file osapi-error.h.

**10.62.2.35 OS\_QUEUE\_INVALID\_SIZE** #define OS\_QUEUE\_INVALID\_SIZE (-11)

Queue invalid size.

Definition at line 107 of file osapi-error.h.

**10.62.2.36 OS\_QUEUE\_TIMEOUT** #define OS\_QUEUE\_TIMEOUT (-10)

Queue timeout.

Definition at line 106 of file osapi-error.h.

**10.62.2.37 OS\_SEM\_FAILURE** #define OS\_SEM\_FAILURE (-6)

Semaphore failure.

Definition at line 102 of file osapi-error.h.

**10.62.2.38 OS\_SEM\_TIMEOUT** #define OS\_SEM\_TIMEOUT (-7)

Semaphore timeout.

Definition at line 103 of file osapi-error.h.

**10.62.2.39 OS\_SUCCESS** #define OS\_SUCCESS (0)

Successful execution.

Definition at line 96 of file osapi-error.h.

**10.62.2.40 OS\_TIMER\_ERR\_INTERNAL** #define OS\_TIMER\_ERR\_INTERNAL (-32)

Timer internal error.

Definition at line 122 of file osapi-error.h.

**10.62.2.41 OS\_TIMER\_ERR\_INVALID\_ARGS** #define OS\_TIMER\_ERR\_INVALID\_ARGS (-29)

Timer invalid arguments.

Definition at line 119 of file osapi-error.h.

**10.62.2.42 OS\_TIMER\_ERR\_TIMER\_ID** #define OS\_TIMER\_ERR\_TIMER\_ID (-30)

Timer ID error.

Definition at line 120 of file osapi-error.h.

**10.62.2.43 OS\_TIMER\_ERR\_UNAVAILABLE** #define OS\_TIMER\_ERR\_UNAVAILABLE (-31)

Timer unavailable.

Definition at line 121 of file osapi-error.h.

## 10.63 OSAL Error Info APIs

### Functions

- static long `OS_StatusToInteger (osal_status_t Status)`  
*Convert a status code to a native "long" type.*
- `int32 OS_GetErrorName (int32 error_num, os_err_name_t *err_name)`  
*Convert an error number to a string.*
- `char * OS_StatusToString (osal_status_t status, os_status_string_t *status_string)`  
*Convert status to a string.*

#### 10.63.1 Detailed Description

#### 10.63.2 Function Documentation

**10.63.2.1 OS\_GetErrorName()** `int32 OS_GetErrorName (`  
    `int32 error_num,`  
    `os_err_name_t * err_name )`

Convert an error number to a string.

##### Parameters

in	<code>error_num</code>	Error number to convert
out	<code>err_name</code>	Buffer to store error string

##### Returns

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

##### Return values

<code>OS_SUCCESS</code>	if successfully converted to a string
<code>OS_INVALID_POINTER</code>	if err_name is NULL
<code>OS_ERROR</code>	if error could not be converted

**10.63.2.2 OS\_StatusToInteger()** `static long OS_StatusToInteger (`  
    `osal_status_t Status ) [inline], [static]`

Convert a status code to a native "long" type.

For printing or logging purposes, this converts the given status code to a "long" (signed integer) value. It should be used in conjunction with the "%ld" conversion specifier in printf-style statements.

##### Parameters

in	<code>Status</code>	Execution status, see <a href="#">OSAL Return Code Defines</a>
----	---------------------	--

**Returns**

Same status value converted to the "long" data type

Definition at line 166 of file osapi-error.h.

```
10.63.2.3 OS_StatusToString() char* OS_StatusToString (
    osal_status_t status,
    os_status_string_t * status_string )
```

Convert status to a string.

**Parameters**

in	<i>status</i>	Status value to convert
out	<i>status_string</i>	Buffer to store status converted to string

**Returns**

Passed in string pointer

## 10.64 OSAL File Access Option Defines

**Macros**

- #define OS\_READ\_ONLY 0
- #define OS\_WRITE\_ONLY 1
- #define OS\_READ\_WRITE 2

### 10.64.1 Detailed Description

### 10.64.2 Macro Definition Documentation

**10.64.2.1 OS\_READ\_ONLY** #define OS\_READ\_ONLY 0

Read only file access

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

**10.64.2.2 OS\_READ\_WRITE** #define OS\_READ\_WRITE 2

Read write file access

Definition at line 37 of file osapi-file.h.

**10.64.2.3 OS\_WRITE\_ONLY** #define OS\_WRITE\_ONLY 1

Write only file access

Definition at line 36 of file osapi-file.h.

## 10.65 OSAL Reference Point For Seek Offset Defines

**Macros**

- #define OS\_SEEK\_SET 0
- #define OS\_SEEK\_CUR 1
- #define OS\_SEEK\_END 2

### 10.65.1 Detailed Description

### 10.65.2 Macro Definition Documentation

#### 10.65.2.1 OS\_SEEK\_CUR #define OS\_SEEK\_CUR 1

Seek offset current

Definition at line 44 of file osapi-file.h.

#### 10.65.2.2 OS\_SEEK\_END #define OS\_SEEK\_END 2

Seek offset end

Definition at line 45 of file osapi-file.h.

#### 10.65.2.3 OS\_SEEK\_SET #define OS\_SEEK\_SET 0

Seek offset set

Definition at line 43 of file osapi-file.h.

## 10.66 OSAL Standard File APIs

### Functions

- `int32 OS_OpenCreate (osal_id_t *filedes, const char *path, int32 flags, int32 access_mode)`  
*Open or create a file.*
- `int32 OS_close (osal_id_t filedes)`  
*Closes an open file handle.*
- `int32 OS_read (osal_id_t filedes, void *buffer, size_t nbytes)`  
*Read from a file handle.*
- `int32 OS_write (osal_id_t filedes, const void *buffer, size_t nbytes)`  
*Write to a file handle.*
- `int32 OS_TimedReadAbs (osal_id_t filedes, void *buffer, size_t nbytes, OS_time_t abstime)`  
*File/Stream input read with a timeout.*
- `int32 OS_TimedRead (osal_id_t filedes, void *buffer, size_t nbytes, int32 timeout)`  
*File/Stream input read with a timeout.*
- `int32 OS_TimedWriteAbs (osal_id_t filedes, const void *buffer, size_t nbytes, OS_time_t abstime)`  
*File/Stream output write with a timeout.*
- `int32 OS_TimedWrite (osal_id_t filedes, const void *buffer, size_t nbytes, int32 timeout)`  
*File/Stream output write with a timeout.*
- `int32 OS_FileAllocate (osal_id_t filedes, osal_offset_t offset, osal_offset_t len)`  
*Pre-allocates space at the given file location.*
- `int32 OS_FileTruncate (osal_id_t filedes, osal_offset_t len)`  
*Changes the size of the file.*
- `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, osal_offset_t offset, uint32 whence)`  
*Seeks to the specified position of an open file.*
- `int32 OS_remove (const char *path)`

*Removes a file from the file system.*

- [int32 OS\\_rename](#) (const char \*old\_filename, const char \*new\_filename)  
*Renames a file.*
- [int32 OS\\_cp](#) (const char \*src, const char \*dest)  
*Copies a single file from src to dest.*
- [int32 OS\\_mv](#) (const char \*src, const char \*dest)  
*Move a single file from src to dest.*
- [int32 OS\\_FDGetInfo](#) (osal\_id\_t filedes, OS\_file\_prop\_t \*fd\_prop)  
*Obtain information about an open file.*
- [int32 OS\\_FileOpenCheck](#) (const char \*Filename)  
*Checks to see if a file is open.*
- [int32 OS\\_CloseAllFiles](#) (void)  
*Close all open files.*
- [int32 OS\\_CloseFileByName](#) (const char \*Filename)  
*Close a file by filename.*

### 10.66.1 Detailed Description

### 10.66.2 Function Documentation

#### 10.66.2.1 OS\_chmod() [int32 OS\\_chmod](#) (

```
    const char * path,  
    uint32 access_mode )
```

Changes the permissions of a file.

##### Parameters

in	<i>path</i>	File to change (must not be null)
in	<i>access_mode</i>	Desired access mode - see <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

#### 10.66.2.2 OS\_close() [int32 OS\\_close](#) (

```
osal_id_t filedes )
```

Closes an open file handle.

This closes regular file handles and any other file-like resource, such as network streams or pipes.

#### Parameters

in	<i>filedes</i>	The handle ID to operate on
----	----------------	-----------------------------

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
<i>OS_ERROR</i>	if an unexpected/unhandled error occurs (return value only verified in coverage test)

Referenced by CF\_WrappedClose().

**10.66.2.3 OS\_CloseAllFiles()** `int32 OS_CloseAllFiles ( void )`

Close all open files.

Closes All open files that were opened through the OSAL

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if one or more file close returned an error (return value only verified in coverage test)

**10.66.2.4 OS\_CloseFileByName()** `int32 OS_CloseFileByName ( const char * Filename )`

Close a file by filename.

Allows a file to be closed by name. This will only work if the name passed in is the same name used to open the file.

#### Parameters

in	<i>Filenname</i>	The file to close (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_INVALID</i>	if the file is not found
<i>OS_ERROR</i>	if the file close returned an error (return value only verified in coverage test)
<i>OS_INVALID_POINTER</i>	if the filename argument is NULL

**10.66.2.5 OS\_cp()** `int32 OS_cp (`  
    `const char * src,`  
    `const char * dest )`

Copies a single file from src to dest.

**Note**

The behavior of this API on an open file is not defined at the OSAL level due to dependencies on the underlying OS which may or may not allow the related operation based on a variety of potential configurations. For portability, it is recommended that applications ensure the file is closed prior to removal.

**Parameters**

in	<i>src</i>	The source file to operate on (must not be null)
in	<i>dest</i>	The destination file (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERROR</i>	if the file could not be accessed
<i>OS_INVALID_POINTER</i>	if src or dest are NULL
<i>OS_FS_ERR_PATH_INVALID</i>	if path cannot be parsed
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if the paths given are too long to be stored locally
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the dest name is too long to be stored locally

**10.66.2.6 OS\_FDGetInfo()** `int32 OS_FDGetInfo (`  
    `osal_id_t filedes,`  
    `OS_file_prop_t * fd_prop )`

Obtain information about an open file.

Copies the information of the given file descriptor into a structure passed in

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
out	<i>fd_prop</i>	Storage buffer for file information (must not be null)

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the file descriptor passed in is invalid
<code>OS_INVALID_POINTER</code>	if the fd_prop argument is NULL

```
10.66.2.7 OS_FileAllocate() int32 OS_FileAllocate (
    osal_id_t filedes,
    osal_offset_t offset,
    osal_offset_t len )
```

Pre-allocates space at the given file location.

Instructs the underlying OS/Filesystem to pre-allocate filesystem blocks for the given file location and length. After this, future writes into the same file area will not fail due to lack of space.

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
in	<i>offset</i>	The offset within the file
in	<i>len</i>	The length of space to pre-allocate

**Note**

Some file systems do not implement this capability

**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_ERR_OUTPUT_TOO_LARGE</code>	if this would cause the file to be too large
<code>OS_ERR_OPERATION_NOT_SUPPORTED</code>	if the filesystem does not support this

```
10.66.2.8 OS_FileOpenCheck() int32 OS_FileOpenCheck (
    const char * Filename )
```

Checks to see if a file is open.

This function takes a filename and determines if the file is open. The function will return success if the file is open.

**Parameters**

in	<i>Filename</i>	The file to operate on (must not be null)
----	-----------------	---

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	if the file is open
<i>OS_ERROR</i>	if the file is not open
<i>OS_INVALID_POINTER</i>	if the filename argument is NULL

Referenced by CF\_CFDP\_S\_Init().

**10.66.2.9 OS\_FileTruncate()** `int32 OS_FileTruncate (`  
`osal_id_t filedes,`  
`osal_offset_t len )`

Changes the size of the file.

This changes the size of the file on disk to the specified value. It may either extend or truncate the file.

**Note**

No data is written to the extended file area when a file is grown using this API call. Depending on the file system in use, data blocks may not be allocated at the same time (a sparse file). Data blocks are allocated at the time file data is written into the extended area.

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
in	<i>len</i>	The desired length of the file

**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_ERR_OUTPUT_TOO_LARGE</i>	if this would cause the file to be too large

**10.66.2.10 OS\_lseek()** `int32 OS_lseek (`  
`osal_id_t filedes,`  
`osal_offset_t 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
----	----------------	-----------------------------

**Parameters**

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**

<a href="#">OS_ERR_INVALID_ID</a>	if the file descriptor passed in is invalid
<a href="#">OS_ERROR</a>	if OS call failed (return value only verified in coverage test)

Referenced by CF\_WrappedLseek().

```
10.66.2.11 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

Referenced by CF\_CFDP\_R\_HandleFileRetention(), and CF\_CFDP\_S\_HandleFileRetention().

```
10.66.2.12 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 CF\_WrappedOpenCreate().

```
10.66.2.13 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**

<code>OS_INVALID_POINTER</code>	if buffer is a null pointer
<code>OS_ERR_INVALID_SIZE</code>	if the passed-in size is not valid
<code>OS_ERROR</code>	if OS call failed (return value only verified in coverage test)
<code>OS_ERR_INVALID_ID</code>	if the file descriptor passed in is invalid
<code>0</code>	if at end of file/stream data

Referenced by CF\_WrappedRead().

#### 10.66.2.14 `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**

<code>in</code>	<code>path</code>	The file to operate on (must not be null)
-----------------	-------------------	---

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERROR</code>	if there is no device or the driver returns error
<code>OS_INVALID_POINTER</code>	if path is NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if path is too long to be stored locally
<code>OS_FS_ERR_PATH_INVALID</code>	if path cannot be parsed
<code>OS_FS_ERR_NAME_TOO_LONG</code>	if the name of the file to remove is too long

Referenced by CF\_CFDP\_R\_HandleFileRetention(), and CF\_CFDP\_S\_HandleFileRetention().

```
10.66.2.15 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

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERROR</a>	if the file could not be opened or renamed.
<a href="#">OS_INVALID_POINTER</a>	if old_filename or new_filename 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 new name is too long to be stored locally

```
10.66.2.16 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.
----------------------------	-----------------------

## Return values

<code>OS_INVALID_POINTER</code>	if path or filestats is NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the path is too long to be stored locally
<code>OS_FS_ERR_NAME_TOO_LONG</code>	if the name of the file is too long to be stored
<code>OS_FS_ERR_PATH_INVALID</code>	if path cannot be parsed
<code>OS_ERROR</code>	if the OS call failed

**10.66.2.17 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	<code>filedes</code>	The handle ID to operate on
out	<code>buffer</code>	Storage location for file data (must not be null)
in	<code>nbytes</code>	Maximum number of bytes to read (must not be zero)
in	<code>timeout</code>	Maximum time to wait, in milliseconds, relative to current time (OS_PEND = forever)

## Returns

Byte count on success or appropriate error code, see [OSAL Return Code Defines](#)

## Return values

<code>OS_ERROR_TIMEOUT</code>	if no data became available during timeout period
<code>OS_ERR_INVALID_ID</code>	if the file descriptor passed in is invalid
<code>OS_ERR_INVALID_SIZE</code>	if the passed-in size is not valid
<code>OS_INVALID_POINTER</code>	if the passed-in buffer is not valid
<code>0</code>	if at end of file/stream data

```
10.66.2.18 OS_TimedReadAbs() int32 OS_TimedReadAbs (
    osal_id_t filedes,
    void * buffer,
    size_t nbytes,
    OS_time_t abstime )
```

File/Stream input read with a timeout.

This implements a time-limited read and is primarily intended for use with sockets but may also work with any other stream-like resource that the underlying OS supports, such as pipes or special devices.

If data is immediately available on the file/socket, this will return that data along with the actual number of bytes that were immediately available. It will not block.

If the file position is at the end of file or end of stream data (e.g. if the remote end has closed the connection), then this function will immediately return 0 without blocking for the timeout period.

If no data is immediately available, but the underlying resource/stream is still connected to a peer, this will wait up to the given timeout for additional data to appear. If no data appears within the timeout period, then this returns the [OS\\_ERROR\\_TIMEOUT](#) status code. This allows the caller to differentiate an open (but idle) socket connection from a connection which has been closed by the remote peer.

In all cases this will return successfully as soon as at least 1 byte of actual data is available. It will not attempt to read the entire input buffer.

If an EOF condition occurs prior to timeout, this function returns zero.

#### Parameters

in	<i>filedes</i>	The handle ID to operate on
out	<i>buffer</i>	Storage location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>abstime</i>	Absolute time at which this function should return, if no data is readable

#### Returns

Byte count on success or appropriate error code, see [OSAL Return Code Defines](#)

#### Return values

<a href="#">OS_ERROR_TIMEOUT</a>	if no data became available during timeout period
<a href="#">OS_ERR_INVALID_ID</a>	if the file descriptor passed in is invalid
<a href="#">OS_ERR_INVALID_SIZE</a>	if the passed-in size is not valid
<a href="#">OS_INVALID_POINTER</a>	if the passed-in buffer is not valid
0	if at end of file/stream data

```
10.66.2.19 OS_TimedWrite() int32 OS_TimedWrite (
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes,
    int32 timeout )
```

File/Stream output write with a timeout.

This implements a time-limited write and is primarily intended for use with sockets but may also work with any other stream-like resource that the underlying OS supports.

If output buffer space is immediately available on the file/socket, this will place data into the buffer and return the actual number of bytes that were queued for output. It will not block.

If no output buffer space is immediately available, this will wait up to the given timeout for space to become available. If no space becomes available within the timeout period, then this returns an error code (not zero).

In all cases this will return successfully as soon as at least 1 byte of actual data is output. It will *not* attempt to write the entire output buffer.

If an EOF condition occurs prior to timeout, this function returns zero.

#### Parameters

in	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>timeout</i>	Maximum time to wait, in milliseconds, relative to current time (OS_PEND = forever)

#### Returns

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

#### Return values

<a href="#"><i>OS_ERROR_TIMEOUT</i></a>	if no data became available during timeout period
<a href="#"><i>OS_ERR_INVALID_ID</i></a>	if the file descriptor passed in is invalid
<a href="#"><i>OS_ERR_INVALID_SIZE</i></a>	if the passed-in size is not valid
<a href="#"><i>OS_INVALID_POINTER</i></a>	if the passed-in buffer is not valid
0	if file/stream cannot accept any more data

#### 10.66.2.20 OS\_TimedWriteAbs()

```
int32 OS_TimedWriteAbs (
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes,
    OS_time_t abstime )
```

File/Stream output write with a timeout.

This implements a time-limited write and is primarily intended for use with sockets but may also work with any other stream-like resource that the underlying OS supports.

If output buffer space is immediately available on the file/socket, this will place data into the buffer and return the actual number of bytes that were queued for output. It will not block.

If no output buffer space is immediately available, this will wait up to the given timeout for space to become available. If no space becomes available within the timeout period, then this returns an error code (not zero).

In all cases this will return successfully as soon as at least 1 byte of actual data is output. It will *not* attempt to write the entire output buffer.

If an EOF condition occurs prior to timeout, this function returns zero.

#### Parameters

in	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)
in	<i>abstime</i>	Absolute time at which this function should return, if no data is readable

**Returns**

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

**Return values**

<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
0	if file/stream cannot accept any more data

```
10.66.2.21 OS_write() int32 OS_write (
    osal_id_t filedes,
    const void * buffer,
    size_t nbytes )
```

Write to a file handle.

Writes to a file. copies up to a maximum of nbytes of buffer to the file described in filedes

**Parameters**

in	<i>filedes</i>	The handle ID to operate on
in	<i>buffer</i>	Source location for file data (must not be null)
in	<i>nbytes</i>	Maximum number of bytes to read (must not be zero)

**Note**

All OSAL error codes are negative int32 values. Failure of this call can be checked by testing if the result is less than 0.

**Returns**

A non-negative byte count or appropriate error code, see [OSAL Return Code Defines](#)

**Return values**

<i>OS_INVALID_POINTER</i>	if buffer is NULL
<i>OS_ERR_INVALID_SIZE</i>	if the passed-in size is not valid
<i>OS_ERROR</i>	if OS call failed (return value only verified in coverage test)
<i>OS_ERR_INVALID_ID</i>	if the file descriptor passed in is invalid
0	if file/stream cannot accept any more data

Referenced by CF\_WrappedWrite().

## 10.67 OSAL File System Level APIs

**Functions**

- `int32 OS_FileSysAddFixedMap (osal_id_t *filesys_id, const char *phys_path, const char *virt_path)`

*Create a fixed mapping between an existing directory and a virtual OSAL mount point.*

- [int32 OS\\_mkfs](#) (char \*address, const char \*devname, const char \*volname, size\_t blocksize, [osal\\_blockcount\\_t](#) numblocks)
 

*Makes a file system on the target.*
- [int32 OS\\_mount](#) (const char \*devname, const char \*mountpoint)
 

*Mounts a file system.*
- [int32 OS\\_initfs](#) (char \*address, const char \*devname, const char \*volname, size\_t blocksize, [osal\\_blockcount\\_t](#) numblocks)
 

*Initializes an existing file system.*
- [int32 OS\\_rmfs](#) (const char \*devname)
 

*Removes a file system.*
- [int32 OS\\_unmount](#) (const char \*mountpoint)
 

*Unmounts a mounted file system.*
- [int32 OS\\_FileSysStatVolume](#) (const char \*name, [OS\\_statvfs\\_t](#) \*statbuf)
 

*Obtains information about size and free space in a volume.*
- [int32 OS\\_chkfs](#) (const char \*name, bool repair)
 

*Checks the health of a file system and repairs it if necessary.*
- [int32 OS\\_GetPhysDriveName](#) (char \*PhysDriveName, const char \*MountPoint)
 

*Obtains the physical drive name associated with a mount point.*
- [int32 OS\\_TranslatePath](#) (const char \*VirtualPath, char \*LocalPath)
 

*Translates an OSAL Virtual file system path to a host Local path.*
- [int32 OS\\_GetFsInfo](#) ([os\\_fsinfo\\_t](#) \*filesys\_info)
 

*Returns information about the file system.*

### 10.67.1 Detailed Description

### 10.67.2 Function Documentation

**10.67.2.1 OS\_chkfs()** [int32 OS\\_chkfs](#) (

```
    const char * name,
    bool repair )
```

Checks the health of a file system and repairs it if necessary.

Checks the drives for inconsistencies and optionally also repairs it

#### Note

not all operating systems implement this function. If the underlying OS does not provide a facility to check the volume, then OS\_ERR\_NOT\_IMPLEMENTED will be returned.

#### Parameters

in	<i>name</i>	The device/path to operate on (must not be null)
in	<i>repair</i>	Whether to also repair inconsistencies

#### Returns

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

### Return values

<code>OS_SUCCESS</code>	Successful execution. (return value only verified in coverage test)
<code>OS_INVALID_POINTER</code>	Name is NULL
<code>OS_ERR_NOT_IMPLEMENTED</code>	Not implemented.
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the name is too long
<code>OS_ERROR</code>	Failed execution. (return value only verified in coverage test)

### 10.67.2.2 OS\_FileSysAddFixedMap()

```
int32 OS_FileSysAddFixedMap (
    osal_id_t * filesys_id,
    const char * phys_path,
    const char * virt_path )
```

Create a fixed mapping between an existing directory and a virtual OSAL mount point.

This mimics the behavior of a "FS\_BASED" entry in the VolumeTable but is registered at runtime. It is intended to be called by the PSP/BSP prior to starting the application.

#### Note

OSAL virtual mount points are required to be a single, non-empty top-level directory name. Virtual path names always follow the form /<virt\_mount\_point>/<relative\_path>/<file>. Only the relative path may be omitted/empty (i.e. /<virt\_mount\_point>/<file>) but the virtual mount point must be present and not an empty string. In particular this means it is not possible to directly refer to files in the "root" of the native file system from OSAL. However it is possible to create a virtual map to the root, such as by calling:

```
OS_FileSysAddFixedMap (&fs_id, "/", "/root");
```

### Parameters

out	<code>filesys_id</code>	A buffer to store the ID of the file system mapping (must not be null)
in	<code>phys_path</code>	The native system directory (an existing mount point) (must not be null)
in	<code>virt_path</code>	The virtual mount point of this filesystem (must not be null)

### Returns

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

### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the overall phys_path is too long
<code>OS_ERR_NAME_TOO_LONG</code>	if the phys_path basename (filesystem name) is too long
<code>OS_INVALID_POINTER</code>	if any argument is NULL

### 10.67.2.3 OS\_FileSysStatVolume()

```
int32 OS_FileSysStatVolume (
    const char * name,
    OS_statvfs_t * statbuf )
```

Obtains information about size and free space in a volume.

Populates the supplied `OS_statvfs_t` structure, which includes the block size and total/free blocks in a file system volume. This replaces two older OSAL calls:

`OS_fsBlocksFree()` is determined by reading the `blocks_free` output struct member `OS_fsBytesFree()` is determined by multiplying `blocks_free` by the `block_size` member

#### Parameters

in	<code>name</code>	The device/path to operate on (must not be null)
out	<code>statbuf</code>	Output structure to populate (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if name or statbuf is NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the name is too long
<code>OS_ERROR</code>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

#### 10.67.2.4 `OS_FS_GetPhysDriveName()`

```
int32 OS_FS_GetPhysDriveName (
    char * PhysDriveName,
    const char * MountPoint )
```

Obtains the physical drive name associated with a mount point.

Returns the name of the physical volume associated with the drive, when given the OSAL mount point of the drive

#### Parameters

out	<code>PhysDriveName</code>	Buffer to store physical drive name (must not be null)
in	<code>MountPoint</code>	OSAL mount point (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if either parameter is NULL
<code>OS_ERR_NAME_NOT_FOUND</code>	if the MountPoint is not mounted in OSAL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the MountPoint is too long

#### 10.67.2.5 `OS_GetFsInfo()`

```
os_fsinfo_t * filesys_info )
```

Returns information about the file system.

Returns information about the file system in an `os_fsinfo_t`. This includes the number of open files and file systems

#### Parameters

out	<code>filesys_info</code>	Buffer to store filesystem information (must not be null)
-----	---------------------------	---

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if <code>filesys_info</code> is NULL

```
10.67.2.6 OS_initfs() int32 OS_initfs (
    char * address,
    const char * devname,
    const char * volname,
    size_t blocksize,
    osal_blockcount_t numblocks )
```

Initializes an existing file system.

Initializes a file system on the target.

#### Note

The "volname" parameter of RAM disks should always begin with the string "RAM", e.g. "RAMDISK" or "RAM0", "RAM1", etc if multiple devices are created. The underlying implementation uses this to select the correct filesystem type/format, and this may also be used to differentiate between RAM disks and real physical disks.

#### Parameters

in	<code>address</code>	The address at which to start the new disk. If <code>address == 0</code> , then space will be allocated by the OS
in	<code>devname</code>	The underlying kernel device to use, if applicable. (must not be null)
in	<code>volname</code>	The name of the volume (see note) (must not be null)
in	<code>blocksize</code>	The size of a single block on the drive
in	<code>numblocks</code>	The number of blocks to allocate for the drive

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if <code>devname</code> or <code>volname</code> are NULL

## Return values

<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the name is too long
<code>OS_FS_ERR_DEVICE_NOT_FREE</code>	if the volume table is full
<code>OS_FS_ERR_DRIVE_NOT_CREATED</code>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

**10.67.2.7 OS\_mkfs()** `int32 OS_mkfs(`

```
    char * address,
    const char * devname,
    const char * volname,
    size_t blocksize,
    osal_blockcount_t numblocks )
```

Makes a file system on the target.

Makes a file system on the target. Highly dependent on underlying OS and dependent on OS volume table definition.

## Note

The "volname" parameter of RAM disks should always begin with the string "RAM", e.g. "RAMDISK" or "RAM0", "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

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if devname or volname is NULL
<code>OS_FS_ERR_PATH_TOO_LONG</code>	if the overall devname or volname is too long
<code>OS_FS_ERR_DEVICE_NOT_FREE</code>	if the volume table is full
<code>OS_FS_ERR_DRIVE_NOT_CREATED</code>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)

**10.67.2.8 OS\_mount()** `int32 OS_mount(`

```
    const char * devname,
    const char * mountpoint )
```

Mounts a file system.

Mounts a file system / block device at the given mount point.

#### Parameters

in	<i>devname</i>	The name of the drive to mount. devname is the same from <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)

**10.67.2.9 OS\_rmfs()** `int32 OS_rmfs (`  
    `const char * devname )`

Removes a file system.

This function will remove or un-map the target file system. Note that this is not the same as un-mounting the file system.

#### Parameters

in	<i>devname</i>	The name of the "generic" drive (must not be null)
----	----------------	--

#### Returns

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

#### Return values

<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)

**10.67.2.10 OS\_TranslatePath()** `int32 OS_TranslatePath (`  
    `const char * VirtualPath,`

```
char * LocalPath )
```

Translates an OSAL Virtual file system path to a host Local path.

Translates a virtual path to an actual system path name

#### Note

The buffer provided in the LocalPath argument is required to be at least OS\_MAX\_PATH\_LEN characters in length.

#### Parameters

in	<i>VirtualPath</i>	OSAL virtual path name (must not be null)
out	<i>LocalPath</i>	Buffer to store native/translated path name (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if either parameter is NULL
<i>OS_FS_ERR_NAME_TOO_LONG</i>	if the filename component is too long
<i>OS_FS_ERR_PATH_INVALID</i>	if either parameter cannot be interpreted as a path
<i>OS_FS_ERR_PATH_TOO_LONG</i>	if either input or output pathnames are too long

### 10.67.2.11 OS\_unmount() `int32 OS_unmount( const char * mountpoint )`

Unmounts a mounted file system.

This function will unmount a drive from the file system and make all open file descriptors useless.

#### Note

Any open file descriptors referencing this file system should be closed prior to unmounting a drive

#### Parameters

in	<i>mountpoint</i>	The mount point to remove from <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
<i>OS_ERR_NAME_NOT_FOUND</i>	if the mountpoint is not mounted in OSAL

**Return values**

<code>OS_ERROR</code>	if an unexpected/unhandled OS error occurs (return value only verified in coverage test)
-----------------------	--

## 10.68 OSAL Heap APIs

**Functions**

- `int32 OS_HeapGetInfo (OS_heap_prop_t *heap_prop)`

*Return current info on the heap.*

### 10.68.1 Detailed Description

### 10.68.2 Function Documentation

**10.68.2.1 OS\_HeapGetInfo()** `int32 OS_HeapGetInfo (`  
`OS_heap_prop_t * heap_prop )`

Return current info on the heap.

**Parameters**

<code>out</code>	<code>heap_prop</code>	Storage buffer for heap info
------------------	------------------------	------------------------------

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if the heap_prop argument is NULL

## 10.69 OSAL Object Type Defines

**Macros**

- `#define OS_OBJECT_TYPE_UNDEFINED 0x00`  
*Object type undefined.*
- `#define OS_OBJECT_TYPE_OS_TASK 0x01`  
*Object task type.*
- `#define OS_OBJECT_TYPE_OS_QUEUE 0x02`  
*Object queue type.*
- `#define OS_OBJECT_TYPE_OS_COUNTSEM 0x03`  
*Object counting semaphore type.*
- `#define OS_OBJECT_TYPE_OS_BINSEM 0x04`  
*Object binary semaphore type.*

- #define **OS\_OBJECT\_TYPE\_OS\_MUTEX** 0x05  
*Object mutex type.*
- #define **OS\_OBJECT\_TYPE\_OS\_STREAM** 0x06  
*Object stream type.*
- #define **OS\_OBJECT\_TYPE\_OS\_DIR** 0x07  
*Object directory type.*
- #define **OS\_OBJECT\_TYPE\_OS\_TIMEBASE** 0x08  
*Object timebase type.*
- #define **OS\_OBJECT\_TYPE\_OS\_TIMECB** 0x09  
*Object timer callback type.*
- #define **OS\_OBJECT\_TYPE\_OS\_MODULE** 0x0A  
*Object module type.*
- #define **OS\_OBJECT\_TYPE\_OS\_FILESYS** 0x0B  
*Object file system type.*
- #define **OS\_OBJECT\_TYPE\_OS\_CONSOLE** 0x0C  
*Object console type.*
- #define **OS\_OBJECT\_TYPE\_OS\_CONDVAR** 0x0D  
*Object condition variable type.*
- #define **OS\_OBJECT\_TYPE\_OS\_RWLOCK** 0x0E  
*Object readers-writer lock type.*
- #define **OS\_OBJECT\_TYPE\_USER** 0x10  
*Object user type.*

#### 10.69.1 Detailed Description

#### 10.69.2 Macro Definition Documentation

**10.69.2.1 OS\_OBJECT\_TYPE\_OS\_BINSEM** #define OS\_OBJECT\_TYPE\_OS\_BINSEM 0x04  
Object binary semaphore type.  
Definition at line 42 of file osapi-idmap.h.

**10.69.2.2 OS\_OBJECT\_TYPE\_OS\_CONDVAR** #define OS\_OBJECT\_TYPE\_OS\_CONDVAR 0x0D  
Object condition variable type.  
Definition at line 51 of file osapi-idmap.h.

**10.69.2.3 OS\_OBJECT\_TYPE\_OS\_CONSOLE** #define OS\_OBJECT\_TYPE\_OS\_CONSOLE 0x0C  
Object console type.  
Definition at line 50 of file osapi-idmap.h.

**10.69.2.4 OS\_OBJECT\_TYPE\_OS\_COUNTSEM** #define OS\_OBJECT\_TYPE\_OS\_COUNTSEM 0x03  
Object counting semaphore type.  
Definition at line 41 of file osapi-idmap.h.

**10.69.2.5 OS\_OBJECT\_TYPE\_OS\_DIR** #define OS\_OBJECT\_TYPE\_OS\_DIR 0x07  
Object directory type.  
Definition at line 45 of file osapi-idmap.h.

**10.69.2.6 OS\_OBJECT\_TYPE\_OS\_FILESYS** #define OS\_OBJECT\_TYPE\_OS\_FILESYS 0x0B  
Object file system type.  
Definition at line 49 of file osapi-idmap.h.

**10.69.2.7 OS\_OBJECT\_TYPE\_OS\_MODULE** #define OS\_OBJECT\_TYPE\_OS\_MODULE 0x0A  
Object module type.  
Definition at line 48 of file osapi-idmap.h.

**10.69.2.8 OS\_OBJECT\_TYPE\_OS\_MUTEX** #define OS\_OBJECT\_TYPE\_OS\_MUTEX 0x05  
Object mutex type.  
Definition at line 43 of file osapi-idmap.h.

**10.69.2.9 OS\_OBJECT\_TYPE\_OS\_QUEUE** #define OS\_OBJECT\_TYPE\_OS\_QUEUE 0x02  
Object queue type.  
Definition at line 40 of file osapi-idmap.h.

**10.69.2.10 OS\_OBJECT\_TYPE\_OS\_RWLOCK** #define OS\_OBJECT\_TYPE\_OS\_RWLOCK 0x0E  
Object readers-writer lock type.  
Definition at line 52 of file osapi-idmap.h.

**10.69.2.11 OS\_OBJECT\_TYPE\_OS\_STREAM** #define OS\_OBJECT\_TYPE\_OS\_STREAM 0x06  
Object stream type.  
Definition at line 44 of file osapi-idmap.h.

**10.69.2.12 OS\_OBJECT\_TYPE\_OS\_TASK** #define OS\_OBJECT\_TYPE\_OS\_TASK 0x01  
Object task type.  
Definition at line 39 of file osapi-idmap.h.

**10.69.2.13 OS\_OBJECT\_TYPE\_OS\_TIMEBASE** #define OS\_OBJECT\_TYPE\_OS\_TIMEBASE 0x08  
Object timebase type.  
Definition at line 46 of file osapi-idmap.h.

**10.69.2.14 OS\_OBJECT\_TYPE\_OS\_TIMECB** #define OS\_OBJECT\_TYPE\_OS\_TIMECB 0x09  
Object timer callback type.  
Definition at line 47 of file osapi-idmap.h.

**10.69.2.15 OS\_OBJECT\_TYPE\_UNDEFINED** #define OS\_OBJECT\_TYPE\_UNDEFINED 0x00  
Object type undefined.  
Definition at line 38 of file osapi-idmap.h.

**10.69.2.16 OS\_OBJECT\_TYPE\_USER** #define OS\_OBJECT\_TYPE\_USER 0x10  
Object user type.  
Definition at line 53 of file osapi-idmap.h.

## 10.70 OSAL Object ID Utility APIs

### Functions

- static unsigned long **OS\_ObjectIdToInteger** (osal\_id\_t object\_id)  
*Obtain an integer value corresponding to an object ID.*
- static osal\_id\_t **OS\_ObjectIdFromInteger** (unsigned long value)  
*Obtain an osal ID corresponding to an integer value.*
- static bool **OS\_ObjectIdEqual** (osal\_id\_t object\_id1, osal\_id\_t object\_id2)  
*Check two OSAL object ID values for equality.*
- static bool **OS\_ObjectIdDefined** (osal\_id\_t object\_id)  
*Check if an object ID is defined.*
- int32 **OS\_GetResourceName** (osal\_id\_t object\_id, char \*buffer, size\_t buffer\_size)  
*Obtain the name of an object given an arbitrary object ID.*
- osal\_objtype\_t **OS\_IdentifyObject** (osal\_id\_t object\_id)  
*Obtain the type of an object given an arbitrary object ID.*
- int32 **OS\_ConvertToArrayIndex** (osal\_id\_t object\_id, osal\_index\_t \*ArrayIndex)  
*Converts an abstract ID into a number suitable for use as an array index.*
- int32 **OS\_ObjectIdToArrayIndex** (osal\_objtype\_t idtype, osal\_id\_t object\_id, osal\_index\_t \*ArrayIndex)  
*Converts an abstract ID into a number suitable for use as an array index.*
- void **OS\_ForEachObject** (osal\_id\_t creator\_id, OS\_ArgCallback\_t callback\_ptr, void \*callback\_arg)  
*call the supplied callback function for all valid object IDs*
- void **OS\_ForEachObjectType** (osal\_objtype\_t objtype, osal\_id\_t creator\_id, OS\_ArgCallback\_t callback\_ptr, void \*callback\_arg)  
*call the supplied callback function for valid object IDs of a specific type*

### 10.70.1 Detailed Description

### 10.70.2 Function Documentation

**10.70.2.1 OS\_ConvertToArrayIndex()** int32 OS\_ConvertToArrayIndex (

```
    osal_id_t object_id,
    osal_index_t * ArrayIndex )
```

Converts an abstract ID into a number suitable for use as an array index.  
This will return a unique zero-based integer number in the range of [0,MAX) for any valid object ID. This may be used by application code as an array index for indexing into local tables.

**Note**

This does NOT verify the validity of the ID, that is left to the caller. This is only the conversion logic.

This routine accepts any object type, and returns a value based on the maximum number of objects for that type. This is equivalent to invoking [OS\\_ObjectIdToArrayIndex\(\)](#) with the idtype set to OS\_OBJECT\_TYPE\_UNDEFINED.

**See also**

[OS\\_ObjectIdToArrayIndex](#)

**Parameters**

in	<i>object_id</i>	The object ID to operate on
out	* <i>ArrayIndex</i>	The Index to return (must not be null)

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the object_id argument is not valid
<a href="#">OS_INVALID_POINTER</a>	if the ArrayIndex is NULL

**10.70.2.2 OS\_ForEachObject()** `void OS_ForEachObject (`

```
    osal_id_t creator_id,
    OS_ArgCallback_t callback_ptr,
    void * callback_arg )
```

call the supplied callback function for all valid object IDs

Loops through all defined OSAL objects of all types and calls callback\_ptr on each one If creator\_id is nonzero then only objects with matching creator id are processed.

**Parameters**

in	<i>creator_id</i>	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	<i>callback_ptr</i>	Function to invoke for each matching object ID
in	<i>callback_arg</i>	Opaque Argument to pass to callback function (may be NULL)

**10.70.2.3 OS\_ForEachObjectType()** `void OS_ForEachObjectType (`

```
    osal_objtype_t objtype,
    osal_id_t creator_id,
    OS_ArgCallback_t callback_ptr,
    void * callback_arg )
```

call the supplied callback function for valid object IDs of a specific type

Loops through all defined OSAL objects of a specific type and calls callback\_ptr on each one If creator\_id is nonzero

then only objects with matching creator id are processed.

#### Parameters

in	<i>objtype</i>	The type of objects to iterate
in	<i>creator_id</i>	Filter objects to those created by a specific task This may be passed as OS_OBJECT_CREATOR_ANY to return all objects
in	<i>callback_ptr</i>	Function to invoke for each matching object ID
in	<i>callback_arg</i>	Opaque Argument to pass to callback function (may be NULL)

```
10.70.2.4 OS_GetResourceName() int32 OS_GetResourceName (
    osal_id_t object_id,
    char * buffer,
    size_t buffer_size )
```

Obtain the name of an object given an arbitrary object ID.

All OSAL resources generally have a name associated with them. This allows application code to retrieve the name of any valid OSAL object ID.

#### Parameters

in	<i>object_id</i>	The object ID to operate on
out	<i>buffer</i>	Buffer in which to store the name (must not be null)
in	<i>buffer_size</i>	Size of the output storage buffer (must not be zero)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the passed-in ID is not a valid OSAL ID
<a href="#">OS_INVALID_POINTER</a>	if the passed-in buffer is invalid
<a href="#">OS_ERR_NAME_TOO_LONG</a>	if the name will not fit in the buffer provided

```
10.70.2.5 OS_IdentifyObject() osal_objtype_t OS_IdentifyObject (
    osal_id_t object_id )
```

Obtain the type of an object given an arbitrary object ID.

Given an arbitrary object ID, get the type of the object

#### Parameters

in	<i>object_id</i>	The object ID to operate on
----	------------------	-----------------------------

**Returns**

The object type portion of the object\_id, see [OSAL Object Type Defines](#) for expected values

**10.70.2.6 OS\_ObjectIdDefined()** static bool OS\_ObjectIdDefined (   
   osal\_id\_t object\_id ) [inline], [static]

Check if an object ID is defined.

The OSAL ID values should be treated as abstract values by applications, and not directly manipulated using standard C operators.

This returns false if the ID is NOT a defined resource (i.e. free/empty/invalid).

**Note**

OS\_ObjectIdDefined(OS\_OBJECT\_ID\_UNDEFINED) is always guaranteed to be false.

**Parameters**

in	<i>object_id</i>	The first object ID
----	------------------	---------------------

Definition at line 151 of file osapi-idmap.h.

References OS\_ObjectIdToInteger().

Referenced by CF\_CFDP\_CloseFiles(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_RecycleTransaction(), and CF\_CFDP\_S\_Init().

**10.70.2.7 OS\_ObjectIdEqual()** static bool OS\_ObjectIdEqual (   
   osal\_id\_t object\_id1,   
   osal\_id\_t object\_id2 ) [inline], [static]

Check two OSAL object ID values for equality.

The OSAL ID values should be treated as abstract values by applications, and not directly manipulated using standard C operators.

This checks two values for equality, replacing the "==" operator.

**Parameters**

in	<i>object_id1</i>	The first object ID
in	<i>object_id2</i>	The second object ID

**Returns**

true if the object IDs are equal

Definition at line 130 of file osapi-idmap.h.

References OS\_ObjectIdToInteger().

**10.70.2.8 OS\_ObjectIdFromInteger()** static osal\_id\_t OS\_ObjectIdFromInteger (   
   unsigned long value ) [inline], [static]

Obtain an osal ID corresponding to an integer value.

Provides the inverse of [OS\\_ObjectIdToInteger\(\)](#). Reconstitutes the original osal\_id\_t type from an integer representation.

**Parameters**

in	<i>value</i>	The integer representation of an OSAL ID
----	--------------	--

**Returns**

The ID value converted to an osal\_id\_t

Definition at line 103 of file osapi-idmap.h.

**10.70.2.9 OS\_ObjectIdToArrayIndex()** `int32 OS_ObjectIdToArrayIndex (`  
`osal_objtype_t idtype,`  
`osal_id_t object_id,`  
`osal_index_t * ArrayIndex )`

Converts an abstract ID into a number suitable for use as an array index.

This will return a unique zero-based integer number in the range of [0,MAX) for any valid object ID. This may be used by application code as an array index for indexing into local tables.

This routine operates on a specific object type, and returns a value based on the maximum number of objects for that type.

If the idtype is passed as `OS_OBJECT_TYPE_UNDEFINED`, then object type verification is skipped and any object ID will be accepted and converted to an index. In this mode, the range of the output depends on the actual passed-in object type.

If the idtype is passed as any other value, the passed-in ID value is first confirmed to be the correct type. This check will guarantee that the output is within an expected range; for instance, if the type is passed as `OS_OBJECT_TYPE_OS_TASK`, then the output index is guaranteed to be between 0 and `OS_MAX_TASKS`-1 after successful conversion.

**Parameters**

in	<i>idtype</i>	The object type to convert
in	<i>object_id</i>	The object ID to operate on
out	<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

**10.70.2.10 OS\_ObjectIdToInteger()** `static unsigned long OS_ObjectIdToInteger (`  
`osal_id_t object_id ) [inline], [static]`

Obtain an integer value corresponding to an object ID.

Obtains an integer representation of an object id, generally for the purpose of printing to the console or system logs. The returned value is of the type "unsigned long" for direct use with printf-style functions. It is recommended to use the "%lx" conversion specifier as the hexadecimal encoding clearly delineates the internal fields.

**Note**

This provides the raw integer value and is *not* suitable for use as an array index, as the result is not zero-based. See the [OS\\_ConvertToArrayIndex\(\)](#) to obtain a zero-based index value.

**Parameters**

in	<i>object_id</i>	The object ID
----	------------------	---------------

**Returns**

integer value representation of object ID

Definition at line 81 of file osapi-idmap.h.

Referenced by CF\_CFDP\_RecycleTransaction(), CF\_WrappedClose(), OS\_ObjectIdDefined(), and OS\_ObjectIdEqual().

## 10.71 OSAL Dynamic Loader and Symbol APIs

**Functions**

- [int32 OS\\_SymbolLookup \(cpuaddr \\*symbol\\_address, const char \\*symbol\\_name\)](#)  
*Find the Address of a Symbol.*
- [int32 OS\\_ModuleSymbolLookup \(osal\\_id\\_t module\\_id, cpuaddr \\*symbol\\_address, const char \\*symbol\\_name\)](#)  
*Find the Address of a Symbol within a module.*
- [int32 OS\\_SymbolTableDump \(const char \\*filename, size\\_t size\\_limit\)](#)  
*Dumps the system symbol table to a file.*
- [int32 OS\\_ModuleLoad \(osal\\_id\\_t \\*module\\_id, const char \\*module\\_name, const char \\*filename, uint32 flags\)](#)  
*Loads an object file.*
- [int32 OS\\_ModuleUnload \(osal\\_id\\_t module\\_id\)](#)  
*Unloads the module file.*
- [int32 OS\\_ModuleInfo \(osal\\_id\\_t module\\_id, OS\\_module\\_prop\\_t \\*module\\_info\)](#)  
*Obtain information about a module.*

### 10.71.1 Detailed Description

### 10.71.2 Function Documentation

**10.71.2.1 OS\_ModuleInfo()** `int32 OS_ModuleInfo (`  
    `osal_id_t module_id,`  
    `OS_module_prop_t * module_info )`

Obtain information about a module.

Returns information about the loadable module

**Parameters**

in	<i>module_id</i>	OSAL ID of the previously the loaded module
out	<i>module_info</i>	Buffer to store module information (must not be null)

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the module id invalid
<code>OS_INVALID_POINTER</code>	if the pointer to the ModuleInfo structure is invalid
<code>OS_ERROR</code>	if an other/unspecified error occurs (return value only verified in coverage test)

```
10.71.2.2 OS_ModuleLoad() int32 OS_ModuleLoad (
    osal_id_t * module_id,
    const char * module_name,
    const char * filename,
    uint32 flags )
```

Loads an object file.

Loads an object file into the running operating system

The "flags" parameter may influence how the loaded module symbols are made available for use in the application. See [OS\\_MODULE\\_FLAG\\_LOCAL\\_SYMBOLS](#) and [OS\\_MODULE\\_FLAG\\_GLOBAL\\_SYMBOLS](#) for descriptions.

**Parameters**

out	<i>module_id</i>	Non-zero OSAL ID corresponding to the loaded module
in	<i>module_name</i>	Name of module (must not be null)
in	<i>filename</i>	File containing the object code to load (must not be null)
in	<i>flags</i>	Options for the loaded module

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if one of the parameters is NULL
<code>OS_ERR_NO_FREE_IDS</code>	if the module table is full
<code>OS_ERR_NAME_TAKEN</code>	if the name is in use
<code>OS_ERR_NAME_TOO_LONG</code>	if the module_name is too long
<code>OS_FS_ERR_PATH_INVALID</code>	if the filename argument is not valid
<code>OS_ERROR</code>	if an other/unspecified error occurs (return value only verified in coverage test)

```
10.71.2.3 OS_ModuleSymbolLookup() int32 OS_ModuleSymbolLookup (
    osal_id_t module_id,
    cpuaddr * symbol_address,
    const char * symbol_name )
```

Find the Address of a Symbol within a module.

This is similar to [OS\\_SymbolLookup\(\)](#) but for a specific module ID. This should be used to look up a symbol in a module that has been loaded with the [OS\\_MODULE\\_FLAG\\_LOCAL\\_SYMBOLS](#) flag.

#### Parameters

in	<i>module_id</i>	Module ID that should contain the symbol
out	<i>symbol_address</i>	Set to the address of the symbol (must not be null)
in	<i>symbol_name</i>	Name of the symbol to look up (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERROR</a>	if the symbol could not be found
<a href="#">OS_INVALID_POINTER</a>	if one of the pointers passed in are NULL

**10.71.2.4 OS\_ModuleUnload()** `int32 OS_ModuleUnload ( osal_id_t module_id )`

Unloads the module file.

Unloads the module file from the running operating system

#### Parameters

in	<i>module_id</i>	OSAL ID of the previously the loaded module
----	------------------	---

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the module id invalid
<a href="#">OS_ERROR</a>	if an other/unspecified error occurs (return value only verified in coverage test)

**10.71.2.5 OS\_SymbolLookup()** `int32 OS_SymbolLookup ( cpuaddr * symbol_address, const char * symbol_name )`

Find the Address of a Symbol.

This calls to the OS dynamic symbol lookup implementation, and/or checks a static symbol table for a matching symbol name.

The static table is intended to support embedded targets that do not have module loading capability or have it disabled.

#### Parameters

out	<i>symbol_address</i>	Set to the address of the symbol (must not be null)
in	<i>symbol_name</i>	Name of the symbol to look up (must not be null)

#### Returns

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

#### Return values

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_ERROR</i></a>	if the symbol could not be found
<a href="#"><i>OS_INVALID_POINTER</i></a>	if one of the pointers passed in are NULL

**10.71.2.6 OS\_SymbolTableDump()** `int32 OS_SymbolTableDump (`  
 `const char * filename,`  
 `size_t size_limit )`

Dumps the system symbol table to a file.

Dumps the system symbol table to the specified filename

#### Note

Not all RTOS implementations support this API. If the underlying module subsystem does not provide a facility to iterate through the symbol table, then the [\*OS\\_ERR\\_NOT\\_IMPLEMENTED\*](#) status code is returned.

#### Parameters

in	<i>filename</i>	File to write to (must not be null)
in	<i>size_limit</i>	Maximum number of bytes to write

#### Returns

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

#### Return values

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_ERR_NOT_IMPLEMENTED</i></a>	Not implemented.
<a href="#"><i>OS_INVALID_POINTER</i></a>	if the filename argument is NULL
<a href="#"><i>OS_FS_ERR_PATH_INVALID</i></a>	if the filename argument is not valid
<a href="#"><i>OS_ERR_OUTPUT_TOO_LARGE</i></a>	if the size_limit was reached before completing all symbols (return value only verified in coverage test)
<a href="#"><i>OS_ERROR</i></a>	if an other/unspecified error occurs (return value only verified in coverage test)

## 10.72 OSAL Mutex APIs

### Functions

- `int32 OS_MutSemCreate (osal_id_t *sem_id, const char *sem_name, uint32 options)`  
*Creates a mutex semaphore.*
- `int32 OS_MutSemGive (osal_id_t sem_id)`  
*Releases the mutex object referenced by sem\_id.*
- `int32 OS_MutSemTake (osal_id_t sem_id)`  
*Acquire the mutex object referenced by sem\_id.*
- `int32 OS_MutSemDelete (osal_id_t sem_id)`  
*Deletes the specified Mutex Semaphore.*
- `int32 OS_MutSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`  
*Find an existing mutex ID by name.*
- `int32 OS_MutSemGetInfo (osal_id_t sem_id, OS_mut_sem_prop_t *mut_prop)`  
*Fill a property object buffer with details regarding the resource.*

#### 10.72.1 Detailed Description

#### 10.72.2 Function Documentation

##### 10.72.2.1 OS\_MutSemCreate() `int32 OS_MutSemCreate (`

```
    osal_id_t * sem_id,
    const char * sem_name,
    uint32 options )
```

Creates a mutex semaphore.

Mutex semaphores are always created in the unlocked (full) state.

#### Parameters

<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>options</code>	reserved for future use. Should be passed as 0.

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if sem_id or sem_name are NULL
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NO_FREE_IDS</code>	if there are no more free mutex IDs
<code>OS_ERR_NAME_TAKEN</code>	if there is already a mutex with the same name
<code>OS_SEM_FAILURE</code>	if the OS call failed (return value only verified in coverage test)

**10.72.2.2 OS\_MutSemDelete()** `int32 OS_MutSemDelete ( osal_id_t sem_id )`

Deletes the specified Mutex Semaphore.

Delete the semaphore. This also frees the respective sem\_id such that it can be used again when another is created.

#### Parameters

in	<code>sem_id</code>	The object ID to delete
----	---------------------	-------------------------

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid mutex
<code>OS_SEM_FAILURE</code>	if an unspecified error occurs (return value only verified in coverage test)

**10.72.2.3 OS\_MutSemGetIdByName()** `int32 OS_MutSemGetIdByName (`

```
    osal_id_t * sem_id,
    const char * sem_name )
```

Find an existing mutex ID by name.

This function tries to find a mutex sem Id given the name of a mut\_sem. The id is returned through sem\_id

#### Parameters

out	<code>sem_id</code>	will be set to the ID of the existing resource
in	<code>sem_name</code>	the name of the existing resource to find (must not be null)

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	is semid or sem_name are NULL pointers
<code>OS_ERR_NAME_TOO_LONG</code>	name length including null terminator greater than <code>OS_MAX_API_NAME</code>
<code>OS_ERR_NAME_NOT_FOUND</code>	if the name was not found in the table

**10.72.2.4 OS\_MutSemGetInfo()** `int32 OS_MutSemGetInfo (`

```
    osal_id_t sem_id,
    OS_mut_sem_prop_t * mut_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info( name and creator) about the specified mutex semaphore.

#### Parameters

in	<i>sem_id</i>	The object ID to operate on
out	<i>mut_prop</i>	The property object buffer to fill (must not be null)

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid semaphore
<i>OS_INVALID_POINTER</i>	if the mut_prop pointer is null

### 10.72.2.5 OS\_MutSemGive() `int32 OS_MutSemGive ( osal_id_t sem_id )`

Releases the mutex object referenced by *sem\_id*.

If there are threads blocked on the mutex object referenced by mutex when this function is called, resulting in the mutex becoming available, the scheduling policy shall determine which thread shall acquire the mutex.

#### Parameters

in	<i>sem_id</i>	The object ID to operate on
----	---------------	-----------------------------

#### Returns

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

#### Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid mutex
<i>OS_SEM_FAILURE</i>	if an unspecified error occurs (return value only verified in coverage test)

### 10.72.2.6 OS\_MutSemTake() `int32 OS_MutSemTake ( osal_id_t sem_id )`

Acquire the mutex object referenced by *sem\_id*.

If the mutex is already locked, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by mutex in the locked state with the calling thread as its owner.

**Parameters**

in	<code>sem-&gt; _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>	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)

## 10.73 OSAL Network ID APIs

**Functions**

- `int32 OS_NetworkGetID (void)`  
*Gets the network ID of the local machine.*
- `int32 OS_NetworkGetHostName (char *host_name, size_t name_len)`  
*Gets the local machine network host name.*

### 10.73.1 Detailed Description

Provides some basic methods to query a network host name and ID

### 10.73.2 Function Documentation

**10.73.2.1 OS\_NetworkGetHostName()** `int32 OS_NetworkGetHostName (`  
`char * host_name,`  
`size_t name_len )`

Gets the local machine network host name.

If configured in the underlying network stack, this function retrieves the local hostname of the system.

**Parameters**

out	<code>host_name</code>	Buffer to hold name information (must not be null)
in	<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.
-------------------------	-----------------------

**Return values**

<code>OS_ERR_INVALID_SIZE</code>	if the name_len is zero
<code>OS_INVALID_POINTER</code>	if the host_name is NULL

**10.73.2.2 OS\_NetworkGetID()** `int32 OS_NetworkGetID (`  
    `void )`

Gets the network ID of the local machine.

The ID is an implementation-defined value and may not be consistent in meaning across different platform types.

**Note**

This API may be removed in a future version of OSAL due to inconsistencies between platforms.

**Returns**

The ID or fixed value of -1 if the host id could not be found. Note it is not possible to differentiate between error codes and valid network IDs here. It is assumed, however, that -1 is never a valid ID.

## 10.74 OSAL Printf APIs

**Functions**

- void `OS_printf` (const char \*string,...) `OS_PRINTF(1`  
*Abstraction for the system printf() call.*
- void `OS_printf_disable` (void)  
*This function disables the output from OS\_printf.*
- void `OS_printf_enable` (void)  
*This function enables the output from OS\_printf.*

### 10.74.1 Detailed Description

### 10.74.2 Function Documentation

**10.74.2.1 OS\_printf()** `void OS_printf (`  
    `const char * string,`  
    `...`  
)

Abstraction for the system printf() call.

This function abstracts out the printf type statements. This is useful for using OS- specific thots that will allow non-polled print statements for the real time systems.

Operates in a manner similar to the printf() call defined by the standard C library and takes all the parameters and formatting options of printf. This abstraction may implement additional buffering, if necessary, to improve the real-time performance of the call.

Strings (including terminator) longer than `OS_BUFFER_SIZE` will be truncated.

The output of this routine also may be dynamically enabled or disabled by the `OS_printf_enable()` and `OS_printf_disable()` calls, respectively.

**Parameters**

<code>in</code>	<code>string</code>	Format string, followed by additional arguments
-----------------	---------------------	---

```
10.74.2.2 OS_printf_disable() void void OS_printf_disable (
    void )
```

This function disables the output from OS\_printf.

```
10.74.2.3 OS_printf_enable() void OS_printf_enable (
    void )
```

This function enables the output from OS\_printf.

## 10.75 OSAL Message Queue APIs

### Functions

- **int32 OS\_QueueCreate (osal\_id\_t \*queue\_id, const char \*queue\_name, osal\_blockcount\_t queue\_depth, size\_t data\_size, uint32 flags)**

*Create a message queue.*
- **int32 OS\_QueueDelete (osal\_id\_t queue\_id)**

*Deletes the specified message queue.*
- **int32 OS\_QueueGet (osal\_id\_t queue\_id, void \*data, size\_t size, size\_t \*size\_copied, int32 timeout)**

*Receive a message on a message queue.*
- **int32 OS\_QueuePut (osal\_id\_t queue\_id, const void \*data, size\_t size, uint32 flags)**

*Put a message on a message queue.*
- **int32 OS\_QueueGetIdByName (osal\_id\_t \*queue\_id, const char \*queue\_name)**

*Find an existing queue ID by name.*
- **int32 OS\_QueueGetInfo (osal\_id\_t queue\_id, OS\_queue\_prop\_t \*queue\_prop)**

*Fill a property object buffer with details regarding the resource.*

### 10.75.1 Detailed Description

### 10.75.2 Function Documentation

```
10.75.2.1 OS_QueueCreate() int32 OS_QueueCreate (
    osal_id_t * queue_id,
    const char * queue_name,
    osal_blockcount_t queue_depth,
    size_t data_size,
    uint32 flags )
```

Create a message queue.

This is the function used to create a queue in the operating system. Depending on the underlying operating system, the memory for the queue will be allocated automatically or allocated by the code that sets up the queue. Queue names must be unique; if the name already exists this function fails. Names cannot be NULL.

#### Parameters

out	<i>queue_id</i>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<i>queue_name</i>	the name of the new resource to create (must not be null)
in	<i>queue_depth</i>	the maximum depth of the queue
in	<i>data_size</i>	the size of each entry in the queue (must not be zero)
in	<i>flags</i>	options for the queue (reserved for future use, pass as 0)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if a pointer passed in is NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NO_FREE_IDS</i>	if there are already the max queues created
<i>OS_ERR_NAME_TAKEN</i>	if the name is already being used on another queue
<i>OS_ERR_INVALID_SIZE</i>	if data_size is 0
<i>OS_QUEUE_INVALID_SIZE</i>	if the queue depth exceeds the limit
<i>OS_ERROR</i>	if the OS create call fails

**10.75.2.2 OS\_QueueDelete()** `int32 OS_QueueDelete ( osal_id_t queue_id )`

Deletes the specified message queue.

This is the function used to delete a queue in the operating system. This also frees the respective queue\_id to be used again when another queue is created.

**Note**

If There are messages on the queue, they will be lost and any subsequent calls to QueueGet or QueuePut to this queue will result in errors

**Parameters**

in	<i>queue_id</i>	The object ID to delete
----	-----------------	-------------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in does not exist
<i>OS_ERROR</i>	if the OS call returns an unexpected error (return value only verified in coverage test)

**10.75.2.3 OS\_QueueGet()** `int32 OS_QueueGet (`

```
    osal_id_t queue_id,
    void * data,
    size_t size,
    size_t * size_copied,
```

```
int32 timeout )
```

Receive a message on a message queue.

If a message is pending, it is returned immediately. Otherwise the calling task will block until a message arrives or the timeout expires.

#### Parameters

in	<i>queue_id</i>	The object ID to operate on
out	<i>data</i>	The buffer to store the received message (must not be null)
in	<i>size</i>	The size of the data buffer (must not be zero)
out	<i>size_copied</i>	Set to the actual size of the message (must not be null)
in	<i>timeout</i>	The maximum amount of time to block, or OS_PEND to wait forever

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the given ID does not exist
<a href="#">OS_INVALID_POINTER</a>	if a pointer passed in is NULL
<a href="#">OS_QUEUE_EMPTY</a>	if the Queue has no messages on it to be received
<a href="#">OS_QUEUE_TIMEOUT</a>	if the timeout was OS_PEND and the time expired
<a href="#">OS_QUEUE_INVALID_SIZE</a>	if the size copied from the queue was not correct
<a href="#">OS_ERROR</a>	if the OS call returns an unexpected error (return value only verified in coverage test)

#### 10.75.2.4 OS\_QueueGetIdByName()

```
int32 OS_QueueGetIdByName (
    osal_id_t * queue_id,
    const char * queue_name )
```

Find an existing queue ID by name.

This function tries to find a queue Id given the name of the queue. The id of the queue is passed back in *queue\_id*.

#### Parameters

out	<i>queue_id</i>	will be set to the ID of the existing resource
in	<i>queue_name</i>	the name of the existing resource to find (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if the name or id pointers are NULL
<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>	the name was not found in the table

```
10.75.2.5 OS_QueueGetInfo() int32 OS_QueueGetInfo (
    osal_id_t queue_id,
    OS_queue_prop_t * queue_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info (name and creator) about the specified queue.

#### Parameters

in	<i>queue_id</i>	The object ID to operate on
out	<i>queue_prop</i>	The property object buffer to fill (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if queue_prop is NULL
<a href="#">OS_ERR_INVALID_ID</a>	if the ID given is not a valid queue

```
10.75.2.6 OS_QueuePut() int32 OS_QueuePut (
```

```
    osal_id_t queue_id,
    const void * data,
    size_t size,
    uint32 flags )
```

Put a message on a message queue.

#### Parameters

in	<i>queue_id</i>	The object ID to operate on
in	<i>data</i>	The buffer containing the message to put (must not be null)
in	<i>size</i>	The size of the data buffer (must not be zero)
in	<i>flags</i>	Currently reserved/unused, should be passed as 0

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the queue id passed in is not a valid queue
<a href="#">OS_INVALID_POINTER</a>	if the data pointer is NULL

### Return values

<code>OS_QUEUE_INVALID_SIZE</code>	if the data message is too large for the queue
<code>OS_QUEUE_FULL</code>	if the queue cannot accept another message
<code>OS_ERROR</code>	if the OS call returns an unexpected error (return value only verified in coverage test)

## 10.76 OSAL RwLock APIs

### Functions

- `int32 OS_RwLockCreate (osal_id_t *rw_id, const char *rw_name, uint32 options)`

*Creates an readers-writer lock (rwlock)*
- `int32 OS_RwLockReadGive (osal_id_t rw_id)`

*Releases the reader lock on the rwlock object referenced by rw\_id.*
- `int32 OS_RwLockWriteGive (osal_id_t rw_id)`

*Releases the rwlock object referenced by rw\_id.*
- `int32 OS_RwLockReadTake (osal_id_t rw_id)`

*Acquire the rwlock object as a read lock as referenced by rw\_id.*
- `int32 OS_RwLockWriteTake (osal_id_t rw_id)`

*Acquire the rwlock object as a write lock as referenced by rw\_id.*
- `int32 OS_RwLockDelete (osal_id_t rw_id)`

*Deletes the specified RwLock.*
- `int32 OS_RwLockGetIdByName (osal_id_t *rw_id, const char *rw_name)`

*Find an existing rwlock ID by name.*
- `int32 OS_RwLockGetInfo (osal_id_t rw_id, OS_rwlock_prop_t *rw_prop)`

*Fill a property object buffer with details regarding the resource.*

### 10.76.1 Detailed Description

### 10.76.2 Function Documentation

**10.76.2.1 OS\_RwLockCreate()** `int32 OS_RwLockCreate (`  
`osal_id_t * rw_id,`  
`const char * rw_name,`  
`uint32 options )`

Creates an readers-writer lock (rwlock)

RwLocks are always created in the unlocked (full) state.

#### Parameters

<code>out</code>	<code>rw_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
<code>in</code>	<code>rw_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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if sem_id or sem_name are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <i>OS_MAX_API_NAME</i>
<i>OS_ERR_NO_FREE_IDS</i>	if there are no more free rwlock Ids
<i>OS_ERR_NAME_TAKEN</i>	if there is already a rwlock with the same name
<i>OS_ERROR</i>	if the OS call failed (return value only verified in coverage test)

**10.76.2.2 OS\_RwLockDelete()** `int32 OS_RwLockDelete ( osal_id_t rw_id )`

Deletes the specified RwLock.

Delete the rwlock. This also frees the respective rw\_id such that it can be used again when another is created.

## Parameters

in	<i>rw_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 rwlock
<i>OS_ERROR</i>	if an unspecified error occurs (return value only verified in coverage test)

**10.76.2.3 OS\_RwLockGetIdByName()** `int32 OS_RwLockGetIdByName ( osal_id_t * rw_id, const char * rw_name )`

Find an existing rwlock ID by name.

This function tries to find a rwlock Id given the name of a rwlock. The id is returned through rw\_id

## Parameters

out	<i>rw_id</i>	will be set to the ID of the existing resource
in	<i>rw_name</i>	the name of the existing resource to find (must not be null)

## Returns

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

## Return values

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if <i>rw_id</i> or <i>rw_name</i> 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

**10.76.2.4 OS\_RwLockGetInfo()** *int32 OS\_RwLockGetInfo(*

```
osal_id_t rw_id,  
OS_rwlock_prop_t * rw_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 rwlock.

## Parameters

<i>in</i>	<i>rw_id</i>	The object ID to operate on
<i>out</i>	<i>rw_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 rwlock
<i>OS_INVALID_POINTER</i>	if the <i>rw_prop</i> pointer is null

**10.76.2.5 OS\_RwLockReadGive()** *int32 OS\_RwLockReadGive(*

```
osal_id_t rw_id )
```

Releases the reader lock on the rwlock object referenced by *rw\_id*.

If this function is called by a thread which holds a read lock on this rwlock object, and there are other threads currently holding read locks, the rwlock will remain in the read locked state. If the function is called by the last thread which held a read lock, the rwlock becomes unlocked.

If this function is called by a thread which holds a write lock on this rwlock object, the behavior is undefined.

If there are threads blocked on the rwlock object referenced by *rw\_id* when this function is called, resulting in the rwlock becoming available, the scheduling policy shall determine which thread shall acquire the rwlock.

## Parameters

<i>in</i>	<i>rw_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 rwlock
<i>OS_ERROR</i>	if an unspecified error occurs (return value only verified in coverage test)

**10.76.2.6 OS\_RwLockReadTake()** `int32 OS_RwLockReadTake ( osal_id_t rw_id )`

Acquire the rwlock object as a read lock as referenced by `rw_id`.

If the rwlock is currently write locked, as in a thread holds a write lock on this rwlock object, the current thread will block until the rwlock becomes available to readers.

If the rwlock is currently read locked and a thread intending to obtain a write lock is waiting, whether or not the current thread is allowed to obtain a read lock depends on the underlying OS rwlock implementation.

If the rwlock is currently read locked and there are no threads waiting to obtain a write lock, the current thread is allowed to obtain a read lock.

If the rwlock is unlocked, the current thread will obtain a read lock on this rwlock object.

**Parameters**

in	<i>rw_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 rwlock
<i>OS_ERROR</i>	if an unspecified error occurs (return value only verified in coverage test)

**10.76.2.7 OS\_RwLockWriteGive()** `int32 OS_RwLockWriteGive ( osal_id_t rw_id )`

Releases the rwlock object referenced by `rw_id`.

If this function is called by a thread which holds a write lock on this rwlock object, the rwlock becomes unlocked.

If this function is called by a thread which holds a read lock on this rwlock object, the behavior is undefined.

If there are threads blocked on the rwlock object referenced by rwlock when this function is called, resulting in the rwlock becoming available, the scheduling policy shall determine which thread shall acquire the rwlock.

**Parameters**

in	<i>rw_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 rwlock
<code>OS_ERROR</code>	if an unspecified error occurs (return value only verified in coverage test)

### 10.76.2.8 `OS_RwLockWriteTake()` `int32 OS_RwLockWriteTake( osal_id_t rw_id )`

Acquire the rwlock object as a write lock as referenced by `rw_id`.

If the rwlock is currently write or read locked, the current thread will block until the rwlock becomes available. If the rwlock is open, the current thread will acquire a write lock on this rwlock object.

**Parameters**

in	<code>rw_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>	the id passed in is not a valid rwlock
<code>OS_ERROR</code>	if an unspecified error occurs (return value only verified in coverage test)

## 10.77 OSAL Select APIs

**Functions**

- `int32 OS_SelectMultipleAbs (OS_FdSet *ReadSet, OS_FdSet *WriteSet, OS_time_t abs_timeout)`  
*Wait for events across multiple file handles.*
- `int32 OS_SelectMultiple (OS_FdSet *ReadSet, OS_FdSet *WriteSet, int32 msecs)`  
*Wait for events across multiple file handles.*
- `int32 OS_SelectSingleAbs (osal_id_t objid, uint32 *StateFlags, OS_time_t abs_timeout)`  
*Wait for events on a single file handle.*
- `int32 OS_SelectSingle (osal_id_t objid, uint32 *StateFlags, int32 msecs)`  
*Wait for events on a single file handle.*
- `int32 OS_SelectFdZero (OS_FdSet *Set)`  
*Clear a FdSet structure.*
- `int32 OS_SelectFdAdd (OS_FdSet *Set, osal_id_t objid)`  
*Add an ID to an FdSet structure.*

- `int32 OS_SelectFdClear (OS_FdSet *Set, osal_id_t objid)`  
*Clear an ID from an FdSet structure.*
- `bool OS_SelectFdIsSet (const OS_FdSet *Set, osal_id_t objid)`  
*Check if an FdSet structure contains a given ID.*

### 10.77.1 Detailed Description

### 10.77.2 Function Documentation

#### 10.77.2.1 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	<i>Set</i>	Pointer to <code>OS_FdSet</code> object to operate on (must not be null)
in	<i>objid</i>	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

#### 10.77.2.2 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 <code>OS_FdSet</code> 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

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_ID</i>	if the objid is not a valid handle

**10.77.2.3 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 <i>OS_FdSet</i> 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

**10.77.2.4 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 <i>OS_FdSet</i> 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

**10.77.2.5 OS\_SelectMultiple()** `int32 OS_SelectMultiple (`

```
OS_FdSet * ReadSet,
OS_FdSet * WriteSet,
int32 msecs )
```

Wait for events across multiple file handles.

Wait for any of the given sets of IDs to become readable or writable

This function will block until any of the following occurs:

- At least one OSAL ID in the ReadSet is readable
- At least one OSAL ID in the WriteSet is writable
- The timeout has elapsed

The sets are input/output parameters. On entry, these indicate the file handle(s) to wait for. On exit, these are set to the actual file handle(s) that have activity.

If the timeout occurs this returns an error code and all output sets should be empty.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS\\_SelectMultipleAbs\(\)](#) for higher timing precision.

#### Note

This does not lock or otherwise protect the file handles in the given sets. If a filehandle supplied via one of the FdSet arguments is closed or modified by another while this function is in progress, the results are undefined. Because of this limitation, it is recommended to use [OS\\_SelectSingle\(\)](#) whenever possible.

#### Parameters

in,out	<i>ReadSet</i>	Set of handles to check/wait to become readable
in,out	<i>WriteSet</i>	Set of handles to check/wait to become writable
in	<i>msecs</i>	Indicates the timeout. Positive values will wait up to that many milliseconds. Zero will not wait (poll). Negative values will wait forever (pend)

#### See also

[OS\\_SelectMultipleAbs\(\)](#)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	If any handle in the ReadSet or WriteSet is readable or writable, respectively
<a href="#">OS_ERROR_TIMEOUT</a>	If no handles in the ReadSet or WriteSet became readable or writable within the timeout
<a href="#">OS_ERR_OPERATION_NOT_SUPPORTED</a>	if a specified handle does not support select
<a href="#">OS_ERR_INVALID_ID</a>	if no valid handles were contained in the ReadSet/WriteSet

**10.77.2.6 OS\_SelectMultipleAbs()** `int32 OS_SelectMultipleAbs (`  
`OS_FdSet * ReadSet,`

```
OS_FdSet * WriteSet,
os_time_t abs_timeout )
```

Wait for events across multiple file handles.

Wait for any of the given sets of IDs to become readable or writable

This function will block until any of the following occurs:

- At least one OSAL ID in the ReadSet is readable
- At least one OSAL ID in the WriteSet is writable
- The timeout has elapsed

The sets are input/output parameters. On entry, these indicate the file handle(s) to wait for. On exit, these are set to the actual file handle(s) that have activity.

If the timeout occurs this returns an error code and all output sets should be empty.

This API is identical to [OS\\_SelectMultiple\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS\\_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS\\_SelectMultiple\(\)](#).

#### Note

This does not lock or otherwise protect the file handles in the given sets. If a filehandle supplied via one of the FdSet arguments is closed or modified by another while this function is in progress, the results are undefined. Because of this limitation, it is recommended to use [OS\\_SelectSingle\(\)](#) whenever possible.

#### Parameters

in, out	<i>ReadSet</i>	Set of handles to check/wait to become readable
in, out	<i>WriteSet</i>	Set of handles to check/wait to become writable
in	<i>abs_timeout</i>	The absolute time that the call may block until

#### See also

[OS\\_SelectMultiple\(\)](#)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	If any handle in the ReadSet or WriteSet is readable or writable, respectively
<a href="#">OS_ERROR_TIMEOUT</a>	If no handles in the ReadSet or WriteSet became readable or writable within the timeout
<a href="#">OS_ERR_OPERATION_NOT_SUPPORTED</a>	if a specified handle does not support select
<a href="#">OS_ERR_INVALID_ID</a>	if no valid handles were contained in the ReadSet/WriteSet

```
10.77.2.7 OS_SelectSingle() int32 OS_SelectSingle(
    osal_id_t objid,
    uint32 * StateFlags,
```

```
int32 msecs )
```

Wait for events on a single file handle.

Wait for a single OSAL filehandle to change state

This function can be used to wait for a single OSAL stream ID to become readable or writable. On entry, the "State←Flags" parameter should be set to the desired state (OS\_STREAM\_STATE\_READABLE and/or OS\_STREAM\_STATE←\_WRITABLE) and upon return the flags will be set to the state actually detected.

As this operates on a single ID, the filehandle is protected during this call, such that another thread accessing the same handle will return an error. However, it is important to note that once the call returns then other threads may then also read/write and affect the state before the current thread can service it.

To mitigate this risk the application may prefer to use the OS\_TimedRead/OS\_TimedWrite calls.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS\\_SelectSingleAbs\(\)](#) for higher timing precision.

#### Parameters

in	<i>objid</i>	The handle ID to select on
in, out	<i>StateFlags</i>	State flag(s) (readable or writable) (must not be null)
in	<i>msecs</i>	Indicates the timeout. Positive values will wait up to that many milliseconds. Zero will not wait (poll). Negative values will wait forever (pend)

#### See also

[OS\\_SelectSingleAbs\(\)](#)

#### Returns

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

#### Return values

<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

#### 10.77.2.8 OS\_SelectSingleAbs() `int32 OS_SelectSingleAbs (`

```
    osal_id_t objid,
    uint32 * StateFlags,
    OS_time_t abs_timeout )
```

Wait for events on a single file handle.

Wait for a single OSAL filehandle to change state

This function can be used to wait for a single OSAL stream ID to become readable or writable. On entry, the "State←Flags" parameter should be set to the desired state (OS\_STREAM\_STATE\_READABLE and/or OS\_STREAM\_STATE←\_WRITABLE) and upon return the flags will be set to the state actually detected.

As this operates on a single ID, the filehandle is protected during this call, such that another thread accessing the same handle will return an error. However, it is important to note that once the call returns then other threads may then also read/write and affect the state before the current thread can service it.

To mitigate this risk the application may prefer to use the OS\_TimedRead/OS\_TimedWrite calls.

This API is identical to [OS\\_SelectSingle\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS\\_GetLocalTime\(\)](#). This allows for a more

precise timeout than what is possible via the normal [OS\\_SelectSingle\(\)](#).

#### Parameters

in	<i>objid</i>	The handle ID to select on
in, out	<i>StateFlags</i>	State flag(s) (readable or writable) (must not be null)
in	<i>abs_timeout</i>	The absolute time that the call may block until

#### See also

[OS\\_SelectSingle\(\)](#)

#### Returns

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

#### Return values

<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

## 10.78 OSAL Shell APIs

### Functions

- `int32 OS_ShellOutputToFile (const char *Cmd, osal\_id\_t filedes)`  
*Executes the command and sends output to a file.*

#### 10.78.1 Detailed Description

#### 10.78.2 Function Documentation

**10.78.2.1 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

## 10.79 OSAL Socket Address APIs

### Functions

- `int32 OS_SocketAddrInit (OS_SockAddr_t *Addr, OS_SocketDomain_t Domain)`  
*Initialize a socket address structure to hold an address of the given family.*
- `int32 OS_SocketAddrToString (char *buffer, size_t buflen, const OS_SockAddr_t *Addr)`  
*Get a string representation of a network host address.*
- `int32 OS_SocketAddrFromString (OS_SockAddr_t *Addr, const char *string)`  
*Set a network host address from a string representation.*
- `int32 OS_SocketAddrGetPort (uint16 *PortNum, const OS_SockAddr_t *Addr)`  
*Get the port number of a network address.*
- `int32 OS_SocketAddrSetPort (OS_SockAddr_t *Addr, uint16 PortNum)`  
*Set the port number of a network address.*

### 10.79.1 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.

### 10.79.2 Function Documentation

**10.79.2.1 OS\_SocketAddrFromString()** `int32 OS_SocketAddrFromString (`  
    `OS_SockAddr_t * Addr,`  
    `const char * string )`

Set a network host address from a string representation.

The specific format of the output string depends on the address family.

The address structure should have been previously initialized using [OS\\_SocketAddrInit\(\)](#) to set the address family type.

#### Note

For IPv4, this would typically be the dotted-decimal format (X.X.X.X). It is up to the discretion of the underlying implementation whether to accept hostnames, as this depends on the availability of DNS services. Since many embedded deployments do not have name services, this should not be relied upon.

**Parameters**

out	<i>Addr</i>	The address buffer to initialize (must not be null)
in	<i>string</i>	The string to initialize the address from (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERROR</i>	if the string cannot be converted to an address

**10.79.2.2 OS\_SocketAddrGetPort()** `int32 OS_SocketAddrGetPort (`

```
    uint16 * PortNum,
    const OS_SockAddr_t * Addr )
```

Get the port number of a network address.

For network protocols that have the concept of a port number (such as TCP/IP and UDP/IP) this function gets the port number from the address structure.

**Parameters**

out	<i>PortNum</i>	Buffer to store the port number (must not be null)
in	<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

**10.79.2.3 OS\_SocketAddrInit()** `int32 OS_SocketAddrInit (`

```
    OS_SockAddr_t * Addr,
    OS_SocketDomain_t Domain )
```

Initialize a socket address structure to hold an address of the given family.

The address is set to a suitable default value for the family.

**Parameters**

out	<i>Addr</i>	The address buffer to initialize (must not be null)
-----	-------------	---

**Parameters**

in	<i>Domain</i>	The address family
----	---------------	--------------------

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if Addr argument is NULL
<i>OS_ERR_NOT_IMPLEMENTED</i>	if the system does not implement the requested domain

**10.79.2.4 OS\_SocketAddrSetPort()** `int32 OS_SocketAddrSetPort (`

```
    OS_SockAddr_t * Addr,  
    uint16 PortNum )
```

Set the port number of a network address.

For network protocols that have the concept of a port number (such as TCP/IP and UDP/IP) this function sets the port number from the address structure.

**Parameters**

out	<i>Addr</i>	The network address buffer (must not be null)
in	<i>PortNum</i>	The port number to set

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_BAD_ADDRESS</i>	if the address domain is not compatible

**10.79.2.5 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**

<code>out</code>	<code>buffer</code>	Buffer to hold the output string (must not be null)
<code>in</code>	<code>buflen</code>	Maximum length of the output string (must not be zero)
<code>in</code>	<code>Addr</code>	The network address buffer to convert (must not be null)

**Returns**

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

**Return values**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_INVALID_POINTER</code>	if argument is NULL
<code>OS_ERR_INVALID_SIZE</code>	if passed-in buflen is not valid
<code>OS_ERROR</code>	if the address cannot be converted to string, or string buffer too small

## 10.80 OSAL Socket Management APIs

**Functions**

- `int32 OS_SocketOpen (osal_id_t *sock_id, OS_SocketDomain_t Domain, OS_SocketType_t Type)`

*Opens a socket.*
- `int32 OS_SocketBind (osal_id_t sock_id, const OS_SockAddr_t *Addr)`

*Binds a socket to a given local address and enter listening (server) mode.*
- `int32 OS_SocketListen (osal_id_t sock_id)`

*Places the specified socket into a listening state.*
- `int32 OS_SocketBindAddress (osal_id_t sock_id, const OS_SockAddr_t *Addr)`

*Binds a socket to a given local address.*
- `int32 OS_SocketConnectAbs (osal_id_t sock_id, const OS_SockAddr_t *Addr, OS_time_t abs_timeout)`

*Connects a socket to a given remote address.*
- `int32 OS_SocketConnect (osal_id_t sock_id, const OS_SockAddr_t *Addr, int32 timeout)`

*Connects a socket to a given remote address.*
- `int32 OS_SocketShutdown (osal_id_t sock_id, OS_SocketShutdownMode_t Mode)`

*Implement graceful shutdown of a stream socket.*
- `int32 OS_SocketAcceptAbs (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, OS_time_t abs_timeout)`

*Waits for and accept the next incoming connection on the given socket.*
- `int32 OS_SocketAccept (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, int32 timeout)`

*Waits for and accept the next incoming connection on the given socket.*
- `int32 OS_SocketRecvFromAbs (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, OS_time_t abs_timeout)`

*Reads data from a message-oriented (datagram) socket.*

- `int32 OS_SocketRecvFrom (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, int32 timeout)`  
*Reads data from a message-oriented (datagram) socket.*
- `int32 OS_SocketSendTo (osal_id_t sock_id, const void *buffer, size_t buflen, const OS_SockAddr_t *RemoteAddr)`  
*Sends data to a message-oriented (datagram) socket.*
- `int32 OS_SocketGetIdByName (osal_id_t *sock_id, const char *sock_name)`  
*Gets an OSAL ID from a given name.*
- `int32 OS_SocketGetInfo (osal_id_t sock_id, OS_socket_prop_t *sock_prop)`  
*Gets information about an OSAL Socket ID.*
- `int32 OS_SocketgetOption (osal_id_t sock_id, OS_socket_option_t opt_id, OS_socket_optval_t *optval)`  
*Gets the value of a socket option.*
- `int32 OS_SocketSetOption (osal_id_t sock_id, OS_socket_option_t opt_id, const OS_socket_optval_t *optval)`  
*Sets the value of a socket option.*

#### 10.80.1 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.

#### 10.80.2 Function Documentation

##### 10.80.2.1 `OS_SocketAccept()` `int32 OS_SocketAccept (`

```
    osal_id_t sock_id,
    osal_id_t * connsock_id,
    OS_SockAddr_t * Addr,
    int32 timeout )
```

Waits for and accept the next incoming connection on the given socket.

This is used for sockets operating in a "server" role. The socket must be a stream type (connection-oriented) and previously bound to a local address using `OS_SocketBind()`. This will block the caller up to the given timeout or until an incoming connection request occurs, whichever happens first.

The new stream connection is then returned to the caller and the original server socket ID can be reused for the next connection.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use `OS_SocketAcceptAbs()` for higher timing precision.

##### Parameters

in	<code>sock_id</code>	The server socket ID, previously bound using <code>OS_SocketBind()</code>
out	<code>connsock_id</code>	The connection socket, a new ID that can be read/written (must not be null)
in	<code>Addr</code>	The remote address of the incoming connection (must not be null)
in	<code>timeout</code>	The maximum amount of time to wait, or <code>OS_PEND</code> to wait forever

**See also**

[OS\\_SocketAcceptAbs\(\)](#)

**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 sock_id 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

**10.80.2.2 OS\_SocketAcceptAbs()**

```
int32 OS_SocketAcceptAbs (
    osal_id_t sock_id,
    osal_id_t * connsock_id,
    OS_SockAddr_t * Addr,
    OS_time_t abs_timeout )
```

Waits for and accept the next incoming connection on the given socket.

This is used for sockets operating in a "server" role. The socket must be a stream type (connection-oriented) and previously bound to a local address using [OS\\_SocketBind\(\)](#). This will block the caller up to the given timeout or until an incoming connection request occurs, whichever happens first.

The new stream connection is then returned to the caller and the original server socket ID can be reused for the next connection.

This API is identical to [OS\\_SocketAccept\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS\\_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS\\_SocketAccept\(\)](#).

**Parameters**

in	<i>sock_id</i>	The server socket ID, previously bound using <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>abs_timeout</i>	The absolute time that the call may block until

**See also**

[OS\\_SocketAccept\(\)](#)

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
----------------------------	-----------------------

**Return values**

<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is not bound or already connected

**10.80.2.3 OS\_SocketBind()** `int32 OS_SocketBind (`  
    `osal_id_t sock_id,`  
    `const OS_SockAddr_t * Addr )`

Binds a socket to a given local address and enter listening (server) mode.

This is a convenience/compatibility routine to perform both [OS\\_SocketBindAddress\(\)](#) and [OS\\_SocketListen\(\)](#) operations in a single call, intended to simplify the setup for a server role.

If the socket is connectionless, then it only binds to the local address.

**Parameters**

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The local address to bind to (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already bound
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

**10.80.2.4 OS\_SocketBindAddress()** `int32 OS_SocketBindAddress (`  
    `osal_id_t sock_id,`  
    `const OS_SockAddr_t * Addr )`

Binds a socket to a given local address.

The specified socket will be bound to the local address and port, if available. This controls the source address reflected in network traffic transmitted via this socket.

After binding to the address, a stream socket may be followed by a call to either [OS\\_SocketListen\(\)](#) for a server role or to [OS\\_SocketConnect\(\)](#) for a client role.

**Parameters**

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The local address to bind to (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already bound
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

```
10.80.2.5 OS_SocketConnect() int32 OS_SocketConnect (
    osal_id_t sock_id,
    const OS_SockAddr_t * Addr,
    int32 timeout )
```

Connects a socket to a given remote address.

The socket will be connected to the remote address and port, if available. This only applies to stream-oriented sockets.

Calling this on a datagram socket will return an error (these sockets should use SendTo/RecvFrom).

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS\\_SocketConnectAbs\(\)](#) for higher timing precision.

**Parameters**

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The remote address to connect to (must not be null)
in	<i>timeout</i>	The maximum amount of time to wait, or <i>OS_PEND</i> to wait forever

**See also**

[OS\\_SocketConnectAbs\(\)](#)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if the socket is already connected
<i>OS_ERR_INVALID_ID</i>	if the <i>sock_id</i> parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket
<i>OS_INVALID_POINTER</i>	if <i>Addr</i> argument is NULL

```
10.80.2.6 OS_SocketConnectAbs() int32 OS_SocketConnectAbs (
```

```
    osal_id_t sock_id,
    const OS_SockAddr_t * Addr,
    OS_time_t abs_timeout )
```

Connects a socket to a given remote address.

The socket will be connected to the remote address and port, if available. This only applies to stream-oriented sockets. Calling this on a datagram socket will return an error (these sockets should use SendTo/RecvFrom).

This API is identical to [OS\\_SocketConnect\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS\\_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS\\_SocketConnect\(\)](#).

#### Parameters

in	<i>sock_id</i>	The socket ID
in	<i>Addr</i>	The remote address to connect to (must not be null)
in	<i>abs_timeout</i>	The absolute time that the call may block until

#### See also

[OS\\_SocketConnect\(\)](#)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INCORRECT_OBJ_STATE</a>	if the socket is already connected
<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_INVALID_POINTER</a>	if <i>Addr</i> argument is NULL

```
10.80.2.7 OS_SocketGetIdByName() int32 OS_SocketGetIdByName (
    osal_id_t * sock_id,
    const char * sock_name )
```

Gets an OSAL ID from a given name.

#### Note

OSAL Sockets use generated names according to the address and type.

#### See also

[OS\\_SocketGetInfo\(\)](#)

#### Parameters

out	<i>sock_id</i>	Buffer to hold result (must not be null)
in	<i>sock_name</i>	Name of socket to find (must not be null)

**Returns**

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

**Return values**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_INVALID_POINTER</a>	if id or 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

**10.80.2.8 OS\_SocketGetInfo()** `int32 OS_SocketGetInfo (`

```
    osal_id_t sock_id,
    OS_socket_prop_t * sock_prop )
```

Gets information about an OSAL Socket ID.

OSAL Sockets use generated names according to the address and type. This allows applications to find the name of a given socket.

**Parameters**

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**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the id passed in is not a valid socket
<a href="#">OS_INVALID_POINTER</a>	if the sock_prop pointer is null

**10.80.2.9 OS\_SocketgetOption()** `int32 OS_SocketgetOption (`

```
    osal_id_t sock_id,
    OS_socket_option_t opt_id,
    OS_socket_optval_t * optval )
```

Gets the value of a socket option.

Gets the state of a per-socket option / configurable item

**Parameters**

in	<i>sock_id</i>	The socket ID
in	<i>opt_id</i>	The socket option ID
out	<i>optval</i>	Buffer to hold socket option value (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 socket
<i>OS_INVALID_POINTER</i>	if the optval pointer is null

**10.80.2.10 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

**10.80.2.11 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

```
10.80.2.12 OS_SocketRecvFrom() int32 OS_SocketRecvFrom (
    osal_id_t sock_id,
    void * buffer,
    size_t buflen,
    OS_SockAddr_t * RemoteAddr,
    int32 timeout )
```

Reads data from a message-oriented (datagram) socket.

If a message is already available on the socket, this should immediately return that data without blocking. Otherwise, it may block up to the given timeout.

The timeout is expressed in milliseconds, relative to the time that the API was invoked. Use [OS\\_SocketRecvFromAbs\(\)](#) for higher timing precision.

**Parameters**

in	<i>sock_id</i>	The socket ID, previously bound using <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

**See also**

[OS\\_SocketRecvFromAbs\(\)](#)

**Returns**

Count of actual bytes received or error status, see [OSAL Return Code Defines](#)

**Return values**

<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_SIZE</i>	if passed-in buflen is not valid
<i>OS_ERR_INVALID_ID</i>	if the sock_id parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

```
10.80.2.13 OS_SocketRecvFromAbs() int32 OS_SocketRecvFromAbs (
```

```
osal_id_t sock_id,
void * buffer,
size_t buflen,
OS_SockAddr_t * RemoteAddr,
OS_time_t abs_timeout )
```

Reads data from a message-oriented (datagram) socket.

If a message is already available on the socket, this should immediately return that data without blocking. Otherwise, it may block up to the given timeout.

This API is identical to [OS\\_SocketRecvFrom\(\)](#) except for the timeout parameter. In this call, timeout is expressed as an absolute value of the OS clock, in the same time domain as obtained via [OS\\_GetLocalTime\(\)](#). This allows for a more precise timeout than what is possible via the normal [OS\\_SocketRecvFrom\(\)](#).

#### Parameters

in	<i>sock_id</i>	The socket ID, previously bound using <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>abs_timeout</i>	The absolute time at which the call should return if nothing received

#### Returns

Count of actual bytes received or error status, see [OSAL Return Code Defines](#)

#### Return values

<a href="#">OS_INVALID_POINTER</a>	if argument is NULL
<a href="#">OS_ERR_INVALID_SIZE</a>	if passed-in buflen is not valid
<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

#### 10.80.2.14 OS\_SocketSendTo() `int32 OS_SocketSendTo (`

```
    osal_id_t sock_id,
    const void * buffer,
    size_t buflen,
    const OS_SockAddr_t * RemoteAddr )
```

Sends data to a message-oriented (datagram) socket.

This sends data in a non-blocking mode. If the socket is not currently able to queue the message, such as if its outbound buffer is full, then this returns an error code.

#### Parameters

in	<i>sock_id</i>	The socket ID, which must be of the datagram type
in	<i>buffer</i>	Pointer to message data to send (must not be null)
in	<i>buflen</i>	The length of the message data to send (must not be zero)
in	<i>RemoteAddr</i>	Buffer containing the remote network address to send to

**Returns**

Count of actual bytes sent or error status, see [OSAL Return Code Defines](#)

**Return values**

<i>OS_INVALID_POINTER</i>	if argument is NULL
<i>OS_ERR_INVALID_SIZE</i>	if passed-in buflen is not valid
<i>OS_ERR_INVALID_ID</i>	if the sock_id parameter is not valid
<i>OS_ERR_INCORRECT_OBJ_TYPE</i>	if the handle is not a socket

**10.80.2.15 OS\_SocketSetOption()** `int32 OS_SocketSetOption (`

```
    osal_id_t sock_id,
    OS_socket_option_t opt_id,
    const OS_socket_optval_t * optval )
```

Sets the value of a socket option.

Sets the state of a per-socket option / configurable item

**Parameters**

in	<i>sock_id</i>	The socket ID
in	<i>opt_id</i>	The socket option ID
in	<i>optval</i>	Socket option value (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 socket
<i>OS_INVALID_POINTER</i>	if the optval pointer is null

**10.80.2.16 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**

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_INVALID_ID</code>	if the <code>sock_id</code> parameter is not valid
<code>OS_ERR_INVALID_ARGUMENT</code>	if the <code>Mode</code> argument is not one of the valid options
<code>OS_ERR_INCORRECT_OBJ_TYPE</code>	if the handle is not a socket
<code>OS_ERR_INCORRECT_OBJ_STATE</code>	if the socket is not connected

## 10.81 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.*

### 10.81.1 Detailed Description

### 10.81.2 Function Documentation

**10.81.2.1 OS\_TaskCreate()** `int32 OS_TaskCreate (`  
    `osal_id_t * task_id,`  
    `const char * task_name,`  
    `osal_task_entry function_pointer,`  
    `osal_stackptr_t stack_pointer,`  
    `size_t stack_size,`

```
    osal_priority_t priority,
    uint32 flags )
```

Creates a task and starts running it.

Creates a task and passes back the id of the task created. Task names must be unique; if the name already exists this function fails. Names cannot be NULL.

Portable applications should always specify the actual stack size in the stack\_size parameter, not 0. This size value is not enforced/checked by OSAL, but is simply passed through to the RTOS for stack creation. Some RTOS implementations may assume 0 means a default stack size while others may actually create a task with no stack.

Unlike stack\_size, the stack\_pointer is optional and can be specified as NULL. In that case, a stack of the requested size will be dynamically allocated from the system heap.

#### Parameters

out	<i>task_id</i>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<i>task_name</i>	the name of the new resource to create (must not be null)
in	<i>function_pointer</i>	the entry point of the new task (must not be null)
in	<i>stack_pointer</i>	pointer to the stack for the task, or NULL to allocate a stack from the system memory heap
in	<i>stack_size</i>	the size of the stack (must not be zero)
in	<i>priority</i>	initial priority of the new task
in	<i>flags</i>	initial options for the new task

#### Returns

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

#### Return values

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_INVALID_POINTER</i></a>	if any of the necessary pointers are NULL
<a href="#"><i>OS_ERR_INVALID_SIZE</i></a>	if the stack_size argument is zero
<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_INVALID_PRIORITY</i></a>	if the priority is bad (return value only verified in coverage test)
<a href="#"><i>OS_ERR_NO_FREE_IDS</i></a>	if there can be no more tasks created
<a href="#"><i>OS_ERR_NAME_TAKEN</i></a>	if the name specified is already used by a task
<a href="#"><i>OS_ERROR</i></a>	if an unspecified/other error occurs (return value only verified in coverage test)

#### 10.81.2.2 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**

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERROR</a>	if an unspecified/other error occurs (return value only verified in coverage test)

Referenced by CF\_CFDP\_InitEngine().

**10.81.2.3 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**

<code>in</code>	<code>task_id</code>	The object 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 ID given to it is invalid
<a href="#">OS_ERROR</a>	if the OS delete call fails (return value only verified in coverage test)

**10.81.2.4 OS\_TaskExit()** `void OS_TaskExit (`  
`void )`

Exits the calling task.

The calling thread is terminated. This function does not return.

**10.81.2.5 OS\_TaskFindIdBySystemData()** `int32 OS_TaskFindIdBySystemData (`  
`osal_id_t * task_id,`  
`const void * sysdata,`  
`size_t sysdata_size )`

Reverse-lookup the OSAL task ID from an operating system ID.

This provides a method by which an external entity may find the OSAL task ID corresponding to a system-defined identifier (e.g. TASK\_ID, pthread\_t, rtems\_id, etc).

Normally OSAL does not expose the underlying OS-specific values to the application, but in some circumstances, such as exception handling, the OS may provide this information directly to a BSP handler outside of the normal OSAL API.

**Parameters**

out	<i>task_id</i>	The buffer where the task id output is stored (must not be null)
in	<i>sysdata</i>	Pointer to the system-provided identification data
in	<i>sysdata_size</i>	Size of the system-provided identification data

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution. (return value only verified in coverage test)
<i>OS_INVALID_POINTER</i>	if a pointer argument is NULL

**10.81.2.6 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)

**10.81.2.7 OS\_TaskGetIdByName()** `int32 OS_TaskGetIdByName (`  
 `oSal_id_t * task_id,`  
 `const char * task_name )`

Find an existing task ID by name.

This function tries to find a task Id given the name of a task

**Parameters**

out	<i>task_id</i>	will be set to the ID of the existing resource
in	<i>task_name</i>	the name of the existing resource to find (must not be null)

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_INVALID_POINTER</i>	if the pointers passed in are NULL
<i>OS_ERR_NAME_TOO_LONG</i>	name length including null terminator greater than <a href="#">OS_MAX_API_NAME</a>
<i>OS_ERR_NAME_NOT_FOUND</i>	if the name wasn't found in the table

```
10.81.2.8 OS_TaskGetInfo() int32 OS_TaskGetInfo (
    osal_id_t task_id,
    OS_task_prop_t * task_prop )
```

Fill a property object buffer with details regarding the resource.

This function will pass back a pointer to structure that contains all of the relevant info (creator, stack size, priority, name) about the specified task.

#### Parameters

in	<i>task_id</i>	The object ID to operate on
out	<i>task_prop</i>	The property object buffer to fill (must not be null)

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the ID passed to it is invalid
<a href="#">OS_INVALID_POINTER</a>	if the task_prop pointer is NULL

```
10.81.2.9 OS_TaskInstallDeleteHandler() int32 OS_TaskInstallDeleteHandler (
    osal_task_entry function_pointer )
```

Installs a handler for when the task is deleted.

This function is used to install a callback that is called when the task is deleted. The callback is called when OS\_TaskDelete is called with the task ID. A task delete handler is useful for cleaning up resources that a task creates, before the task is removed from the system.

#### Parameters

in	<i>function_pointer</i>	function to be called when task exits
----	-------------------------	---------------------------------------

#### Returns

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

#### Return values

<a href="#">OS_ERR_INVALID_ID</a>	if the calling context is not an OSAL task
-----------------------------------	--

```
10.81.2.10 OS_TaskSetPriority() int32 OS_TaskSetPriority (
    osal_id_t task_id,
    osal_priority_t new_priority )
```

Sets the given task to a new priority.

#### Parameters

in	<i>task_id</i>	The object ID to operate on
in	<i>new_priority</i>	Set the new priority

#### Returns

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

#### Return values

<a href="#">OS_SUCCESS</a>	Successful execution.
<a href="#">OS_ERR_INVALID_ID</a>	if the ID passed to it is invalid
<a href="#">OS_ERR_INVALID_PRIORITY</a>	if the priority is greater than the max allowed (return value only verified in coverage test)
<a href="#">OS_ERROR</a>	if an unspecified/other error occurs (return value only verified in coverage test)

## 10.82 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.*

#### 10.82.1 Detailed Description

#### 10.82.2 Function Documentation

**10.82.2.1 OS\_TimeBaseCreate()** `int32 OS_TimeBaseCreate (`  
`osal_id_t * timebase_id,`  
`const char * timebase_name,`  
`OS_TimerSync_t external_sync )`

Create an abstract Time Base resource.

An OSAL time base is an abstraction of a "timer tick" that can, in turn, be used for measurement of elapsed time between events.

Time bases can be simulated by the operating system using the OS kernel-provided timing facilities, or based on a hardware timing source if provided by the BSP.

A time base object has a servicing task associated with it, that runs at elevated priority and will thereby interrupt user-level tasks when timing ticks occur.

If the `external_sync` function is passed as `NULL`, the operating system kernel timing resources will be utilized for a simulated timer tick.

If the `external_sync` function is not `NULL`, this should point to a BSP-provided function that will block the calling task until the next tick occurs. This can be used for synchronizing with hardware events.

#### Note

When provisioning a tunable RTOS kernel, such as RTEMS, the kernel should be configured to support at least `(OS_MAX_TASKS + OS_MAX_TIMEBASES)` threads, to account for the helper threads associated with time base objects.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

#### Parameters

out	<code>timebase_id</code>	will be set to the non-zero ID of the newly-created resource (must not be null)
in	<code>timebase_name</code>	The name of the time base (must not be null)
in	<code>external_sync</code>	A synchronization function for BSP hardware-based timer ticks

#### Returns

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

#### Return values

<code>OS_SUCCESS</code>	Successful execution.
<code>OS_ERR_NAME_TAKEN</code>	if the name specified is already used
<code>OS_ERR_NO_FREE_IDS</code>	if there can be no more timebase resources created
<code>OS_ERR_INCORRECT_OBJ_STATE</code>	if called from timer/timebase context
<code>OS_ERR_NAME_TOO_LONG</code>	if the <code>timebase_name</code> is too long
<code>OS_INVALID_POINTER</code>	if a pointer argument is <code>NULL</code>

**10.82.2.2 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	<code>timebase_id</code>	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

```
10.82.2.3 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

```
10.82.2.4 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 <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

```
10.82.2.5 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.
-------------------	-----------------------

## Return values

<code>OS_ERR_INVALID_ID</code>	if the id passed in is not a valid timebase
<code>OS_INVALID_POINTER</code>	if the timebase_prop pointer is null
<code>OS_ERR_INCORRECT_OBJ_STATE</code>	if called from timer/timebase context

```
10.82.2.6 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	<code>timebase_id</code>	The timebase resource to configure
in	<code>start_time</code>	The amount of delay for the first tick, in microseconds.
in	<code>interval_time</code>	The amount of delay between ticks, in microseconds.

## 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 timebase
<code>OS_ERR_INCORRECT_OBJ_STATE</code>	if called from timer/timebase context
<code>OS_TIMER_ERR_INVALID_ARGS</code>	if start_time or interval_time are out of range

**10.83 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.*

### 10.83.1 Detailed Description

### 10.83.2 Function Documentation

**10.83.2.1 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

<code>out</code>	<code>timer_id</code>	Will be set to the non-zero resource ID of the timer object (must not be null)
<code>in</code>	<code>timer_name</code>	Name of the timer object (must not be null)
<code>in</code>	<code>timebase_id</code>	The time base resource to use as a reference
<code>in</code>	<code>callback_ptr</code>	Application-provided function to invoke (must not be null)
<code>in</code>	<code>callback_arg</code>	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)

```
10.83.2.2 OS_TimerCreate() int32 OS_TimerCreate (
    osal_id_t * timer_id,
    const char * timer_name,
    uint32 * clock_accuracy,
    OS_TimerCallback_t callback_ptr )
```

Create a timer object.

A timer object is a resource that invokes the specified application-provided function upon timer expiration. Timers may be one-shot or periodic in nature.

This function creates a dedicated (hidden) time base object to service this timer, which is created and deleted with the timer object itself. The internal time base is configured for an OS simulated timer tick at the same interval as the timer. The callback function should be declared according to the `OS_TimerCallback_t` function pointer type. The `timer_id` value is passed to the callback function.

**Note**

`clock_accuracy` comes from the underlying OS tick value. The nearest integer microsecond value is returned, so may not be exact.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**See also**

[OS\\_TimerCallback\\_t](#)

**Parameters**

<code>out</code>	<code>timer_id</code>	Will be set to the non-zero resource ID of the timer object (must not be null)
<code>in</code>	<code>timer_name</code>	Name of the timer object (must not be null)
<code>out</code>	<code>clock_accuracy</code>	Expected precision of the timer, in microseconds. This is the underlying tick value rounded to the nearest microsecond integer. (must not be null)
<code>in</code>	<code>callback_ptr</code>	The function pointer of the timer callback (must not be null).

**Returns**

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

**Return values**

<a href="#"><code>OS_SUCCESS</code></a>	Successful execution.
<a href="#"><code>OS_INVALID_POINTER</code></a>	if any parameters are NULL
<a href="#"><code>OS_ERR_NAME_TOO_LONG</code></a>	name length including null terminator greater than <a href="#"><code>OS_MAX_API_NAME</code></a>
<a href="#"><code>OS_ERR_NAME_TAKEN</code></a>	if the name is already in use by another timer.
<a href="#"><code>OS_ERR_NO_FREE_IDS</code></a>	if all of the timers are already allocated.
<a href="#"><code>OS_ERR_INCORRECT_OBJ_STATE</code></a>	if invoked from a timer context
<a href="#"><code>OS_TIMER_ERR_INTERNAL</code></a>	if there was an error programming the OS timer (return value only verified in coverage test)

**10.83.2.3 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**

<a href="#"><code>OS_SUCCESS</code></a>	Successful execution.
<a href="#"><code>OS_ERR_INVALID_ID</code></a>	if the timer_id is invalid.
<a href="#"><code>OS_TIMER_ERR_INTERNAL</code></a>	if there was a problem deleting the timer in the host OS (return value only verified in coverage test)
<a href="#"><code>OS_ERR_INCORRECT_OBJ_STATE</code></a>	if called from timer/timebase context

**10.83.2.4 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**

<a href="#"><i>OS_SUCCESS</i></a>	Successful execution.
<a href="#"><i>OS_INVALID_POINTER</i></a>	if timer_id or timer_name are NULL pointers
<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>	if the name was not found in the table
<a href="#"><i>OS_ERR_INCORRECT_OBJ_STATE</i></a>	if called from timer/timebase context

**10.83.2.5 OS\_TimerGetInfo()**

```
int32 OS_TimerGetInfo (
    osal_id_t timer_id,
    OS_timer_prop_t * timer_prop )
```

Gets information about an existing timer.

This function takes timer\_id, and looks it up in the OS table. It puts all of the information known about that timer into a structure pointer to by timer\_prop.

**Note**

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**Parameters**

in	<i>timer_id</i>	The timer ID to operate on
out	<i>timer_prop</i>	<p>Buffer containing timer properties (must not be null)</p> <ul style="list-style-type: none"> <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**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the id passed in is not a valid timer
<i>OS_INVALID_POINTER</i>	if the timer_prop pointer is null
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context

```
10.83.2.6 OS_TimerSet() int32 OS_TimerSet (
    osal_id_t timer_id,
    uint32 start_time,
    uint32 interval_time )
```

Configures a periodic or one shot timer.

This function programs the timer with a start time and an optional interval time. The start time is the time in microseconds when the user callback function will be called. If the interval time is non-zero, the timer will be reprogrammed with that interval in microseconds to call the user callback function periodically. If the start time and interval time are zero, the function will return an error.

For a "one-shot" timer, the start\_time configures the expiration time, and the interval\_time should be passed as zero to indicate the timer is not to be automatically reset.

**Note**

The resolution of the times specified is limited to the clock accuracy returned in the OS\_TimerCreate call. If the times specified in the start\_msec or interval\_msec parameters are less than the accuracy, they will be rounded up to the accuracy of the timer.

This configuration API must not be used from the context of a timer callback. Timers should only be configured from the context of normal OSAL tasks.

**Parameters**

in	<i>timer_id</i>	The timer ID to operate on
in	<i>start_time</i>	Time in microseconds to the first expiration
in	<i>interval_time</i>	Time in microseconds between subsequent intervals, value of zero will only call the user callback function once after the start_msec time.

**Returns**

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

**Return values**

<i>OS_SUCCESS</i>	Successful execution.
<i>OS_ERR_INVALID_ID</i>	if the timer_id is not valid.
<i>OS_TIMER_ERR_INTERNAL</i>	if there was an error programming the OS timer (return value only verified in coverage test)
<i>OS_ERR_INCORRECT_OBJ_STATE</i>	if called from timer/timebase context
<i>OS_TIMER_ERR_INVALID_ARGS</i>	if the start_time or interval_time is out of range, or both 0

## 11 Data Structure Documentation

### 11.1 CCSDS\_ExtendedHeader Struct Reference

CCSDS packet extended header.

```
#include <ccsds_hdr.h>
```

#### Data Fields

- `uint8 Subsystem [2]`  
*subsystem qualifier*
- `uint8 SystemId [2]`  
*system qualifier*

#### 11.1.1 Detailed Description

CCSDS packet extended header.

Definition at line 73 of file ccsds\_hdr.h.

#### 11.1.2 Field Documentation

##### 11.1.2.1 Subsystem `uint8 CCSDS_ExtendedHeader::Subsystem[2]`

subsystem qualifier

Definition at line 75 of file ccsds\_hdr.h.

##### 11.1.2.2 SystemId `uint8 CCSDS_ExtendedHeader::SystemId[2]`

system qualifier

Definition at line 82 of file ccsds\_hdr.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/msg/fsw/inc/ccsds\\_hdr.h](#)

### 11.2 CCSDS\_PrimaryHeader Struct Reference

CCSDS packet primary header.

```
#include <ccsds_hdr.h>
```

#### Data Fields

- `uint8 StreamId [2]`  
*packet identifier word (stream ID)*
- `uint8 Sequence [2]`  
*packet sequence word*
- `uint8 Length [2]`  
*packet length word*

#### 11.2.1 Detailed Description

CCSDS packet primary header.

Definition at line 51 of file ccsds\_hdr.h.

### 11.2.2 Field Documentation

#### 11.2.2.1 Length `uint8 CCSDS_PrimaryHeader::Length[2]`

packet length word

Definition at line 65 of file `ccsds_hdr.h`.

#### 11.2.2.2 Sequence `uint8 CCSDS_PrimaryHeader::Sequence[2]`

packet sequence word

Definition at line 60 of file `ccsds_hdr.h`.

#### 11.2.2.3 StreamId `uint8 CCSDS_PrimaryHeader::StreamId[2]`

packet identifier word (stream ID)

Definition at line 53 of file `ccsds_hdr.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/msg/fsw/inc/ccsds_hdr.h`

## 11.3 CF\_AbandonCmd Struct Reference

Abandon command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CF_Transaction_Payload_t Payload`

#### 11.3.1 Detailed Description

Abandon command structure.

For command details see `CF_ABANDON_CC`

Definition at line 280 of file `default_cf_msgstruct.h`.

### 11.3.2 Field Documentation

#### 11.3.2.1 CommandHeader `CFE_MSG_CommandHeader_t CF_AbandonCmd::CommandHeader`

Command header.

Definition at line 282 of file `default_cf_msgstruct.h`.

#### 11.3.2.2 Payload `CF_Transaction_Payload_t CF_AbandonCmd::Payload`

Definition at line 283 of file `default_cf_msgstruct.h`.

Referenced by `CF_AbandonCmd()`.

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgstruct.h`

## 11.4 CF\_AppData\_t Struct Reference

The CF application global state structure.

```
#include <cf_app.h>
```

### Data Fields

- `CF_HkPacket_t` `hk`
- `uint32` `RunStatus`
- `CFE_SB_PipeId_t` `CmdPipe`
- `CFE_TBL_Handle_t` `config_handle`
- `CF_ConfigTable_t *` `config_table`
- `CF_Engine_t` `engine`

### 11.4.1 Detailed Description

The CF application global state structure.

This contains all variables related to CF application state

Definition at line 85 of file cf\_app.h.

### 11.4.2 Field Documentation

#### 11.4.2.1 CmdPipe `CFE_SB_PipeId_t` CF\_AppData\_t::`CmdPipe`

Definition at line 91 of file cf\_app.h.

Referenced by `CF_AppInit()`, and `CF_AppMain()`.

#### 11.4.2.2 config\_handle `CFE_TBL_Handle_t` CF\_AppData\_t::`config_handle`

Definition at line 93 of file cf\_app.h.

Referenced by `CF_CheckTables()`, and `CF_TableInit()`.

#### 11.4.2.3 config\_table `CF_ConfigTable_t *` CF\_AppData\_t::`config_table`

Definition at line 94 of file cf\_app.h.

Referenced by `CF_CFDP_AppendTlv()`, `CF_CFDP_ArmAckTimer()`, `CF_CFDP_ArmInactTimer()`, `CF_CFDP_CheckAckNakCount()`, `CF_CFDP_GetTempName()`, `CF_CFDP_InitEngine()`, `CF_CFDP_MsgOutGet()`, `CF_CFDP_ProcessPollingDirectories()`, `CF_CFDP_R_SendNak()`, `CF_CFDP_ReceiveMessage()`, `CF_CFDP_ReceivePdu()`, `CF_CFDP_S_HandleFileRetention()`, `CF_CFDP_S_SendFileData()`, `CF_CFDP_SendAck()`, `CF_CFDP_SendEof()`, `CF_CFDP_SendFin()`, `CF_CFDP_SendMd()`, `CF_CFDP_TxFile_Initiate()`, `CF_CheckTables()`, `CF_DoEnableDisableDequeue()`, `CF_DoEnableDisablePolldir()`, `CF_GetSetParamCmd()`, `CF_TableInit()`, `CF_Timer_Sec2Ticks()`, and `CF_ValidateMaxOutgoingCmd()`.

#### 11.4.2.4 engine `CF_Engine_t` CF\_AppData\_t::`engine`

Definition at line 96 of file cf\_app.h.

Referenced by `CF_CFDP_CycleEngine()`, `CF_CFDP_DisableEngine()`, `CF_CFDP_InitEngine()`, `CF_CFDP_MsgOutGet()`, `CF_CFDP_PlaybackDir()`, `CF_CFDP_ProcessPlaybackDirectories()`, `CF_CFDP_ProcessPlaybackDirectory()`, `CF_CFDP_ProcessPollingDirectories()`, `CF_CFDP_ReceiveMessage()`, `CF_CFDP_ReceivePdu()`, `CF_CFDP_Send()`, `CF_CFDP_StartRxTransaction()`, `CF_CFDP_TxFile()`, `CF_CFDP_TxFile_Initiate()`, `CF_CheckTables()`, `CF_CList_InsertAfter_Ex()`, `CF_CList_InsertBack_Ex()`, `CF_CList_Remove_Ex()`, `CF_DequeueTransaction()`, `CF_DisableEngineCmd()`, `CF_DoPurgeQueue()`, `CF_EnableEngineCmd()`, `CF_FindTransactionBySequenceNumberAllChannels()`.

CF\_FreeTransaction(), CF\_GetChannelFromTxn(), CF\_InsertSortPrio(), CF\_MoveTransaction(), CF\_TraverseAllTransactions\_All\_Channels(), CF\_TsnChanAction(), and CF\_WriteQueueCmd().

#### 11.4.2.5 **hk** `CF_HkPacket_t` CF\_AppData\_t::hk

Definition at line 87 of file cf\_app.h.

Referenced by CF\_AbandonCmd(), CF\_AppInit(), CF\_AppPipe(), CF\_CancelCmd(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_CycleEngine(), CF\_CFDP\_DisableEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvDrop(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvHold(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvPh(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_Send(), CF\_CFDP\_StartRxTransaction(), CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DisableEngineCmd(), CF\_DoFreezeThaw(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(), CF\_FreezeCmd(), CF\_SetParamCmd(), CF\_MoveTransaction(), CF\_NoopCmd(), CF\_PlaybackDirCmd(), CF\_ProcessGroundCommand(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_SendHkCmd(), CF\_ThawCmd(), CF\_TxFileCmd(), and CF\_WriteQueueCmd().

#### 11.4.2.6 **RunStatus** `uint32` CF\_AppData\_t::RunStatus

Definition at line 89 of file cf\_app.h.

Referenced by CF\_AppInit(), CF\_AppMain(), and CF\_CheckTables().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf\\_app.h](#)

## 11.5 CF\_CancelCmd Struct Reference

Cancel command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t` CommandHeader  
*Command header.*
- `CF_Transaction_Payload_t` Payload

#### 11.5.1 Detailed Description

Cancel command structure.

For command details see [CF\\_CANCEL\\_CC](#)

Definition at line 269 of file default\_cf\_msgstruct.h.

#### 11.5.2 Field Documentation

##### 11.5.2.1 **CommandHeader** `CFE_MSG_CommandHeader_t` CF\_CancelCmd::CommandHeader

Command header.

Definition at line 271 of file default\_cf\_msgstruct.h.

**11.5.2.2 Payload** [CF\\_Transaction\\_Payload\\_t](#) CF\_CancelCmd::Payload

Definition at line 272 of file default\_cf\_msgstruct.h.

Referenced by CF\_CancelCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

**11.6 CF\_CFDP\_FileDirectiveDispatchTable\_t Struct Reference**

A table of receive handler functions based on file directive code.

#include &lt;cf\_cfdp\_dispatch.h&gt;

**Data Fields**

- [CF\\_CFDP\\_StateRecvFunc\\_t fdirective \[CF\\_CFDP\\_FileDirective\\_INVALID\\_MAX\]](#)  
*a separate recv handler for each possible file directive PDU in this state*

**11.6.1 Detailed Description**

A table of receive handler functions based on file directive code.

For PDUs identified as a "file directive" type - generally anything other than file data - this provides a table to branch to a different handler function depending on the value of the file directive code.

Definition at line 87 of file cf\_cfdp\_dispatch.h.

**11.6.2 Field Documentation****11.6.2.1 fdirective** [CF\\_CFDP\\_StateRecvFunc\\_t](#) CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective [[CF\\_CFDP\\_FileDirective\\_INVALID\\_MAX](#)]

a separate recv handler for each possible file directive PDU in this state

Definition at line 90 of file cf\_cfdp\_dispatch.h.

Referenced by CF\_CFDP\_R1\_Recv(), CF\_CFDP\_R2\_Recv(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_S1\_Recv(), CF\_CFDP\_S2\_Recv(), and CF\_CFDP\_S\_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/fsr/src/cf\\_cfdp\\_dispatch.h](#)

**11.7 CF\_CFDP\_1v Struct Reference**

Structure representing CFDP LV Object format.

#include &lt;cf\_cfdp\_pdu.h&gt;

**Data Fields**

- [CF\\_CFDP\\_uint8\\_t length](#)

*Length of data field.***11.7.1 Detailed Description**

Structure representing CFDP LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

Defined per table 5-2 of CCSDS 727.0-B-5

Definition at line 164 of file cf\_cfdp\_pdu.h.

### 11.7.2 Field Documentation

#### 11.7.2.1 length `CF_CFDP_uint8_t` CF\_CFDP\_lv::length

Length of data field.

Definition at line 166 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeLV(), and CF\_CFDP\_EncodeLV().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_pdu.h](#)

## 11.8 CF\_CFDP\_PduAck Struct Reference

Structure representing CFDP Acknowledge PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint8_t directive_and_subtype_code`
- `CF_CFDP_uint8_t cc_and_transaction_status`

#### 11.8.1 Detailed Description

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

Definition at line 319 of file cf\_cfdp\_pdu.h.

### 11.8.2 Field Documentation

#### 11.8.2.1 cc\_and\_transaction\_status `CF_CFDP_uint8_t` CF\_CFDP\_PduAck::cc\_and\_transaction\_status

Definition at line 322 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeAck(), and CF\_CFDP\_EncodeAck().

#### 11.8.2.2 directive\_and\_subtype\_code `CF_CFDP_uint8_t` CF\_CFDP\_PduAck::directive\_and\_subtype\_code

Definition at line 321 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeAck(), and CF\_CFDP\_EncodeAck().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_pdu.h](#)

## 11.9 CF\_CFDP\_PduEof Struct Reference

Structure representing CFDP End of file PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint8_t cc`
- `CF_CFDP_uint32_t crc`
- `CF_CFDP_uint32_t size`

### 11.9.1 Detailed Description

Structure representing CFDP End of file PDU.

Defined per section 5.2.2 / table 5-6 of CCSDS 727.0-B-5

Definition at line 297 of file cf\_cfdp\_pdu.h.

### 11.9.2 Field Documentation

#### 11.9.2.1 cc `CF_CFDP_uint8_t` CF\_CFDP\_PduEof::cc

Definition at line 299 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeEof(), and CF\_CFDP\_EncodeEof().

#### 11.9.2.2 crc `CF_CFDP_uint32_t` CF\_CFDP\_PduEof::crc

Definition at line 300 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeEof(), and CF\_CFDP\_EncodeEof().

#### 11.9.2.3 size `CF_CFDP_uint32_t` CF\_CFDP\_PduEof::size

Definition at line 301 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeEof(), and CF\_CFDP\_EncodeEof().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_pdu.h](#)

## 11.10 CF\_CFDP\_PduFileDataContent Struct Reference

PDU file data content typedef for limit checking outgoing\_file\_chunk\_size table value and set parameter command.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `uint8 data [CF_MAX_PDU_SIZE - sizeof(CF_CFDP_PduFileDataHeader_t) - CF_CFDP_MIN_HEADER_SIZE]`

### 11.10.1 Detailed Description

PDU file data content typedef for limit checking outgoing\_file\_chunk\_size table value and set parameter command.

This definition allows for the largest data block possible, as CF\_MAX\_PDU\_SIZE - the minimum possible header size. In practice the outgoing file chunk size is limited by whichever is smaller; the remaining data, remaining space in the packet, and outgoing\_file\_chunk\_size.

Definition at line 380 of file cf\_cfdp\_pdu.h.

### 11.10.2 Field Documentation

#### 11.10.2.1 data `uint8 CF_CFDP_PduFileDataContent::data [CF_MAX_PDU_SIZE - sizeof(CF_CFDP_PduFileDataHeader_t) - CF_CFDP_MIN_HEADER_SIZE]`

Definition at line 382 of file cf\_cfdp\_pdu.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_pdu.h](#)

## 11.11 CF\_CFDP\_PduFileDataHeader Struct Reference

PDU file data header.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint32_t offset`

#### 11.11.1 Detailed Description

PDU file data header.

Definition at line 361 of file cf\_cfdp\_pdu.h.

#### 11.11.2 Field Documentation

##### 11.11.2.1 `offset` `CF_CFDP_uint32_t CF_CFDP_PduFileDataHeader::offset`

NOTE: while this is the only fixed/required field in the data PDU, it may have segment metadata prior to this, depending on how the fields in the base header are set

Definition at line 368 of file cf\_cfdp\_pdu.h.

Referenced by `CF_CFDP_DecodeFileDataHeader()`, and `CF_CFDP_EncodeFileDataHeader()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

## 11.12 CF\_CFDP\_PduFileDirectiveHeader Struct Reference

Structure representing CFDP File Directive Header.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint8_t directive_code`

#### 11.12.1 Detailed Description

Structure representing CFDP File Directive Header.

Defined per section 5.2 of CCSDS 727.0-B-5

Definition at line 151 of file cf\_cfdp\_pdu.h.

#### 11.12.2 Field Documentation

##### 11.12.2.1 `directive_code` `CF_CFDP_uint8_t CF_CFDP_PduFileDirectiveHeader::directive_code`

Definition at line 153 of file cf\_cfdp\_pdu.h.

Referenced by `CF_CFDP_DecodeFileDirectiveHeader()`, and `CF_CFDP_EncodeFileDirectiveHeader()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

## 11.13 CF\_CFDP\_PduFin Struct Reference

Structure representing CFDP Finished PDU.

```
#include <cf_cfdp_pdu.h>
```

**Data Fields**

- [CF\\_CFDP\\_uint8\\_t flags](#)

**11.13.1 Detailed Description**

Structure representing CFDP Finished PDU.

Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

Definition at line 309 of file cf\_cfdp\_pdu.h.

**11.13.2 Field Documentation****11.13.2.1 flags [CF\\_CFDP\\_uint8\\_t CF\\_CFDP\\_PduFin::flags](#)**

Definition at line 311 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeFin(), and CF\_CFDP\_EncodeFin().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_pdu.h](#)

**11.14 CF\_CFDP\_PduHeader Struct Reference**

Structure representing base CFDP PDU header.

```
#include <cf_cfdp_pdu.h>
```

**Data Fields**

- [CF\\_CFDP\\_uint8\\_t flags](#)  
*Flags indicating the PDU type, direction, mode, etc.*
- [CF\\_CFDP\\_uint16\\_t length](#)  
*Length of the entire PDU, in octets.*
- [CF\\_CFDP\\_uint8\\_t eid\\_tsn\\_lengths](#)  
*Lengths of the EID+TSN data (bitfields)*

**11.14.1 Detailed Description**

Structure representing base CFDP PDU header.

This header appears at the beginning of all CFDP PDUs, of all types. Note that the header is variable length, it also contains source and destination entity IDs, and the transaction sequence number.

Defined per section 5.1 of CCSDS 727.0-B-5

**Note**

this contains variable length data for the EID+TSN, which is *not* included in this definition. As a result, the `sizeof(CF_CFDP_PduHeader_t)` reflects only the size of the fixed fields. Use `CF_HeaderSize()` to get the actual size of this structure.

Definition at line 137 of file cf\_cfdp\_pdu.h.

**11.14.2 Field Documentation**

**11.14.2.1 eid\_tsn\_lengths** `CF_CFDP_uint8_t` `CF_CFDP_PduHeader::eid_tsn_lengths`  
Lengths of the EID+TSN data (bitfields)  
Definition at line 141 of file `cf_cfdp_pdu.h`.  
Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderWithoutSize()`.

**11.14.2.2 flags** `CF_CFDP_uint8_t` `CF_CFDP_PduHeader::flags`  
Flags indicating the PDU type, direction, mode, etc.  
Definition at line 139 of file `cf_cfdp_pdu.h`.  
Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderWithoutSize()`.

**11.14.2.3 length** `CF_CFDP_uint16_t` `CF_CFDP_PduHeader::length`  
Length of the entire PDU, in octets.  
Definition at line 140 of file `cf_cfdp_pdu.h`.  
Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderFinalSize()`.  
The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp_pdu.h`

## 11.15 CF\_CFDP\_PduMd Struct Reference

Structure representing CFDP Metadata PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint8_t segmentation_control`
- `CF_CFDP_uint32_t size`

#### 11.15.1 Detailed Description

Structure representing CFDP Metadata PDU.  
Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5  
Definition at line 352 of file `cf_cfdp_pdu.h`.

#### 11.15.2 Field Documentation

**11.15.2.1 segmentation\_control** `CF_CFDP_uint8_t` `CF_CFDP_PduMd::segmentation_control`  
Definition at line 354 of file `cf_cfdp_pdu.h`.  
Referenced by `CF_CFDP_DecodeMd()`, and `CF_CFDP_EncodeMd()`.

**11.15.2.2 size** `CF_CFDP_uint32_t` `CF_CFDP_PduMd::size`  
Definition at line 355 of file `cf_cfdp_pdu.h`.  
Referenced by `CF_CFDP_DecodeMd()`, and `CF_CFDP_EncodeMd()`.  
The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp_pdu.h`

## 11.16 CF\_CFDP\_PduNak Struct Reference

Structure representing CFDP Non-Acknowledge PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint32_t scope_start`
- `CF_CFDP_uint32_t scope_end`

#### 11.16.1 Detailed Description

Structure representing CFDP Non-Acknowledge PDU.

Defined per section 5.2.6 / table 5-10 of CCSDS 727.0-B-5

Definition at line 341 of file cf\_cfdp\_pdu.h.

#### 11.16.2 Field Documentation

##### 11.16.2.1 `scope_end` `CF_CFDP_uint32_t CF_CFDP_PduNak::scope_end`

Definition at line 344 of file cf\_cfdp\_pdu.h.

Referenced by `CF_CFDP_DecodeNak()`, and `CF_CFDP_EncodeNak()`.

##### 11.16.2.2 `scope_start` `CF_CFDP_uint32_t CF_CFDP_PduNak::scope_start`

Definition at line 343 of file cf\_cfdp\_pdu.h.

Referenced by `CF_CFDP_DecodeNak()`, and `CF_CFDP_EncodeNak()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

## 11.17 CF\_CFDP\_R\_SubstateDispatchTable\_t Struct Reference

A dispatch table for receive file transactions, receive side.

```
#include <cf_cfdp_dispatch.h>
```

### Data Fields

- const `CF_CFDP_FileDirectiveDispatchTable_t * state [CF_RxSubState_NUM_STATES]`

#### 11.17.1 Detailed Description

A dispatch table for receive file transactions, receive side.

This is used for "receive file" transactions upon receipt of a directive PDU. Depending on the sub-state of the transaction, a different action may be taken.

Definition at line 99 of file cf\_cfdp\_dispatch.h.

#### 11.17.2 Field Documentation

**11.17.2.1 state** const CF\_CFDP\_FileDirectiveDispatchTable\_t\* CF\_CFDP\_R\_SubstateDispatchTable\_t::state[CF\_RxSubState\_NUM\_STATES]

Definition at line 101 of file cf\_cfdp\_dispatch.h.

Referenced by CF\_CFDP\_R1\_Recv(), CF\_CFDP\_R2\_Recv(), and CF\_CFDP\_R\_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_dispatch.h](#)

## 11.18 CF\_CFDP\_S\_SubstateRecvDispatchTable\_t Struct Reference

A dispatch table for send file transactions, receive side.

```
#include <cf_cfdp_dispatch.h>
```

### Data Fields

- const CF\_CFDP\_FileDirectiveDispatchTable\_t\* substate[CF\_TxSubState\_NUM\_STATES]

#### 11.18.1 Detailed Description

A dispatch table for send file transactions, receive side.

This is used for "send file" transactions upon receipt of a directive PDU. Depending on the sub-state of the transaction, a different action may be taken.

Definition at line 110 of file cf\_cfdp\_dispatch.h.

#### 11.18.2 Field Documentation

**11.18.2.1 substate** const CF\_CFDP\_FileDirectiveDispatchTable\_t\* CF\_CFDP\_S\_SubstateRecvDispatchTable\_t::substate[CF\_TxSubState\_NUM\_STATES]

Definition at line 112 of file cf\_cfdp\_dispatch.h.

Referenced by CF\_CFDP\_S1\_Recv(), CF\_CFDP\_S2\_Recv(), and CF\_CFDP\_S\_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_dispatch.h](#)

## 11.19 CF\_CFDP\_S\_SubstateSendDispatchTable\_t Struct Reference

A dispatch table for send file transactions, transmit side.

```
#include <cf_cfdp_dispatch.h>
```

### Data Fields

- CF\_CFDP\_StateSendFunc\_t substate[CF\_TxSubState\_NUM\_STATES]

#### 11.19.1 Detailed Description

A dispatch table for send file transactions, transmit side.

This is used for "send file" transactions to generate the next PDU to be sent. Depending on the sub-state of the transaction, a different action may be taken.

Definition at line 121 of file cf\_cfdp\_dispatch.h.

#### 11.19.2 Field Documentation

**11.19.2.1 substate** `CF_CFDP_StateSendFunc_t` `CF_CFDP_S_SubstateSendDispatchTable_t::substate[CF_TxSubState_NUM_STAT]`  
 Definition at line 123 of file cf\_cfdp\_dispatch.h.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_dispatch.h`

## 11.20 CF\_CFDP\_SegmentRequest Struct Reference

Structure representing CFDP Segment Request.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint32_t offset_start`
- `CF_CFDP_uint32_t offset_end`

#### 11.20.1 Detailed Description

Structure representing CFDP Segment Request.

Defined per section 5.2.6 / table 5-11 of CCSDS 727.0-B-5

Definition at line 330 of file cf\_cfdp\_pdu.h.

#### 11.20.2 Field Documentation

**11.20.2.1 offset\_end** `CF_CFDP_uint32_t` `CF_CFDP_SegmentRequest::offset_end`

Definition at line 333 of file cf\_cfdp\_pdu.h.

Referenced by `CF_CFDP_DecodeSegmentRequest()`, and `CF_CFDP_EncodeSegmentRequest()`.

**11.20.2.2 offset\_start** `CF_CFDP_uint32_t` `CF_CFDP_SegmentRequest::offset_start`

Definition at line 332 of file cf\_cfdp\_pdu.h.

Referenced by `CF_CFDP_DecodeSegmentRequest()`, and `CF_CFDP_EncodeSegmentRequest()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

## 11.21 CF\_CFDP\_Tick\_args Struct Reference

Structure for use with the `CF_CFDP_DoTick()` functions.

```
#include <cf_cfdp.h>
```

### Data Fields

- `CF_Channel_t * chan`  
*channel structure*
- `const CF_Transaction_t * resume_point`  
*skip to this txn (if not null)*
- `void(* fn )(CF_Transaction_t *)`  
*function pointer*

### 11.21.1 Detailed Description

Structure for use with the `CF_CFDP_DoTick()` functions.  
Definition at line 34 of file `cf_cfdp.h`.

### 11.21.2 Field Documentation

#### 11.21.2.1 `chan CF_Channel_t* CF_CFDP_Tick_args::chan`

channel structure

Definition at line 36 of file `cf_cfdp.h`.

Referenced by `CF_CFDP_DoTick()`, and `CF_CFDP_TickTransactions()`.

#### 11.21.2.2 `fn void(* CF_CFDP_Tick_args::fn) (CF_Transaction_t *)`

function pointer

Definition at line 38 of file `cf_cfdp.h`.

Referenced by `CF_CFDP_DoTick()`, and `CF_CFDP_TickTransactions()`.

#### 11.21.2.3 `resume_point const CF_Transaction_t* CF_CFDP_Tick_args::resume_point`

skip to this txn (if not null)

Definition at line 37 of file `cf_cfdp.h`.

Referenced by `CF_CFDP_DoTick()`, and `CF_CFDP_TickTransactions()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp.h`

## 11.22 CF\_CFDP\_tlv Struct Reference

Structure representing CFDP TLV Object format.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `CF_CFDP_uint8_t type`

*Nature of data field.*

- `CF_CFDP_uint8_t length`

*Length of data field.*

### 11.22.1 Detailed Description

Structure representing CFDP TLV Object format.

These Type + Length + Value pairs used in several CFDP PDU types, typically for file storage requests (section 5.4).

Defined per table 5-3 of CCSDS 727.0-B-5

Definition at line 177 of file `cf_cfdp_pdu.h`.

### 11.22.2 Field Documentation

**11.22.2.1 length** `CF_CFDP_uint8_t` `CF_CFDP_tlv::length`

Length of data field.

Definition at line 180 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeTLV(), and CF\_CFDP\_EncodeTLV().

**11.22.2.2 type** `CF_CFDP_uint8_t` `CF_CFDP_tlv::type`

Nature of data field.

Definition at line 179 of file cf\_cfdp\_pdu.h.

Referenced by CF\_CFDP\_DecodeTLV(), and CF\_CFDP\_EncodeTLV().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_pdu.h](#)

**11.23 CF\_CFDP\_TxnRecvDispatchTable\_t Struct Reference**

A table of receive handler functions based on transaction state.

```
#include <cf_cfdp_dispatch.h>
```

**Data Fields**

- `CF_CFDP_StateRecvFunc_t rx [CF_TxnState_INVALID]`  
*a separate recv handler for each possible file directive PDU in this state*

**11.23.1 Detailed Description**

A table of receive handler functions based on transaction state.

This reflects the main dispatch table for the receive side of a transaction. Each possible state has a corresponding function pointer in the table to implement the PDU receive action(s) associated with that state.

Definition at line 74 of file cf\_cfdp\_dispatch.h.

**11.23.2 Field Documentation****11.23.2.1 rx** `CF_CFDP_StateRecvFunc_t` `CF_CFDP_TxnRecvDispatchTable_t::rx [CF_TxnState_INVALID]`

a separate recv handler for each possible file directive PDU in this state

Definition at line 77 of file cf\_cfdp\_dispatch.h.

Referenced by CF\_CFDP\_DispatchRecv(), and CF\_CFDP\_RxStateDispatch().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_dispatch.h](#)

**11.24 CF\_CFDP\_TxnSendDispatchTable\_t Struct Reference**

A table of transmit handler functions based on transaction state.

```
#include <cf_cfdp_dispatch.h>
```

**Data Fields**

- `CF_CFDP_StateSendFunc_t tx [CF_TxnState_INVALID]`  
*Transmit handler function.*

### 11.24.1 Detailed Description

A table of transmit handler functions based on transaction state.

This reflects the main dispatch table for the transmit side of a transaction. Each possible state has a corresponding function pointer in the table to implement the PDU transmit action(s) associated with that state.

Definition at line 62 of file cf\_cfdp\_dispatch.h.

### 11.24.2 Field Documentation

#### 11.24.2.1 tx CF\_CFDP\_StateSendFunc\_t CF\_CFDP\_TxnSendDispatchTable\_t::tx[CF\_TxnState\_INVALID]

Transmit handler function.

Definition at line 64 of file cf\_cfdp\_dispatch.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_dispatch.h](#)

## 11.25 CF\_CFDP\_uint16\_t Struct Reference

Encoded 16-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- [uint8 octets \[2\]](#)

### 11.25.1 Detailed Description

Encoded 16-bit value in the CFDP PDU.

Definition at line 103 of file cf\_cfdp\_pdu.h.

### 11.25.2 Field Documentation

#### 11.25.2.1 octets uint8 CF\_CFDP\_uint16\_t::octets[2]

Definition at line 105 of file cf\_cfdp\_pdu.h.

Referenced by CF\_Codec\_Load\_uint16(), and CF\_Codec\_Store\_uint16().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_pdu.h](#)

## 11.26 CF\_CFDP\_uint32\_t Struct Reference

Encoded 32-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- [uint8 octets \[4\]](#)

### 11.26.1 Detailed Description

Encoded 32-bit value in the CFDP PDU.

Definition at line 111 of file cf\_cfdp\_pdu.h.

## 11.26.2 Field Documentation

### 11.26.2.1 octets `uint8 CF_CFDP_uint32_t::octets[4]`

Definition at line 113 of file cf\_cfdp\_pdu.h.

Referenced by CF\_Codec\_Load\_uint32(), and CF\_Codec\_Store\_uint32().

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp_pdu.h`

## 11.27 CF\_CFDP\_uint64\_t Struct Reference

Encoded 64-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `uint8 octets [8]`

### 11.27.1 Detailed Description

Encoded 64-bit value in the CFDP PDU.

Definition at line 119 of file cf\_cfdp\_pdu.h.

## 11.27.2 Field Documentation

### 11.27.2.1 octets `uint8 CF_CFDP_uint64_t::octets[8]`

Definition at line 121 of file cf\_cfdp\_pdu.h.

Referenced by CF\_Codec\_Load\_uint64(), and CF\_Codec\_Store\_uint64().

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp_pdu.h`

## 11.28 CF\_CFDP\_uint8\_t Struct Reference

Encoded 8-bit value in the CFDP PDU.

```
#include <cf_cfdp_pdu.h>
```

### Data Fields

- `uint8 octets [1]`

### 11.28.1 Detailed Description

Encoded 8-bit value in the CFDP PDU.

Definition at line 95 of file cf\_cfdp\_pdu.h.

## 11.28.2 Field Documentation

**11.28.2.1 octets** `uint8 CF_CFDP_uint8_t::octets[1]`

Definition at line 97 of file cf\_cfdp\_pdu.h.

Referenced by CF\_Codec\_Load\_uint8(), and CF\_Codec\_Store\_uint8().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_pdu.h`

## 11.29 CF\_ChAction\_BoolArg Struct Reference

An object to use with channel-scope actions requiring only a boolean argument.

```
#include <cf_cmd.h>
```

### Data Fields

- `bool barg`

#### 11.29.1 Detailed Description

An object to use with channel-scope actions requiring only a boolean argument.

Definition at line 58 of file cf\_cmd.h.

#### 11.29.2 Field Documentation

**11.29.2.1 barg** `bool CF_ChAction_BoolArg::barg`

Definition at line 60 of file cf\_cmd.h.

Referenced by CF\_DoEnableDisableDequeue(), and CF\_DoFreezeThaw().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cmd.h`

## 11.30 CF\_ChAction\_BoolMsgArg Struct Reference

An object to use with channel-scope actions that require the message value.

```
#include <cf_cmd.h>
```

### Data Fields

- `const CF_UnionArgs_Payload_t * data`
- `bool barg`

#### 11.30.1 Detailed Description

An object to use with channel-scope actions that require the message value.

This combines a boolean action arg with the command message value

Definition at line 87 of file cf\_cmd.h.

#### 11.30.2 Field Documentation

**11.30.2.1 barg** `bool CF_ChAction_BoolMsgArg::barg`

Definition at line 90 of file cf\_cmd.h.

Referenced by CF\_DoEnableDisablePolldir().

**11.30.2.2 data** const [CF\\_UnionArgs\\_Payload\\_t](#)\* CF\_ChanAction\_BoolMsgArg::data  
Definition at line 89 of file cf\_cmd.h.

Referenced by CF\_DoEnableDisablePolldir().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cmd.h](#)

## 11.31 CF\_ChanAction\_MsgArg Struct Reference

An object to use with channel-scope actions that require the message value.

```
#include <cf_cmd.h>
```

### Data Fields

- const [CF\\_UnionArgs\\_Payload\\_t](#) \* **data**

#### 11.31.1 Detailed Description

An object to use with channel-scope actions that require the message value.

This combines a boolean action arg with the command message value

Definition at line 98 of file cf\_cmd.h.

#### 11.31.2 Field Documentation

**11.31.2.1 data** const [CF\\_UnionArgs\\_Payload\\_t](#)\* CF\_ChanAction\_MsgArg::data

Definition at line 100 of file cf\_cmd.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cmd.h](#)

## 11.32 CF\_ChanAction\_SuspResArg Struct Reference

An object to use with channel-scope actions for suspend/resume.

```
#include <cf_cmd.h>
```

### Data Fields

- int **same**
- bool **action**

#### 11.32.1 Detailed Description

An object to use with channel-scope actions for suspend/resume.

This combines a boolean action arg with an output that indicates if it was a change or not.

Definition at line 76 of file cf\_cmd.h.

#### 11.32.2 Field Documentation

**11.32.2.1 action** bool CF\_ChanAction\_SuspResArg::action

Definition at line 79 of file cf\_cmd.h.

Referenced by CF\_DoSuspRes\_Txn().

**11.32.2.2 same** int CF\_ChanAction\_SuspResArg::same  
out param – indicates at least one action was set to its current value  
Definition at line 78 of file cf\_cmd.h.  
Referenced by CF\_DoSuspRes(), and CF\_DoSuspRes\_Txn().  
The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cmd.h](#)

## 11.33 CF\_Channel Struct Reference

Channel state object.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- [CF\\_CListNode\\_t \\* qs \[CF\\_QueueIdx\\_NUM\]](#)
- [CF\\_CListNode\\_t \\* cs \[CF\\_Direction\\_NUM\]](#)
- [CFE\\_SB\\_Pipeld\\_t pipe](#)
- [uint32 num\\_cmd\\_tx](#)
- [CF\\_Playback\\_t playback \[CF\\_MAX\\_COMMANDED\\_PLAYBACK\\_DIRECTORIES\\_PER\\_CHAN\]](#)
- [CF\\_Poll\\_t poll \[CF\\_MAX\\_POLLING\\_DIR\\_PER\\_CHAN\]](#)
- [osal\\_id\\_t sem\\_id](#)  
*semaphore id for output pipe*
- [uint32 outgoing\\_counter](#)
- [const CF\\_Transaction\\_t \\* tick\\_resume](#)
- [bool tx\\_blocked](#)

### 11.33.1 Detailed Description

Channel state object.

This keeps the state of CF channels

Each CF channel has a separate transaction list, PDU throttle, playback, and poll state, as well as separate addresses on the underlying message transport (e.g. SB).

Definition at line 381 of file cf\_cfdp\_types.h.

### 11.33.2 Field Documentation

#### 11.33.2.1 cs [CF\\_CListNode\\_t\\* CF\\_Channel::cs\[CF\\_Direction\\_NUM\]](#)

Definition at line 384 of file cf\_cfdp\_types.h.

Referenced by CF\_GetChunkListHead().

#### 11.33.2.2 num\_cmd\_tx [uint32 CF\\_Channel::num\\_cmd\\_tx](#)

Definition at line 388 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), and CF\_CFDP\_TxFile().

#### 11.33.2.3 outgoing\_counter [uint32 CF\\_Channel::outgoing\\_counter](#)

Definition at line 397 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CycleEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_S\_Tick\_NewData(), and CF\_CFDP\_TickTransactions().

**11.33.2.4 pipe** [CFE\\_SB\\_PipeId\\_t](#) CF\_Channel::pipe

Definition at line 386 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_DisableEngine(), CF\_CFDP\_InitEngine(), and CF\_CFDP\_ReceiveMessage().

**11.33.2.5 playback** [CF\\_Playback\\_t](#) CF\_Channel::playback[[CF\\_MAX\\_COMMANDED\\_PLAYBACK\\_DIRECTORIES\\_PER\\_CHAN](#)]

Definition at line 390 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_DisableEngine(), CF\_CFDP\_PlaybackDir(), and CF\_CFDP\_ProcessPlaybackDirectories().

**11.33.2.6 poll** [CF\\_Poll\\_t](#) CF\_Channel::poll[[CF\\_MAX\\_POLLING\\_DIR\\_PER\\_CHAN](#)]

Definition at line 393 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_DisableEngine(), and CF\_CFDP\_ProcessPollingDirectories().

**11.33.2.7 qs** [CF\\_CListNode\\_t\\*](#) CF\_Channel::qs[[CF\\_QueueIdx\\_NUM](#)]

Definition at line 383 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_DisableEngine(), CF\_CFDP\_StartFirstPending(), CF\_CFDP\_TickTransactions(), CF\_CListInsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), CF\_DoPurgeQueue(), CF\_FindTransactionBySequenceNumber(), CF\_FindUnusedTransaction(), CF\_InsertSortPrio(), CF\_MoveTransaction(), CF\_TraverseAllTransactions(), CF\_WriteHistoryQueueDataToFile(), and CF\_WriteTxnQueueDataToFile().

**11.33.2.8 sem\_id** [osal\\_id\\_t](#) CF\_Channel::sem\_id

semaphore id for output pipe

Definition at line 395 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_InitEngine(), and CF\_CFDP\_MsgOutGet().

**11.33.2.9 tick\_resume** const [CF\\_Transaction\\_t\\*](#) CF\_Channel::tick\_resume

Definition at line 404 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CompleteTick(), and CF\_CFDP\_TickTransactions().

**11.33.2.10 tx\_blocked** bool CF\_Channel::tx\_blocked

Set true if PDU transmission was blocked due to limits

Definition at line 406 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CompleteTick(), CF\_CFDP\_CycleEngine(), CF\_CFDP\_DoTick(), CF\_CFDP\_MsgOutGet(), and CF\_CFDP\_TickTransactions().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf\\_cfdp\\_types.h](#)

**11.34 CF\_ChannelConfig Struct Reference**

Configuration entry for CFDP channel.

#include &lt;default\_cf\_tbldefs.h&gt;

**Data Fields**

- [uint32 max\\_outgoing\\_messages\\_per\\_wakeup](#)

*max number of messages to send per wakeup (0 - unlimited)*

- `uint32 rx_max_messages_per_wakeup`  
*max number of rx messages to process per wakeup*
- `uint32 ack_timer_s`  
*Acknowledge timer in seconds.*
- `uint32 nak_timer_s`  
*Non-acknowledge timer in seconds.*
- `uint32 inactivity_timer_s`  
*Inactivity timer in seconds.*
- `uint8 ack_limit`
- `uint8 nak_limit`
- `CFE_SB_MsgId_Atom_t mid_input`  
*msgid integer value for incoming messages*
- `CFE_SB_MsgId_Atom_t mid_output`  
*msgid integer value for outgoing messages*
- `uint16 pipe_depth_input`  
*depth of pipe to receive incoming PDU*
- `CF_PollDir_t polldir [CF_MAX_POLLING_DIR_PER_CHAN]`  
*Configuration for polled directories.*
- `char sem_name [OS_MAX_API_NAME]`  
*name of throttling semaphore in TO*
- `uint8 dequeue_enabled`  
*if 1, then the channel will make pending transactions active*
- `char move_dir [OS_MAX_PATH_LEN]`  
*Move directory if not empty.*

### 11.34.1 Detailed Description

Configuration entry for CFDP channel.

Definition at line 57 of file default\_cf\_tbldefs.h.

### 11.34.2 Field Documentation

#### 11.34.2.1 `ack_limit uint8 CF_ChannelConfig::ack_limit`

number of times to retry ACK (for ex, send FIN and wait for fin-ack)

Definition at line 66 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_CheckAckNakCount()`, and `CF_GetSetParamCmd()`.

#### 11.34.2.2 `ack_timer_s uint32 CF_ChannelConfig::ack_timer_s`

Acknowledge timer in seconds.

Definition at line 62 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_ArmAckTimer()`, `CF_CFDP_ArmInactTimer()`, and `CF_GetSetParamCmd()`.

#### 11.34.2.3 `dequeue_enabled uint8 CF_ChannelConfig::dequeue_enabled`

if 1, then the channel will make pending transactions active

Definition at line 77 of file default\_cf\_tbldefs.h.

Referenced by `CF_DoEnableDisableDequeue()`.

**11.34.2.4 inactivity\_timer\_s** `uint32` `CF_ChannelConfig::inactivity_timer_s`  
Inactivity timer in seconds.

Definition at line 64 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_ArmlnactTimer()`, and `CF_GetSetParamCmd()`.

**11.34.2.5 max\_outgoing\_messages\_per\_wakeup** `uint32` `CF_ChannelConfig::max_outgoing_messages_per_wakeup`

max number of messages to send per wakeup (0 - unlimited)

Definition at line 59 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_MsgOutGet()`, and `CF_GetSetParamCmd()`.

**11.34.2.6 mid\_input** `CFE_SB_MsgId_Atom_t` `CF_ChannelConfig::mid_input`

msgid integer value for incoming messages

Definition at line 69 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_InitEngine()`.

**11.34.2.7 mid\_output** `CFE_SB_MsgId_Atom_t` `CF_ChannelConfig::mid_output`

msgid integer value for outgoing messages

Definition at line 70 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_MsgOutGet()`.

**11.34.2.8 move\_dir** `char` `CF_ChannelConfig::move_dir[OS_MAX_PATH_LEN]`

Move directory if not empty.

Definition at line 78 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_S_HandleFileRetention()`.

**11.34.2.9 nak\_limit** `uint8` `CF_ChannelConfig::nak_limit`

number of times to retry NAK before giving up (resets on a single response)

Definition at line 67 of file default\_cf\_tbldefs.h.

Referenced by `CF_GetSetParamCmd()`.

**11.34.2.10 nak\_timer\_s** `uint32` `CF_ChannelConfig::nak_timer_s`

Non-acknowledge timer in seconds.

Definition at line 63 of file default\_cf\_tbldefs.h.

Referenced by `CF_GetSetParamCmd()`.

**11.34.2.11 pipe\_depth\_input** `uint16` `CF_ChannelConfig::pipe_depth_input`

depth of pipe to receive incoming PDU

Definition at line 72 of file default\_cf\_tbldefs.h.

Referenced by `CF_CFDP_InitEngine()`.

**11.34.2.12 polldir** `CF_PollDir_t` `CF_ChannelConfig::polldir[CF_MAX_POLLING_DIR_PER_CHAN]`

Configuration for polled directories.

Definition at line 74 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories(), and CF\_DoEnableDisablePolldir().

**11.34.2.13 rx\_max\_messages\_per\_wakeup** `uint32` `CF_ChannelConfig::rx_max_messages_per_wakeup`  
max number of rx messages to process per wakeup  
Definition at line 60 of file default\_cf\_tbldefs.h.  
Referenced by CF\_CFDP\_ReceiveMessage().

**11.34.2.14 sem\_name** `char` `CF_ChannelConfig::sem_name[OS_MAX_API_NAME]`  
name of throttling semaphore in TO  
Definition at line 76 of file default\_cf\_tbldefs.h.  
Referenced by CF\_CFDP\_InitEngine(), and CF\_ValidateMaxOutgoingCmd().  
The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_tbldefs.h`

## 11.35 CF\_Chunk Struct Reference

Pairs an offset with a size to identify a specific piece of a file.

```
#include <cf_chunk.h>
```

### Data Fields

- **CF\_Offset\_t offset**  
*The start offset of the chunk within the file.*
- **CF\_Size\_t size**  
*The size of the chunk.*

### 11.35.1 Detailed Description

Pairs an offset with a size to identify a specific piece of a file.

Definition at line 38 of file cf\_chunk.h.

### 11.35.2 Field Documentation

#### 11.35.2.1 offset `CF_Offset_t` `CF_Chunk::offset`

The start offset of the chunk within the file.

Definition at line 40 of file cf\_chunk.h.

Referenced by CF\_CFDP\_R2\_GapCompute(), CF\_CFDP\_S\_Tick\_Nak(), CF\_ChunkList\_ComputeGaps(), CF\_ChunkList\_RemoveFromFirst(), CF\_Chunks\_CombineNext(), CF\_Chunks\_CombinePrevious(), and CF\_Chunks\_FindInsertPosition().

#### 11.35.2.2 size `CF_Size_t` `CF_Chunk::size`

The size of the chunk.

Definition at line 41 of file cf\_chunk.h.

Referenced by CF\_CFDP\_R2\_GapCompute(), CF\_CFDP\_S\_Tick\_Nak(), CF\_ChunkList\_ComputeGaps(), CF\_ChunkList\_RemoveFromFirst(), CF\_Chunks\_CombineNext(), CF\_Chunks\_CombinePrevious(), CF\_Chunks\_FindSmallestSize(), and CF\_Chunks\_Insert().

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_chunk.h`

## 11.36 CF\_ChunkList Struct Reference

A list of CF\_Chunk\_t pairs.

```
#include <cf_chunk.h>
```

### Data Fields

- **CF\_ChunkIdx\_t count**  
*number of chunks currently in the array*
- **CF\_ChunkIdx\_t max\_chunks**  
*maximum number of chunks allowed in the list (allocation size)*
- **CF\_Chunk\_t \* chunks**  
*chunk list array*

### 11.36.1 Detailed Description

A list of CF\_Chunk\_t pairs.

This list is ordered by chunk offset, from lowest to highest

Definition at line 49 of file cf\_chunk.h.

### 11.36.2 Field Documentation

#### 11.36.2.1 chunks `CF_Chunk_t* CF_ChunkList::chunks`

chunk list array

Definition at line 53 of file cf\_chunk.h.

Referenced by CF\_ChunkList\_ComputeGaps(), CF\_ChunkList\_GetFirstChunk(), CF\_ChunkList\_RemoveFromFirst(), CF\_ChunkListInit(), CF\_ChunkListReset(), CF\_Chunks\_CombineNext(), CF\_Chunks\_CombinePrevious(), CF\_Chunks\_EraseChunk(), CF\_Chunks\_EraseRange(), CF\_Chunks\_FindInsertPosition(), CF\_Chunks\_FindSmallestSize(), CF\_Chunks\_Insert(), and CF\_Chunks\_InsertChunk().

#### 11.36.2.2 count `CF_ChunkIdx_t CF_ChunkList::count`

number of chunks currently in the array

Definition at line 51 of file cf\_chunk.h.

Referenced by CF\_CFDP\_R\_SendNak(), CF\_ChunkList\_ComputeGaps(), CF\_ChunkList\_GetFirstChunk(), CF\_ChunkListReset(), CF\_Chunks\_CombineNext(), CF\_Chunks\_EraseChunk(), CF\_Chunks\_EraseRange(), CF\_Chunks\_FindInsertPosition(), CF\_Chunks\_FindSmallestSize(), CF\_Chunks\_Insert(), and CF\_Chunks\_InsertChunk().

#### 11.36.2.3 max\_chunks `CF_ChunkIdx_t CF_ChunkList::max_chunks`

maximum number of chunks allowed in the list (allocation size)

Definition at line 52 of file cf\_chunk.h.

Referenced by CF\_CFDP\_R\_SendNak(), CF\_ChunkListInit(), CF\_ChunkListReset(), CF\_Chunks\_Insert(), and CF\_Chunks\_InsertChunk().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_chunk.h`

## 11.37 CF\_ChunkWrapper Struct Reference

Wrapper around a CF\_ChunkList\_t object.

```
#include <cf_cfdp_types.h>
```

**Data Fields**

- `CF_ChunkList_t chunks`
- `CF_CListNode_t cl_node`

**11.37.1 Detailed Description**

Wrapper around a `CF_ChunkList_t` object.

This allows a `CF_ChunkList_t` to be stored within a CList data storage structure

Definition at line 194 of file `cf_cfdp_types.h`.

**11.37.2 Field Documentation****11.37.2.1 chunks `CF_ChunkList_t` `CF_ChunkWrapper::chunks`**

Definition at line 196 of file `cf_cfdp_types.h`.

Referenced by `CF_CFDP_InitEngine()`, `CF_CFDP_R_CheckComplete()`, `CF_CFDP_R_ProcessFd()`, `CF_CFDP_R_SendNak()`, `CF_CFDP_S2_SubstateNak()`, and `CF_CFDP_S_Tick_Nak()`.

**11.37.2.2 cl\_node `CF_CListNode_t` `CF_ChunkWrapper::cl_node`**

Definition at line 197 of file `cf_cfdp_types.h`.

Referenced by `CF_CFDP_InitEngine()`, and `CF_CFDP_RecycleTransaction()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp_types.h`

**11.38 CF\_CListNode Struct Reference**

Node link structure.

```
#include <cf_clist.h>
```

**Data Fields**

- struct `CF_CListNode * next`
- struct `CF_CListNode * prev`

**11.38.1 Detailed Description**

Node link structure.

Definition at line 52 of file `cf_clist.h`.

**11.38.2 Field Documentation****11.38.2.1 next struct `CF_CListNode*` `CF_CListNode::next`**

Definition at line 54 of file `cf_clist.h`.

Referenced by `CF_CList_InitNode()`, `CF_CList_InsertAfter()`, `CF_CList_InsertBack()`, `CF_CList_InsertFront()`, `CF_CList_Remove()`, and `CF_CList_Traverse()`.

**11.38.2.2 prev** struct [CF\\_CListNode](#)\* CF\_CListNode::prev

Definition at line 55 of file cf\_clist.h.

Referenced by CF\_CList\_InitNode(), CF\_CList\_InsertAfter(), CF\_CList\_InsertBack(), CF\_CList\_InsertFront(), CF\_CList\_Remove(), and CF\_CList\_Traverse\_R().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_clist.h](#)

## 11.39 CF\_Codec\_BitField Struct Reference

### Data Fields

- [uint32 shift](#)
- [uint32 mask](#)

#### 11.39.1 Detailed Description

Definition at line 33 of file cf\_codec.c.

#### 11.39.2 Field Documentation

**11.39.2.1 mask** [uint32](#) CF\_Codec\_BitField::mask

Definition at line 36 of file cf\_codec.c.

**11.39.2.2 shift** [uint32](#) CF\_Codec\_BitField::shift

Definition at line 35 of file cf\_codec.c.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_codec.c](#)

## 11.40 CF\_CodecState Struct Reference

Tracks the current state of an encode or decode operation.

```
#include <cf_codec.h>
```

### Data Fields

- [bool is\\_valid](#)  
*whether decode is valid or not. Set false on end of decode or error condition.*
- [size\\_t next\\_offset](#)  
*Offset of next byte to encode/decode, current position in PDU.*
- [size\\_t max\\_size](#)  
*Maximum number of bytes in the PDU.*

#### 11.40.1 Detailed Description

Tracks the current state of an encode or decode operation.

This encapsulates the common state between encode and decode

Definition at line 38 of file cf\_codec.h.

### 11.40.2 Field Documentation

#### 11.40.2.1 `is_valid` `bool CF_CodecState::is_valid`

whether decode is valid or not. Set false on end of decode or error condition.

Definition at line 40 of file cf\_codec.h.

Referenced by CF\_CFDP\_CodecIsOK(), CF\_CFDP\_CodecReset(), and CF\_CFDP\_CodecSetDone().

#### 11.40.2.2 `max_size` `size_t CF_CodecState::max_size`

Maximum number of bytes in the PDU.

Definition at line 42 of file cf\_codec.h.

Referenced by CF\_CFDP\_CodecCheckSize(), CF\_CFDP\_CodecGetRemain(), CF\_CFDP\_CodecGetSize(), CF\_CFDP\_CodecReset(), CF\_CFDP\_DecodeStart(), and CF\_CFDP\_EncodeStart().

#### 11.40.2.3 `next_offset` `size_t CF_CodecState::next_offset`

Offset of next byte to encode/decode, current position in PDU.

Definition at line 41 of file cf\_codec.h.

Referenced by CF\_CFDP\_CodecCheckSize(), CF\_CFDP\_CodecGetPosition(), CF\_CFDP\_CodecGetRemain(), and CF\_CFDP\_CodecReset().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf\\_codec.h](#)

## 11.41 CF\_ConfigTable Struct Reference

Top-level CFDP configuration structure.

```
#include <default_cf_tblstruct.h>
```

### Data Fields

- `uint32 ticks_per_second`  
*expected ticks per second to CFDP app*
- `uint32 rx_crc_calc_bytes_per_wakeup`  
*max number of bytes per wakeup to calculate r2 CRC for recv'd file (must be 1024-byte aligned)*
- `CF_EntityId_t local_eid`  
*the local entity ID of the CF app*
- `CF_ChannelConfig_t chan [CF_NUM_CHANNELS]`  
*Channel configuration.*
- `uint16 outgoing_file_chunk_size`  
*maximum size of outgoing file data chunk in a PDU. Limited by CF\_MAX\_PDU\_SIZE minus the PDU header(s)*
- `char tmp_dir [CF_FILENAME_MAX_PATH]`  
*directory to put temp files*
- `char fail_dir [CF_FILENAME_MAX_PATH]`  
*fail directory*

### 11.41.1 Detailed Description

Top-level CFDP configuration structure.

Definition at line 40 of file default\_cf\_tblstruct.h.

### 11.41.2 Field Documentation

#### 11.41.2.1 chan `CF_ChannelConfig_t` CF\_ConfigTable:::chan[CF\_NUM\_CHANNELS]

Channel configuration.

Definition at line 49 of file default\_cf\_tblstruct.h.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_InitEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_ReceiveMessage(), CF\_CFDP\_S\_HandleFileRetention(), CF\_DoEnableDisableDequeue(), CF\_DoEnableDisablePolldir(), CF\_GetSetParamCmd(), and CF\_ValidateMaxOutgoingCmd().

#### 11.41.2.2 fail\_dir `char` CF\_ConfigTable:::fail\_dir[CF\_FILENAME\_MAX\_PATH]

fail directory

Definition at line 54 of file default\_cf\_tblstruct.h.

Referenced by CF\_CFDP\_S\_HandleFileRetention().

#### 11.41.2.3 local\_eid `CF_EntityId_t` CF\_ConfigTable:::local\_eid

the local entity ID of the CF app

Definition at line 47 of file default\_cf\_tblstruct.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_ReceivePdu(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_TxFile\_Initiate(), and CF\_GetSetParamCmd().

#### 11.41.2.4 outgoing\_file\_chunk\_size `uint16` CF\_ConfigTable:::outgoing\_file\_chunk\_size

maximum size of outgoing file data chunk in a PDU. Limited by CF\_MAX\_PDU\_SIZE minus the PDU header(s)

Definition at line 51 of file default\_cf\_tblstruct.h.

Referenced by CF\_CFDP\_S\_SendFileData(), CF\_GetSetParamCmd(), and CF\_ValidateConfigTable().

#### 11.41.2.5 rx\_crc\_calc\_bytes\_per\_wakeup `uint32` CF\_ConfigTable:::rx\_crc\_calc\_bytes\_per\_wakeup

max number of bytes per wakeup to calculate r2 CRC for recv'd file (must be 1024-byte aligned)

Definition at line 43 of file default\_cf\_tblstruct.h.

Referenced by CF\_GetSetParamCmd(), and CF\_ValidateConfigTable().

#### 11.41.2.6 ticks\_per\_second `uint32` CF\_ConfigTable:::ticks\_per\_second

expected ticks per second to CFDP app

Definition at line 42 of file default\_cf\_tblstruct.h.

Referenced by CF\_GetSetParamCmd(), CF\_Timer\_Sec2Ticks(), and CF\_ValidateConfigTable().

#### 11.41.2.7 tmp\_dir `char` CF\_ConfigTable:::tmp\_dir[CF\_FILENAME\_MAX\_PATH]

directory to put temp files

Definition at line 53 of file default\_cf\_tblstruct.h.

Referenced by CF\_CFDP\_GetTempName(), and CF\_CFDP\_InitEngine().

The documentation for this struct was generated from the following file:

- apps/cf/config/default\_cf\_tblstruct.h

## 11.42 CF\_Crc Struct Reference

CRC state object.

```
#include <cf_crc.h>
```

### Data Fields

- `uint32 working`
- `uint32 result`
- `uint8 index`

#### 11.42.1 Detailed Description

CRC state object.

Definition at line 34 of file cf\_crc.h.

#### 11.42.2 Field Documentation

##### 11.42.2.1 `index uint8 CF_Crc::index`

Definition at line 38 of file cf\_crc.h.

Referenced by `CF_CRC_Digest()`, and `CF_CRC_Finalize()`.

##### 11.42.2.2 `result uint32 CF_Crc::result`

Definition at line 37 of file cf\_crc.h.

Referenced by `CF_CFDP_R_CheckCrc()`, `CF_CFDP_SendEof()`, `CF_CFDP_SendEotPkt()`, `CF_CRC_Digest()`, and `CF_CRC_Finalize()`.

##### 11.42.2.3 `working uint32 CF_Crc::working`

Definition at line 36 of file cf\_crc.h.

Referenced by `CF_CRC_Digest()`, and `CF_CRC_Finalize()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_crc.h`

## 11.43 CF\_DecoderState Struct Reference

Current state of a decode operation.

```
#include <cf_codec.h>
```

### Data Fields

- `CF_CodecState_t codec_state`

*Common state.*

- `const uint8 * base`

*Pointer to start of encoded PDU data.*

#### 11.43.1 Detailed Description

Current state of a decode operation.

State structure for decodes

Definition at line 61 of file cf\_codec.h.

### 11.43.2 Field Documentation

#### 11.43.2.1 **base** const uint8\* CF\_DecoderState::base

Pointer to start of encoded PDU data.

Definition at line 64 of file cf\_codec.h.

Referenced by CF\_CFDP\_DecodeStart(), and CF\_CFDP\_DoDecodeChunk().

#### 11.43.2.2 **codec\_state** CF\_CodecState\_t CF\_DecoderState::codec\_state

Common state.

Definition at line 63 of file cf\_codec.h.

Referenced by CF\_CFDP\_DecodeStart(), and CF\_CFDP\_DoDecodeChunk().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf\\_codec.h](#)

## 11.44 CF\_DisableDequeueCmd Struct Reference

DisableDequeue command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_UnionArgs\\_Payload\\_t Payload](#)  
*Generic command arguments.*

#### 11.44.1 Detailed Description

DisableDequeue command structure.

For command details see [CF\\_DISABLE\\_DEQUEUE\\_CC](#)

Definition at line 148 of file default\_cf\_msgstruct.h.

### 11.44.2 Field Documentation

#### 11.44.2.1 **CommandHeader** CFE\_MSG\_CommandHeader\_t CF\_DisableDequeueCmd::CommandHeader

Command header.

Definition at line 150 of file default\_cf\_msgstruct.h.

#### 11.44.2.2 **Payload** CF\_UnionArgs\_Payload\_t CF\_DisableDequeueCmd::Payload

Generic command arguments.

Definition at line 151 of file default\_cf\_msgstruct.h.

Referenced by CF\_DisableDequeueCmd().

The documentation for this struct was generated from the following file:

- apps/cf/config/[default\\_cf\\_msgstruct.h](#)

## 11.45 CF\_DisableDirPollingCmd Struct Reference

DisableDirPolling command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_UnionArgs\\_Payload\\_t Payload](#)  
*Generic command arguments.*

### 11.45.1 Detailed Description

DisableDirPolling command structure.

For command details see [CF\\_DISABLE\\_DIR\\_POLLING\\_CC](#)

Definition at line 170 of file default\_cf\_msgstruct.h.

### 11.45.2 Field Documentation

#### 11.45.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_DisableDirPollingCmd::CommandHeader

Command header.

Definition at line 172 of file default\_cf\_msgstruct.h.

#### 11.45.2.2 Payload [CF\\_UnionArgs\\_Payload\\_t](#) CF\_DisableDirPollingCmd::Payload

Generic command arguments.

Definition at line 173 of file default\_cf\_msgstruct.h.

Referenced by [CF\\_DisableDirPollingCmd\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.46 CF\_DisableEngineCmd Struct Reference

DisableEngine command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*

### 11.46.1 Detailed Description

DisableEngine command structure.

For command details see [CF\\_DISABLE\\_ENGINE\\_CC](#)

Definition at line 94 of file default\_cf\_msgstruct.h.

### 11.46.2 Field Documentation

**11.46.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CF_DisableEngineCmd::CommandHeader`  
Command header.

Definition at line 96 of file default\_cf\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.47 CF\_EnableDequeueCmd Struct Reference

EnableDequeue command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CF_UnionArgs_Payload_t Payload`  
*Generic command arguments.*

### 11.47.1 Detailed Description

EnableDequeue command structure.

For command details see [CF\\_ENABLE\\_DEQUEUE\\_CC](#)

Definition at line 137 of file default\_cf\_msgstruct.h.

### 11.47.2 Field Documentation

**11.47.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CF_EnableDequeueCmd::CommandHeader`

Command header.

Definition at line 139 of file default\_cf\_msgstruct.h.

**11.47.2.2 Payload** `CF_UnionArgs_Payload_t` `CF_EnableDequeueCmd::Payload`

Generic command arguments.

Definition at line 140 of file default\_cf\_msgstruct.h.

Referenced by `CF_EnableDequeueCmd()`.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.48 CF\_EnableDirPollingCmd Struct Reference

EnableDirPolling command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CF_UnionArgs_Payload_t Payload`  
*Generic command arguments.*

### 11.48.1 Detailed Description

EnableDirPolling command structure.

For command details see [CF\\_ENABLE\\_DIR\\_POLLING\\_CC](#)

Definition at line 159 of file default\_cf\_msgstruct.h.

### 11.48.2 Field Documentation

#### 11.48.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_EnableDirPollingCmd::CommandHeader

Command header.

Definition at line 161 of file default\_cf\_msgstruct.h.

#### 11.48.2.2 Payload [CF\\_UnionArgs\\_Payload\\_t](#) CF\_EnableDirPollingCmd::Payload

Generic command arguments.

Definition at line 162 of file default\_cf\_msgstruct.h.

Referenced by CF\_EnableDirPollingCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.49 CF\_EnableEngineCmd Struct Reference

EnableEngine command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*

### 11.49.1 Detailed Description

EnableEngine command structure.

For command details see [CF\\_ENABLE\\_ENGINE\\_CC](#)

Definition at line 84 of file default\_cf\_msgstruct.h.

### 11.49.2 Field Documentation

#### 11.49.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_EnableEngineCmd::CommandHeader

Command header.

Definition at line 86 of file default\_cf\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.50 CF\_EncoderState Struct Reference

Current state of an encode operation.

```
#include <cf_codec.h>
```

**Data Fields**

- `CF_CodecState_t codec_state`  
*Common state.*
- `uint8 * base`  
*Pointer to start of encoded PDU data.*

**11.50.1 Detailed Description**

Current state of an encode operation.

State structure for encodes

Definition at line 50 of file cf\_codec.h.

**11.50.2 Field Documentation****11.50.2.1 base `uint8* CF_EncoderState::base`**

Pointer to start of encoded PDU data.

Definition at line 53 of file cf\_codec.h.

Referenced by CF\_CFDP\_DoEncodeChunk(), CF\_CFDP\_EncodeHeaderFinalSize(), and CF\_CFDP\_EncodeStart().

**11.50.2.2 codec\_state `CF_CodecState_t CF_EncoderState::codec_state`**

Common state.

Definition at line 52 of file cf\_codec.h.

Referenced by CF\_CFDP\_DoEncodeChunk(), and CF\_CFDP\_EncodeStart().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_codec.h`

**11.51 CF\_Engine Struct Reference**

An engine represents a pairing to a local EID.

```
#include <cf_cfdp_types.h>
```

**Data Fields**

- `CF_TransactionSeq_t seq_num`
- `CF_Output_t out`
- `CF_Input_t in`
- `CF_Transaction_t transactions [CF_NUM_TRANSACTIONS]`
- `CF_History_t histories [CF_NUM_HISTORIES]`
- `CF_Channel_t channels [CF_NUM_CHANNELS]`
- `CF_ChunkWrapper_t chunks [CF_NUM_TRANSACTIONS *CF_Direction_NUM]`
- `CF_Chunk_t chunk_mem [CF_NUM_CHUNKS_ALL_CHANNELS]`
- `bool enabled`

**11.51.1 Detailed Description**

An engine represents a pairing to a local EID.

Each engine can have at most CF\_MAX\_SIMULTANEOUS\_TRANSACTIONS

Definition at line 439 of file cf\_cfdp\_types.h.

## 11.51.2 Field Documentation

### 11.51.2.1 channels `CF_Channel_t CF_Engine::channels[CF_NUM_CHANNELS]`

Definition at line 449 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CycleEngine(), CF\_CFDP\_DisableEngine(), CF\_CFDP\_InitEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_PlaybackDir(), CF\_CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_ReceiveMessage(), CF\_CFDP\_ReceivePdu(), CF\_CFDP\_StartRxTransaction(), CF\_CFDP\_TxFile(), CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), CF\_DoPurgeQueue(), CF\_FindTransactionBySequenceNumberAllChannels(), CF\_FreeTransaction(), CF\_GetChannelFromTxn(), CF\_InsertSortPrio(), CF\_MoveTransaction(), CF\_TraverseAllTransactionsAll\_Channels(), CF\_TsnChanAction(), and CF\_WriteQueueCmd().

### 11.51.2.2 chunk\_mem `CF_Chunk_t CF_Engine::chunk_mem[CF_NUM_CHUNKS_ALL_CHANNELS]`

Definition at line 452 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_InitEngine().

### 11.51.2.3 chunks `CF_ChunkWrapper_t CF_Engine::chunks[CF_NUM_TRANSACTIONS *CF_Direction_NUM]`

Definition at line 451 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_InitEngine().

### 11.51.2.4 enabled `bool CF_Engine::enabled`

Definition at line 454 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CycleEngine(), CF\_CFDP\_DisableEngine(), CF\_CFDP\_InitEngine(), CF\_CheckTables(), CF\_DisableEngineCmd(), and CF\_EnableEngineCmd().

### 11.51.2.5 histories `CF_History_t CF_Engine::histories[CF_NUM_HISTORIES]`

Definition at line 448 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_InitEngine().

### 11.51.2.6 in `CF_Input_t CF_Engine::in`

Definition at line 444 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ReceiveMessage().

### 11.51.2.7 out `CF_Output_t CF_Engine::out`

Definition at line 443 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_MsgOutGet(), and CF\_CFDP\_Send().

### 11.51.2.8 seq\_num `CF_TransactionSeq_t CF_Engine::seq_num`

Definition at line 441 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_TxFile\_Initiate().

**11.51.2.9 transactions** `CF_Transaction_t` `CF_Engine::transactions[CF_NUM_TRANSACTIONS]`

Definition at line 447 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_InitEngine().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.52 CF\_EotPacket Struct Reference

End of transaction packet.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`  
*Telemetry header.*
- `CF_EotPacket_Payload_t Payload`

### 11.52.1 Detailed Description

End of transaction packet.

Definition at line 56 of file default\_cf\_msgstruct.h.

### 11.52.2 Field Documentation

#### 11.52.2.1 Payload `CF_EotPacket_Payload_t` `CF_EotPacket::Payload`

Definition at line 59 of file default\_cf\_msgstruct.h.

Referenced by CF\_CFDP\_SendEotPkt().

#### 11.52.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CF_EotPacket::TelemetryHeader`

Telemetry header.

Definition at line 58 of file default\_cf\_msgstruct.h.

Referenced by CF\_CFDP\_SendEotPkt().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.53 CF\_EotPacket\_Payload Struct Reference

End of transaction packet.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- `CF_TransactionSeq_t seq_num`  
*transaction identifier, stays constant for entire transfer*
- `uint32 channel`  
*Channel number.*
- `uint32 direction`  
*direction of this transaction*
- `uint32 state`

*Transaction state.*

- **uint32 txn\_stat**  
*final status code of transaction (extended CFDP CC)*
- **CF\_EntityId\_t src\_eid**  
*the source eid of the transaction*
- **CF\_EntityId\_t peer\_eid**  
*peer\_eid is always the "other guy", same src\_eid for RX*
- **uint32 fsize**  
*File size.*
- **uint32 crc\_result**  
*CRC result.*
- **CF\_TxnFilenames\_t fnames**  
*file names associated with this transaction*

### 11.53.1 Detailed Description

End of transaction packet.

Definition at line 127 of file default\_cf\_msgdefs.h.

### 11.53.2 Field Documentation

#### 11.53.2.1 **channel** `uint32 CF_EotPacket_Payload::channel`

Channel number.

Definition at line 130 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_SendEotPkt().

#### 11.53.2.2 **crc\_result** `uint32 CF_EotPacket_Payload::crc_result`

CRC result.

Definition at line 137 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_SendEotPkt().

#### 11.53.2.3 **direction** `uint32 CF_EotPacket_Payload::direction`

direction of this transaction

Definition at line 131 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_SendEotPkt().

#### 11.53.2.4 **fnames** `CF_TxnFilenames_t CF_EotPacket_Payload::fnames`

file names associated with this transaction

Definition at line 138 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_SendEotPkt().

#### 11.53.2.5 **fsize** `uint32 CF_EotPacket_Payload::fsize`

File size.

Definition at line 136 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_SendEotPkt().

**11.53.2.6 peer\_eid** `CF_EntityId_t` `CF_EotPacket_Payload::peer_eid`  
 peer\_eid is always the "other guy", same src\_eid for RX  
 Definition at line 135 of file default\_cf\_msgdefs.h.  
 Referenced by CF\_CFDP\_SendEotPkt().

**11.53.2.7 seq\_num** `CF_TransactionSeq_t` `CF_EotPacket_Payload::seq_num`  
 transaction identifier, stays constant for entire transfer  
 Definition at line 129 of file default\_cf\_msgdefs.h.  
 Referenced by CF\_CFDP\_SendEotPkt().

**11.53.2.8 src\_eid** `CF_EntityId_t` `CF_EotPacket_Payload::src_eid`  
 the source eid of the transaction  
 Definition at line 134 of file default\_cf\_msgdefs.h.  
 Referenced by CF\_CFDP\_SendEotPkt().

**11.53.2.9 state** `uint32` `CF_EotPacket_Payload::state`  
 Transaction state.  
 Definition at line 132 of file default\_cf\_msgdefs.h.  
 Referenced by CF\_CFDP\_SendEotPkt().

**11.53.2.10 txn\_stat** `uint32` `CF_EotPacket_Payload::txn_stat`  
 final status code of transaction (extended CFDP CC)  
 Definition at line 133 of file default\_cf\_msgdefs.h.  
 Referenced by CF\_CFDP\_SendEotPkt().  
 The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.54 CF\_Flags\_Common Struct Reference

Data that applies to all types of transactions.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- `uint8 q_index`  
*Q index this is in.*
- `bool close_req`
- `bool ack_timer_armed`
- `bool suspended`
- `bool canceled`
- `bool is_complete`
- `bool crc_complete`
- `bool inactivity_fired`  
*set whenever the inactivity timeout expires*
- `bool keep_history`  
*whether history should be preserved during recycle*

### 11.54.1 Detailed Description

Data that applies to all types of transactions.  
Definition at line 236 of file cf\_cfdp\_types.h.

### 11.54.2 Field Documentation

#### 11.54.2.1 **ack\_timer\_armed** bool CF\_Flags\_Common::ack\_timer\_armed

Definition at line 241 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_R2\_Recv(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_AckTimerTick(), and CF\_CFDP\_S\_CheckState().

#### 11.54.2.2 **canceled** bool CF\_Flags\_Common::canceled

Definition at line 243 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CancelTransaction().

#### 11.54.2.3 **close\_req** bool CF\_Flags\_Common::close\_req

Indicates if a FIN should be used in class 1 (optional)

Definition at line 240 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckState\_FILESTORE(), CF\_CFDP\_RecvMd(), CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), and CF\_CFDP\_SendMd().

#### 11.54.2.4 **crc\_complete** bool CF\_Flags\_Common::crc\_complete

Latches that the CRC computation is completed

Definition at line 245 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), and CF\_CFDP\_S\_CheckState().

#### 11.54.2.5 **inactivity\_fired** bool CF\_Flags\_Common::inactivity\_fired

set whenever the inactivity timeout expires

Definition at line 246 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_Tick(), and CF\_CFDP\_S\_Tick().

#### 11.54.2.6 **is\_complete** bool CF\_Flags\_Common::is\_complete

Latches that all expected PDUs (MD + all FD + EOF) are processed

Definition at line 244 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), and CF\_CFDP\_S\_HandleFileRetention().

#### 11.54.2.7 **keep\_history** bool CF\_Flags\_Common::keep\_history

whether history should be preserved during recycle

Definition at line 247 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), and CF\_CFDP\_RecycleTransaction().

**11.54.2.8 q\_index uint8 CF\_Flags\_Common::q\_index**

Q index this is in.

Definition at line 238 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), CF\_CFDP\_GetClass(), CF\_CFDP\_StartRxTransaction(), CF\_DequeueTransaction(), CF\_InsertSortPrio(), and CF\_MoveTransaction().

**11.54.2.9 suspended bool CF\_Flags\_Common::suspended**

Definition at line 242 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_DoTick(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_S\_Tick\_NewData(), and CF\_DoSuspRes\_Txn().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.55 CF\_Flags\_Rx Struct Reference

Flags that apply to receive transactions.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- [CF\\_Flags\\_Common\\_t com](#)
- [bool tempfile\\_created](#)
- [bool md\\_recv](#)
- [uint8 eof\\_count](#)
- [uint8 eof\\_ack\\_count](#)
- [bool finack\\_recv](#)
- [bool send\\_nak](#)
- [bool send\\_fin](#)

### 11.55.1 Detailed Description

Flags that apply to receive transactions.

Definition at line 253 of file cf\_cfdp\_types.h.

### 11.55.2 Field Documentation

**11.55.2.1 com CF\_Flags\_Common\_t CF\_Flags\_Rx::com**

Definition at line 255 of file cf\_cfdp\_types.h.

**11.55.2.2 eof\_ack\_count uint8 CF\_Flags\_Rx::eof\_ack\_count**

Count of EOF-ACKs sent to peer

Definition at line 261 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_Tick\_Maintenance().

**11.55.2.3 eof\_count** `uint8 CF_Flags_Rx::eof_count`

Count of EOF PDUs received

Definition at line 260 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_SubstateRecvEof(), and CF\_CFDP\_R\_Tick\_Maintenance().

**11.55.2.4 finack\_recv** `bool CF_Flags_Rx::finack_recv`

Latches that the fin-ack PDU is received

Definition at line 262 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R2\_SubstateRecvFinAck(), and CF\_CFDP\_R\_CheckState\_FINACK().

**11.55.2.5 md\_recv** `bool CF_Flags_Rx::md_recv`

Latches that the MD PDU is received

Definition at line 259 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_R\_SendNak(), and CF\_CFDP\_R\_SubstateRecvMd().

**11.55.2.6 send\_fin** `bool CF_Flags_Rx::send_fin`

Indicates need to send FIN to peer

Definition at line 265 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_FINACK(), and CF\_CFDP\_R\_Tick\_Maintenance().

**11.55.2.7 send\_nak** `bool CF_Flags_Rx::send_nak`

Indicates need to send NAK to peer

Definition at line 264 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), and CF\_CFDP\_R\_Tick\_Maintenance().

**11.55.2.8 tempfile\_created** `bool CF_Flags_Rx::tempfile_created`

Latches that the tempfile was created at txn start

Definition at line 257 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_HandleFileRetention(), and CF\_CFDP\_R\_Init().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf\\_cfdp\\_types.h](#)

## 11.56 CF\_Flags\_Tx Struct Reference

Flags that apply to send transactions.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- [CF\\_Flags\\_Common\\_t com](#)
- `bool cmd_tx`  
*indicates transaction is commanded (ground) tx*
- `bool fd_nak_pending`

- bool `eof_ack_recv`
- uint8 `fin_count`
- uint8 `fin_ack_count`
- bool `send_md`
- bool `send_eof`

### 11.56.1 Detailed Description

Flags that apply to send transactions.

Definition at line 271 of file cf\_cfdp\_types.h.

### 11.56.2 Field Documentation

#### 11.56.2.1 cmd\_tx bool CF\_Flags\_Tx::cmd\_tx

indicates transaction is commanded (ground) tx

Definition at line 275 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), CF\_CFDP\_S\_HandleFileRetention(), and CF\_CFDP\_TxFile().

#### 11.56.2.2 com CF\_Flags\_Common\_t CF\_Flags\_Tx::com

Definition at line 273 of file cf\_cfdp\_types.h.

#### 11.56.2.3 eof\_ack\_recv bool CF\_Flags\_Tx::eof\_ack\_recv

Latches that the EOF-ACK was received

Definition at line 278 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), and CF\_CFDP\_S2\_SubstateEofAck().

#### 11.56.2.4 fd\_nak\_pending bool CF\_Flags\_Tx::fd\_nak\_pending

Peer sent a NAK on file data

Definition at line 277 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_S2\_SubstateNak(), and CF\_CFDP\_S\_Tick\_Nak().

#### 11.56.2.5 fin\_ack\_count uint8 CF\_Flags\_Tx::fin\_ack\_count

Count of FIN-ACKs sent to peer

Definition at line 280 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), and CF\_CFDP\_S\_Tick\_Maintenance().

#### 11.56.2.6 fin\_count uint8 CF\_Flags\_Tx::fin\_count

Count of FIN PDUs received

Definition at line 279 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_SubstateRecvFin(), and CF\_CFDP\_S\_Tick\_Maintenance().

**11.56.2.7 send\_eof** `bool CF_Flags_Tx::send_eof`

Indicates need to send EOF to peer

Definition at line 283 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_CheckState\_←DATA\_EOF(), and CF\_CFDP\_S\_Tick\_Maintenance().

**11.56.2.8 send\_md** `bool CF_Flags_Tx::send_md`

Indicates need to send MD to peer

Definition at line 282 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_Init(), and CF\_CFDP\_S\_Tick\_Maintenance().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.57 CF\_FreezeCmd Struct Reference

Freeze command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_UnionArgs\\_Payload\\_t Payload](#)  
*Generic command arguments.*

### 11.57.1 Detailed Description

Freeze command structure.

For command details see [CF\\_FREEZE\\_CC](#)

Definition at line 115 of file default\_cf\_msgstruct.h.

### 11.57.2 Field Documentation

#### 11.57.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_FreezeCmd::CommandHeader

Command header.

Definition at line 117 of file default\_cf\_msgstruct.h.

#### 11.57.2.2 Payload [CF\\_UnionArgs\\_Payload\\_t](#) CF\_FreezeCmd::Payload

Generic command arguments.

Definition at line 118 of file default\_cf\_msgstruct.h.

Referenced by CF\_FreezeCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.58 CF\_GapComputeArgs\_t Struct Reference

Argument for Gap Compute function.

```
#include <cf_cfdp_r.h>
```

**Data Fields**

- [CF\\_Transaction\\_t \\* txn](#)  
*Current transaction being processed.*
- [CF\\_Logical\\_PduNak\\_t \\* nak](#)  
*Current NAK PDU contents.*

**11.58.1 Detailed Description**

Argument for Gap Compute function.

This is used in conjunction with CF\_CFDP\_R2\_GapCompute

Definition at line 39 of file cf\_cfdp\_r.h.

**11.58.2 Field Documentation****11.58.2.1 nak [CF\\_Logical\\_PduNak\\_t \\* CF\\_GapComputeArgs\\_t::nak](#)**

Current NAK PDU contents.

Definition at line 42 of file cf\_cfdp\_r.h.

Referenced by CF\_CFDP\_R2\_GapCompute().

**11.58.2.2 txn [CF\\_Transaction\\_t \\* CF\\_GapComputeArgs\\_t::txn](#)**

Current transaction being processed.

Definition at line 41 of file cf\_cfdp\_r.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf\\_cfdp\\_r.h](#)

**11.59 CF\_GetParam\_Payload Struct Reference**

Get parameter command structure.

```
#include <default_cf_msgdefs.h>
```

**Data Fields**

- [uint8 key](#)  
*Parameter key, see [CF\\_GetSet\\_ValueID\\_t](#).*
- [uint8 chan\\_num](#)  
*Channel number.*

**11.59.1 Detailed Description**

Get parameter command structure.

For command details see [CF\\_GET\\_PARAM\\_CC](#)

Definition at line 217 of file default\_cf\_msgdefs.h.

**11.59.2 Field Documentation**

**11.59.2.1 chan\_num** `uint8 CF_GetParam_Payload::chan_num`

Channel number.

Definition at line 220 of file default\_cf\_msgdefs.h.

Referenced by CF\_GetParamCmd().

**11.59.2.2 key** `uint8 CF_GetParam_Payload::key`

Parameter key, see [CF\\_SetValueID\\_t](#).

Definition at line 219 of file default\_cf\_msgdefs.h.

Referenced by CF\_GetParamCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.60 CF\_GetParamCmd Struct Reference

Get parameter command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_GetParam\\_Payload\\_t Payload](#)

#### 11.60.1 Detailed Description

Get parameter command structure.

For command details see [CF\\_GET\\_PARAM\\_CC](#)

Definition at line 192 of file default\_cf\_msgstruct.h.

#### 11.60.2 Field Documentation

**11.60.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_GetParamCmd::CommandHeader

Command header.

Definition at line 194 of file default\_cf\_msgstruct.h.

**11.60.2.2 Payload** [CF\\_GetParam\\_Payload\\_t](#) CF\_GetParamCmd::Payload

Definition at line 195 of file default\_cf\_msgstruct.h.

Referenced by CF\_GetParamCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.61 CF\_History Struct Reference

CF History entry.

```
#include <cf_cfdp_types.h>
```

**Data Fields**

- **CF\_TxnFilenames\_t fnames**  
*file names associated with this history entry*
- **CF\_CListNode\_t cl\_node**  
*for connection to a CList*
- **CF\_Direction\_t dir**  
*direction of this history entry*
- **CF\_TxnStatus\_t txn\_stat**  
*final status of operation*
- **CF\_EntityId\_t src\_eid**  
*the source eid of the transaction*
- **CF\_EntityId\_t peer\_eid**  
*peer\_eid is always the "other guy", same src\_eid for RX*
- **CF\_TransactionSeq\_t seq\_num**  
*transaction identifier, stays constant for entire transfer*

**11.61.1 Detailed Description**

CF History entry.

Records CF app operations for future reference

Definition at line 178 of file cf\_cfdp\_types.h.

**11.61.2 Field Documentation****11.61.2.1 cl\_node [CF\\_CListNode\\_t](#) CF\_History::cl\_node**

for connection to a CList

Definition at line 181 of file cf\_cfdp\_types.h.

Referenced by [CF\\_CFDP\\_InitEngine\(\)](#), [CF\\_CFDP\\_RecycleTransaction\(\)](#), [CF\\_FindUnusedTransaction\(\)](#), and [CF\\_ResetHistory\(\)](#).

**11.61.2.2 dir [CF\\_Direction\\_t](#) CF\_History::dir**

direction of this history entry

Definition at line 182 of file cf\_cfdp\_types.h.

Referenced by [CF\\_CFDP\\_AllocChunkList\(\)](#), [CF\\_CFDP\\_CheckAckNakCount\(\)](#), [CF\\_CFDP\\_FinishTransaction\(\)](#), [CF\\_CFDP\\_IsSender\(\)](#), [CF\\_CFDP\\_RecvHold\(\)](#), [CF\\_CFDP\\_RecycleTransaction\(\)](#), [CF\\_CFDP\\_SendEotPkt\(\)](#), [CF\\_FindUnusedTransaction\(\)](#), [CF\\_Traverse\\_WriteHistoryQueueEntryToFile\(\)](#), and [CF\\_WriteHistoryEntryToFile\(\)](#).

**11.61.2.3 fnames [CF\\_TxnFilenames\\_t](#) CF\_History::fnames**

file names associated with this history entry

Definition at line 180 of file cf\_cfdp\_types.h.

Referenced by [CF\\_CFDP\\_ProcessPlaybackDirectory\(\)](#), [CF\\_CFDP\\_R\\_HandleFileRetention\(\)](#), [CF\\_CFDP\\_RecvMd\(\)](#), [CF\\_CFDP\\_S\\_HandleFileRetention\(\)](#), [CF\\_CFDP\\_S\\_Init\(\)](#), [CF\\_CFDP\\_SendEotPkt\(\)](#), [CF\\_CFDP\\_SendMd\(\)](#), [CF\\_CFDP\\_TxFile\(\)](#), [CF\\_CFDP\\_TxFile\\_Initiate\(\)](#), and [CF\\_WriteHistoryEntryToFile\(\)](#).

**11.61.2.4 peer\_eid** [CF\\_EntityId\\_t](#) CF\_History::peer\_eid

peer\_eid is always the "other guy", same src\_eid for RX

Definition at line 185 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendEotPkt(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_SetupRxTransaction(), CF\_CFDP\_TxFile\_Initiate(), and CF\_WriteHistoryEntryToFile().

**11.61.2.5 seq\_num** [CF\\_TransactionSeq\\_t](#) CF\_History::seq\_num

transaction identifier, stays constant for entire transfer

Definition at line 186 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_AllocChunkList(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_GetTempName(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendEotPkt(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_SetupRxTransaction(), CF\_CFDP\_TxFile\_Initiate(), CF\_FindTransactionBySequenceNumber\_Impl(), and CF\_WriteHistoryEntryToFile().

**11.61.2.6 src\_eid** [CF\\_EntityId\\_t](#) CF\_History::src\_eid

the source eid of the transaction

Definition at line 184 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_GetTempName(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvMd(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_SendEotPkt(), CF\_CFDP\_SetupRxTransaction(), CF\_CFDP\_TxFile\_Initiate(), CF\_FindTransactionBySequenceNumber\_Impl(), and CF\_WriteHistoryEntryToFile().

**11.61.2.7 txn\_stat** [CF\\_TxnStatus\\_t](#) CF\_History::txn\_stat

final status of operation

Definition at line 183 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_GetTxnStatus(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendEotPkt(), CF\_CFDP\_SendFin(), CF\_CFDP\_SetTxnStatus(), and CF\_WriteHistoryEntryToFile().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf\\_cfdp\\_types.h](#)

## 11.62 CF\_HkChannel\_Data Struct Reference

Housekeeping channel data.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- [CF\\_HkCounters\\_t](#) counters  
*Counters.*
- [uint16](#) q\_size [[CF\\_QueueIdx\\_NUM](#)]

*Queue sizes.*

- **uint8 poll\_counter**

*Number of active polling directories.*

- **uint8 playback\_counter**

*Number of active playback directories.*

- **uint8 frozen**

*Frozen state: 0 == not frozen, else frozen.*

- **uint8 spare [7]**

*Alignment spare (uint64 values in the counters)*

### 11.62.1 Detailed Description

Housekeeping channel data.

Definition at line 103 of file default\_cf\_msgdefs.h.

### 11.62.2 Field Documentation

#### 11.62.2.1 **counters** `CF_HkCounters_t CF_HkChannel_Data::counters`

Counters.

Definition at line 105 of file default\_cf\_msgdefs.h.

Referenced by `CF_CFDP_CheckAckNakCount()`, `CF_CFDP_PlaybackDir_Initiate()`, `CF_CFDP_R2_SubstateRecvFinAck()`, `CF_CFDP_R_CalcCrcChunk()`, `CF_CFDP_R_CalcCrcStart()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_R_DispatchRecv()`, `CF_CFDP_R_Init()`, `CF_CFDP_R_ProcessFd()`, `CF_CFDP_R_SendNak()`, `CF_CFDP_R_SubstateRecvEof()`, `CF_CFDP_R_Tick()`, `CF_CFDP_RecvDrop()`, `CF_CFDP_RecvFd()`, `CF_CFDP_RecvHold()`, `CF_CFDP_RecvMd()`, `CF_CFDP_RecvPh()`, `CF_CFDP_S2_SubstateEofAck()`, `CF_CFDP_S2_SubstateNak()`, `CF_CFDP_S_DispatchRecv()`, `CF_CFDP_S_Init()`, `CF_CFDP_S_SendFileData()`, `CF_CFDP_S_Tick()`, `CF_CFDP_Send()`, and `CF_ResetCountersCmd()`.

#### 11.62.2.2 **frozen** `uint8 CF_HkChannel_Data::frozen`

Frozen state: 0 == not frozen, else frozen.

Definition at line 109 of file default\_cf\_msgdefs.h.

Referenced by `CF_CFDP_CycleEngine()`, `CF_CFDP_MsgOutGet()`, and `CF_DoFreezeThaw()`.

#### 11.62.2.3 **playback\_counter** `uint8 CF_HkChannel_Data::playback_counter`

Number of active playback directories.

Definition at line 108 of file default\_cf\_msgdefs.h.

Referenced by `CF_CFDP_ProcessPlaybackDirectories()`.

#### 11.62.2.4 **poll\_counter** `uint8 CF_HkChannel_Data::poll_counter`

Number of active polling directories.

Definition at line 107 of file default\_cf\_msgdefs.h.

Referenced by `CF_CFDP_ProcessPollingDirectories()`.

**11.62.2.5 q\_size** `uint16 CF_HkChannel_Data::q_size[CF_QueueIdx_NUM]`

Queue sizes.

Definition at line 106 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_DisableEngine(), CF\_CFDP\_StartRxTransaction(), CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), and CF\_MoveTransaction().

**11.62.2.6 spare** `uint8 CF_HkChannel_Data::spare[7]`

Alignment spare (uint64 values in the counters)

Definition at line 110 of file default\_cf\_msgdefs.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.63 CF\_HkCmdCounters Struct Reference

Housekeeping command counters.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- `uint16 cmd`  
*Command success counter.*
- `uint16 err`  
*Command error counter.*

### 11.63.1 Detailed Description

Housekeeping command counters.

Definition at line 39 of file default\_cf\_msgdefs.h.

### 11.63.2 Field Documentation

**11.63.2.1 cmd** `uint16 CF_HkCmdCounters::cmd`

Command success counter.

Definition at line 41 of file default\_cf\_msgdefs.h.

Referenced by CF\_AbandonCmd(), CF\_CancelCmd(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DisableEngineCmd(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(), CF\_FreezeCmd(), CF\_GetSetParamCmd(), CF\_NoopCmd(), CF\_PlaybackDirCmd(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_ThawCmd(), CF\_TxFileCmd(), and CF\_WriteQueueCmd().

**11.63.2.2 err** `uint16 CF_HkCmdCounters::err`

Command error counter.

Definition at line 42 of file default\_cf\_msgdefs.h.

Referenced by CF\_AbandonCmd(), CF\_AppPipe(), CF\_CancelCmd(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(), CF\_FreezeCmd(), CF\_GetSetParamCmd(), CF\_PlaybackDirCmd(), CF\_ProcessGroundCommand(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_ThawCmd(), CF\_TxFileCmd(), and CF\_WriteQueueCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.64 CF\_HkCounters Struct Reference

Housekeeping counters.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- [CF\\_HkSent\\_t sent](#)

*Sent counters.*

- [CF\\_HkRecv\\_t recv](#)

*Received counters.*

- [CF\\_HkFault\\_t fault](#)

*Fault counters.*

### 11.64.1 Detailed Description

Housekeeping counters.

Definition at line 93 of file default\_cf\_msgdefs.h.

### 11.64.2 Field Documentation

#### 11.64.2.1 [fault CF\\_HkFault\\_t](#) CF\_HkCounters::fault

Fault counters.

Definition at line 97 of file default\_cf\_msgdefs.h.

Referenced by [CF\\_CFDP\\_CheckAckNakCount\(\)](#), [CF\\_CFDP\\_PlaybackDir\\_Initiate\(\)](#), [CF\\_CFDP\\_R\\_CalcCrcChunk\(\)](#), [CF\\_CFDP\\_R\\_CalcCrcStart\(\)](#), [CF\\_CFDP\\_R\\_CheckCrc\(\)](#), [CF\\_CFDP\\_R\\_Init\(\)](#), [CF\\_CFDP\\_R\\_ProcessFd\(\)](#), [CF\\_CFDP\\_R\\_Tick\(\)](#), [CF\\_CFDP\\_S\\_Init\(\)](#), [CF\\_CFDP\\_S\\_SendFileData\(\)](#), [CF\\_CFDP\\_S\\_Tick\(\)](#), and [CF\\_ResetCountersCmd\(\)](#).

#### 11.64.2.2 [recv CF\\_HkRecv\\_t](#) CF\_HkCounters::recv

Received counters.

Definition at line 96 of file default\_cf\_msgdefs.h.

Referenced by [CF\\_CFDP\\_R2\\_SubstateRecvFinAck\(\)](#), [CF\\_CFDP\\_R\\_DispatchRecv\(\)](#), [CF\\_CFDP\\_R\\_ProcessFd\(\)](#), [CF\\_CFDP\\_R\\_SubstateRecvEof\(\)](#), [CF\\_CFDP\\_RecvDrop\(\)](#), [CF\\_CFDP\\_RecvFd\(\)](#), [CF\\_CFDP\\_RecvHold\(\)](#), [CF\\_CFDP\\_RecvMd\(\)](#), [CF\\_CFDP\\_RecvPh\(\)](#), [CF\\_CFDP\\_S2\\_SubstateEofAck\(\)](#), [CF\\_CFDP\\_S2\\_SubstateNak\(\)](#), [CF\\_CFDP\\_S\\_DispatchRecv\(\)](#), and [CF\\_ResetCountersCmd\(\)](#).

#### 11.64.2.3 [sent CF\\_HkSent\\_t](#) CF\_HkCounters::sent

Sent counters.

Definition at line 95 of file default\_cf\_msgdefs.h.

Referenced by [CF\\_CFDP\\_R\\_SendNak\(\)](#), [CF\\_CFDP\\_S\\_SendFileData\(\)](#), [CF\\_CFDP\\_Send\(\)](#), and [CF\\_ResetCountersCmd\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.65 CF\_HkFault Struct Reference

Housekeeping fault counters.

```
#include <default_cf_msgdefs.h>
```

## Data Fields

- `uint16 file_open`  
*File open fault counter.*
- `uint16 file_read`  
*File read fault counter.*
- `uint16 file_seek`  
*File seek fault counter.*
- `uint16 file_write`  
*File write fault counter.*
- `uint16 file_rename`  
*File rename fault counter.*
- `uint16 directory_read`  
*Directory read fault counter.*
- `uint16 crc_mismatch`  
*CRC mismatch fault counter.*
- `uint16 file_size_mismatch`  
*File size mismatch fault counter.*
- `uint16 nak_limit`  
*NAK limit exceeded fault counter.*
- `uint16 ack_limit`  
*ACK limit exceeded fault counter.*
- `uint16 inactivity_timer`  
*Inactivity timer exceeded counter.*
- `uint16 spare`  
*Alignment spare to avoid implicit padding.*

### 11.65.1 Detailed Description

Housekeeping fault counters.

Definition at line 74 of file default\_cf\_msgdefs.h.

### 11.65.2 Field Documentation

#### 11.65.2.1 `ack_limit` `uint16 CF_HkFault::ack_limit`

ACK limit exceeded fault counter.

Definition at line 85 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_CheckAckNakCount().

#### 11.65.2.2 `crc_mismatch` `uint16 CF_HkFault::crc_mismatch`

CRC mismatch fault counter.

Definition at line 82 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_CheckCrc().

**11.65.2.3 directory\_read** `uint16 CF_HkFault::directory_read`

Directory read fault counter.

Definition at line 81 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate().

**11.65.2.4 file\_open** `uint16 CF_HkFault::file_open`

File open fault counter.

Definition at line 76 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_Init(), and CF\_CFDP\_S\_Init().

**11.65.2.5 file\_read** `uint16 CF_HkFault::file_read`

File read fault counter.

Definition at line 77 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), and CF\_CFDP\_S\_SendFileData().

**11.65.2.6 file\_rename** `uint16 CF_HkFault::file_rename`

File rename fault counter.

Definition at line 80 of file default\_cf\_msgdefs.h.

**11.65.2.7 file\_seek** `uint16 CF_HkFault::file_seek`

File seek fault counter.

Definition at line 78 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_S\_Init(), and CF\_CFDP\_S\_SendFileData().

**11.65.2.8 file\_size\_mismatch** `uint16 CF_HkFault::file_size_mismatch`

File size mismatch fault counter.

Definition at line 83 of file default\_cf\_msgdefs.h.

**11.65.2.9 file\_write** `uint16 CF_HkFault::file_write`

File write fault counter.

Definition at line 79 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_ProcessFd().

**11.65.2.10 inactivity\_timer** `uint16 CF_HkFault::inactivity_timer`

Inactivity timer exceeded counter.

Definition at line 86 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_Tick(), and CF\_CFDP\_S\_Tick().

**11.65.2.11 nak\_limit** `uint16 CF_HkFault::nak_limit`

NAK limit exceeded fault counter.

Definition at line 84 of file default\_cf\_msgdefs.h.

**11.65.2.12 spare** `uint16 CF_HkFault::spare`

Alignment spare to avoid implicit padding.

Definition at line 87 of file default\_cf\_msgdefs.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.66 CF\_HkPacket Struct Reference

Housekeeping packet.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_TelemetryHeader_t TelemetryHeader`  
*Telemetry header.*
- `CF_HkPacket_Payload_t Payload`

### 11.66.1 Detailed Description

Housekeeping packet.

Definition at line 47 of file default\_cf\_msgstruct.h.

### 11.66.2 Field Documentation

#### 11.66.2.1 Payload

`CF_HkPacket_Payload_t CF_HkPacket::Payload`

Definition at line 50 of file default\_cf\_msgstruct.h.

Referenced by CF\_AbandonCmd(), CF\_AppPipe(), CF\_CancelCmd(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP->  
\_CycleEngine(), CF\_CFDP\_DisableEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_PlaybackDir\_Initiate(), CF->  
CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R2\_SubstateRecvFinAck(),  
CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_Dispatch->  
Recv(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecv->  
Eof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvDrop(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvHold(), CF\_CFDP->  
RecvMd(), CF\_CFDP\_RecvPh(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S->  
DispatchRecv(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_Send(), CF->  
CFDP\_StartRxTransaction(), CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF->  
DequeueTransaction(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DisableEngineCmd(), CF\_Do->  
FreezeThaw(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(),  
CF\_FreezeCmd(), CF\_GetSetParamCmd(), CF\_MoveTransaction(), CF\_NoopCmd(), CF\_PlaybackDirCmd(), CF->  
ProcessGroundCommand(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_ThawCmd(), CF\_TxFileCmd(),  
and CF\_WriteQueueCmd().

#### 11.66.2.2 TelemetryHeader

`CFE_MSG_TelemetryHeader_t CF_HkPacket::TelemetryHeader`

Telemetry header.

Definition at line 49 of file default\_cf\_msgstruct.h.

Referenced by CF\_AppInit(), and CF\_SendHkCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.67 CF\_HkPacket\_Payload Struct Reference

Housekeeping packet.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- **CF\_HkCmdCounters\_t counters**  
*Command counters.*
- **uint8 spare [4]**  
*Alignment spare (CF\_HkCmdCounters\_t is 4 bytes)*
- **CF\_HkChannel\_Data\_t channel\_hk [CF\_NUM\_CHANNELS]**  
*Per channel housekeeping data.*

### 11.67.1 Detailed Description

Housekeeping packet.

Definition at line 116 of file default\_cf\_msgdefs.h.

### 11.67.2 Field Documentation

#### 11.67.2.1 channel\_hk [CF\\_HkChannel\\_Data\\_t](#) CF\_HkPacket\_Payload::channel\_hk [CF\_NUM\_CHANNELS]

Per channel housekeeping data.

Definition at line 121 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_CycleEngine(), CF\_CFDP\_DisableEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvDrop(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvHold(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvPh(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_Send(), CF\_CFDP\_StartRxTransaction(), CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), CF\_DoFreezeThaw(), CF\_MoveTransaction(), and CF\_ResetCountersCmd().

#### 11.67.2.2 counters [CF\\_HkCmdCounters\\_t](#) CF\_HkPacket\_Payload::counters

Command counters.

Definition at line 118 of file default\_cf\_msgdefs.h.

Referenced by CF\_AbandonCmd(), CF\_AppPipe(), CF\_CancelCmd(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DisableEngineCmd(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(), CF\_FreezeCmd(), CF\_SetParamCmd(), CF\_NoopCmd(), CF\_PlaybackDirCmd(), CF\_ProcessGroundCommand(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_ThawCmd(), CF\_TxFileCmd(), and CF\_WriteQueueCmd().

#### 11.67.2.3 spare [uint8](#) CF\_HkPacket\_Payload::spare [4]

Alignment spare (CF\_HkCmdCounters\_t is 4 bytes)

Definition at line 119 of file default\_cf\_msgdefs.h.

The documentation for this struct was generated from the following file:

- apps/cf/config/default\_cf\_msgdefs.h

## 11.68 CF\_HkRecv Struct Reference

Housekeeping received counters.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- `uint64 file_data_bytes`  
*Received File data bytes.*
- `uint32 pdu`  
*Received PDUs with valid header counter.*
- `uint32 error`  
*Received PDUs with error counter, see related event for cause.*
- `uint16 spurious`  
*Received PDUs with invalid directive code for current context or file directive FIN without matching active transaction counter, see related event for cause.*
- `uint16 dropped`  
*Received PDUs dropped due to a transaction error.*
- `uint32 nak_segment_requests`  
*Received NAK segment requests counter.*

### 11.68.1 Detailed Description

Housekeeping received counters.

Definition at line 58 of file default\_cf\_msgdefs.h.

### 11.68.2 Field Documentation

#### 11.68.2.1 `dropped uint16 CF_HkRecv::dropped`

Received PDUs dropped due to a transaction error.

Definition at line 67 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_DispatchRecv(), and CF\_CFDP\_RecvDrop().

#### 11.68.2.2 `error uint32 CF_HkRecv::error`

Received PDUs with error counter, see related event for cause.

Definition at line 62 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvPh(), CF\_CFDP\_S2\_SubstateEofAck(), and CF\_CFDP\_S2\_SubstateNak().

#### 11.68.2.3 `file_data_bytes uint64 CF_HkRecv::file_data_bytes`

Received File data bytes.

Definition at line 60 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_ProcessFd().

**11.68.2.4 nak\_segment\_requests** `uint32 CF_HkRecv::nak_segment_requests`

Received NAK segment requests counter.

Definition at line 68 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_S2\_SubstateNak().

**11.68.2.5 pdu** `uint32 CF_HkRecv::pdu`

Received PDUs with valid header counter.

Definition at line 61 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_RecvPh().

**11.68.2.6 spurious** `uint16 CF_HkRecv::spurious`

Received PDUs with invalid directive code for current context or file directive FIN without matching active transaction counter, see related event for cause.

Definition at line 63 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_RecvHold(), and CF\_CFDP\_S\_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.69 CF\_HkSent Struct Reference

Housekeeping sent counters.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- `uint64 file_data_bytes`  
*Sent File data bytes.*
- `uint32 pdu`  
*Sent PDUs counter.*
- `uint32 nak_segment_requests`  
*Sent NAK segment requests counter.*

### 11.69.1 Detailed Description

Housekeeping sent counters.

Definition at line 48 of file default\_cf\_msgdefs.h.

### 11.69.2 Field Documentation

**11.69.2.1 file\_data\_bytes** `uint64 CF_HkSent::file_data_bytes`

Sent File data bytes.

Definition at line 50 of file default\_cf\_msgdefs.h.

Referenced by CF\_CFDP\_S\_SendFileData().

**11.69.2.2 nak\_segment\_requests** `uint32 CF_HkSent::nak_segment_requests`  
Sent NAK segment requests counter.  
Definition at line 52 of file default\_cf\_msgdefs.h.  
Referenced by CF\_CFDP\_R\_SendNak().

**11.69.2.3 pdu** `uint32 CF_HkSent::pdu`  
Sent PDUs counter.  
Definition at line 51 of file default\_cf\_msgdefs.h.  
Referenced by CF\_CFDP\_Send().  
The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

## 11.70 CF\_Input Struct Reference

CF engine input state.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- `CFE_SB_Buffer_t * msg`  
*Binary message received from underlying transport.*
- `CF_DecoderState_t decode`  
*Decoding state (while interpreting message)*
- `CF_Logical_PduBuffer_t rx_pdudata`  
*Rx PDU logical values.*

### 11.70.1 Detailed Description

CF engine input state.

Keeps the state of the current input PDU in the CF engine

Definition at line 427 of file cf\_cfdp\_types.h.

### 11.70.2 Field Documentation

**11.70.2.1 decode** `CF_DecoderState_t CF_Input::decode`  
Decoding state (while interpreting message)  
Definition at line 430 of file cf\_cfdp\_types.h.  
Referenced by CF\_CFDP\_ReceiveMessage().

**11.70.2.2 msg** `CFE_SB_Buffer_t* CF_Input::msg`  
Binary message received from underlying transport.  
Definition at line 429 of file cf\_cfdp\_types.h.

**11.70.2.3 rx\_pdudata** `CF_Logical_PduBuffer_t` `CF_Input::rx_pdudata`  
 Rx PDU logical values.

Definition at line 431 of file `cf_cfdp_types.h`.

Referenced by `CF_CFDP_ReceiveMessage()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_types.h`

## 11.71 CF\_Logical\_IntHeader Union Reference

A union of all possible internal header types in a PDU.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `CF_Logical_PduEof_t eof`  
*valid when pdu\_type=0 + directive\_code=EOF (4)*
- `CF_Logical_PduFin_t fin`  
*valid when pdu\_type=0 + directive\_code=FIN (5)*
- `CF_Logical_PduAck_t ack`  
*valid when pdu\_type=0 + directive\_code=ACK (6)*
- `CF_Logical_PduMd_t md`  
*valid when pdu\_type=0 + directive\_code=METADATA (7)*
- `CF_Logical_PduNak_t nak`  
*valid when pdu\_type=0 + directive\_code=NAK (8)*
- `CF_Logical_PduFileDataHeader_t fd`  
*valid when pdu\_type=1 (directive\_code is not applicable)*

### 11.71.1 Detailed Description

A union of all possible internal header types in a PDU.

The specific entry which applies depends on the combination of pdu type and directive code.

Definition at line 313 of file `cf_logical_pdu.h`.

### 11.71.2 Field Documentation

#### 11.71.2.1 ack

`CF_Logical_PduAck_t` `CF_Logical_IntHeader::ack`

*valid when pdu\_type=0 + directive\_code=ACK (6)*

Definition at line 317 of file `cf_logical_pdu.h`.

Referenced by `CF_CFDP_R2_SubstateRecvFinAck()`, `CF_CFDP_RecvAck()`, `CF_CFDP_S2_SubstateEofAck()`, and `CF_CFDP_SendAck()`.

#### 11.71.2.2 eof

`CF_Logical_PduEof_t` `CF_Logical_IntHeader::eof`

*valid when pdu\_type=0 + directive\_code=EOF (4)*

Definition at line 315 of file `cf_logical_pdu.h`.

Referenced by `CF_CFDP_R_SubstateRecvEof()`, `CF_CFDP_RecvEof()`, and `CF_CFDP_SendEof()`.

**11.71.2.3 fd** `CF_Logical_PduFileDataHeader_t` `CF_Logical_IntHeader::fd`  
valid when `pdu_type=1` (`directive_code` is not applicable)

Definition at line 320 of file `cf_logical_pdu.h`.

Referenced by `CF_CFDP_R_ProcessFd()`, `CF_CFDP_RecvFd()`, and `CF_CFDP_S_SendFileData()`.

**11.71.2.4 fin** `CF_Logical_PduFin_t` `CF_Logical_IntHeader::fin`  
valid when `pdu_type=0 + directive_code=FIN (5)`

Definition at line 316 of file `cf_logical_pdu.h`.

Referenced by `CF_CFDP_RecvFin()`, `CF_CFDP_S_SubstateRecvFin()`, and `CF_CFDP_SendFin()`.

**11.71.2.5 md** `CF_Logical_PduMd_t` `CF_Logical_IntHeader::md`  
valid when `pdu_type=0 + directive_code=METADATA (7)`

Definition at line 318 of file `cf_logical_pdu.h`.

Referenced by `CF_CFDP_RecvMd()`, and `CF_CFDP_SendMd()`.

**11.71.2.6 nak** `CF_Logical_PduNak_t` `CF_Logical_IntHeader::nak`

valid when `pdu_type=0 + directive_code=NAK (8)`

Definition at line 319 of file `cf_logical_pdu.h`.

Referenced by `CF_CFDP_R_SendNak()`, `CF_CFDP_RecvNak()`, `CF_CFDP_S2_SubstateNak()`, and `CF_CFDP_SendNak()`.

The documentation for this union was generated from the following file:

- [apps/cf/fsw/src/cf\\_logical\\_pdu.h](#)

## 11.72 CF\_Logical\_Lv Struct Reference

Structure representing logical LV Object format.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `uint8 length`  
*Length of data field.*
- `const void * data_ptr`  
*Source of actual data in original location.*

### 11.72.1 Detailed Description

Structure representing logical LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

These are only used for string data (mostly filenames) so the data can refer directly to the encoded bits, it does not necessarily need to be duplicated here.

Definition at line 145 of file `cf_logical_pdu.h`.

### 11.72.2 Field Documentation

**11.72.2.1 data\_ptr const void\* CF\_Logical\_Lv::data\_ptr**

Source of actual data in original location.

Definition at line 148 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_CopyStringFromLV(), CF\_CFDP\_DecodeLV(), CF\_CFDP\_EncodeLV(), and CF\_CFDP\_SendMd().

**11.72.2.2 length uint8 CF\_Logical\_Lv::length**

Length of data field.

Definition at line 147 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_CopyStringFromLV(), CF\_CFDP\_DecodeLV(), CF\_CFDP\_EncodeLV(), CF\_CFDP\_RecvMd(), and CF\_CFDP\_SendMd().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_logical\\_pdu.h](#)

## 11.73 CF\_Logical\_PduAck Struct Reference

Structure representing CFDP Acknowledge PDU.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- [uint8 ack\\_directive\\_code](#)  
*directive code of the PDU being ACK'ed*
- [uint8 ack\\_subtype\\_code](#)  
*depends on ack\_directive\_code*
- [CF\\_CFDP\\_ConditionCode\\_t cc](#)
- [CF\\_CFDP\\_AckTxnStatus\\_t txn\\_status](#)

### 11.73.1 Detailed Description

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

Definition at line 253 of file cf\_logical\_pdu.h.

### 11.73.2 Field Documentation

**11.73.2.1 ack\_directive\_code uint8 CF\_Logical\_PduAck::ack\_directive\_code**

directive code of the PDU being ACK'ed

Definition at line 255 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeAck(), CF\_CFDP\_EncodeAck(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_S2\_SubstateEofAck(), and CF\_CFDP\_SendAck().

**11.73.2.2 ack\_subtype\_code uint8 CF\_Logical\_PduAck::ack\_subtype\_code**

depends on ack\_directive\_code

Definition at line 256 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeAck(), CF\_CFDP\_EncodeAck(), and CF\_CFDP\_SendAck().

**11.73.2.3 cc CF\_CFDP\_ConditionCode\_t** CF\_Logical\_PduAck::cc

Definition at line 257 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeAck(), CF\_CFDP\_EncodeAck(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP←\_S2\_SubstateEofAck(), and CF\_CFDP\_SendAck().

**11.73.2.4 txn\_status CF\_CFDP\_AckTxnStatus\_t** CF\_Logical\_PduAck::txn\_status

Definition at line 258 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeAck(), CF\_CFDP\_EncodeAck(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP←\_S2\_SubstateEofAck(), and CF\_CFDP\_SendAck().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_logical\\_pdu.h](#)

## 11.74 CF\_Logical\_PduBuffer Struct Reference

Encapsulates the entire PDU information.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- struct [CF\\_EncoderState](#) \* penc
- struct [CF\\_DecoderState](#) \* pdec
- [CF\\_Logical\\_PduHeader\\_t](#) pdu\_header
  - Data in PDU header is applicable to all packets.*
- [CF\\_Logical\\_PduFileDirectiveHeader\\_t](#) fdirective
  - The directive code applies to file directive PDUs, where the pdu\_type in the common header is 0. Otherwise this value should be set to 0 for data PDUs (which is a reserved value and does not alias any valid directive code).*
- [CF\\_Logical\\_IntHeader\\_t](#) int\_header
  - The internal header is specific to the type of PDU being processed. This is a union of all those possible types. See the union definition for which member applies to a given processing cycle.*
- uint32 content\_crc
  - Some PDU types might have a CRC at the end. If so, this field reflects the value of that CRC. Its presence/validity depends on the pdu\_type and crc\_flag in the pdu\_header.*

### 11.74.1 Detailed Description

Encapsulates the entire PDU information.

Definition at line 327 of file cf\_logical\_pdu.h.

### 11.74.2 Field Documentation

**11.74.2.1 content\_crc uint32** CF\_Logical\_PduBuffer::content\_crc

Some PDU types might have a CRC at the end. If so, this field reflects the value of that CRC. Its presence/validity depends on the pdu\_type and crc\_flag in the pdu\_header.

Note that all CFDP CRCs are 32 bits in length, the blue book does not permit for any other size.

Definition at line 366 of file cf\_logical\_pdu.h.

**11.74.2.2 fdirective** `CF_Logical_PduFileDirectiveHeader_t CF_Logical_PduBuffer::fdirective`

The directive code applies to file directive PDUs, where the pdu\_type in the common header is 0. Otherwise this value should be set to 0 for data PDUs (which is a reserved value and does not alias any valid directive code).

Definition at line 348 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_RecvHold(), CF\_CFDP\_RecvPh(), and CF\_CFDP\_S\_DispatchRecv().

**11.74.2.3 int\_header** `CF_Logical_IntHeader_t CF_Logical_PduBuffer::int_header`

The internal header is specific to the type of PDU being processed. This is a union of all those possible types. See the union definition for which member applies to a given processing cycle.

Definition at line 356 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_RecvAck(), CF\_CFDP\_RecvEof(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvFin(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvNak(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), and CF\_CFDP\_SendNak().

**11.74.2.4 pdec** `struct CF_DecoderState* CF_Logical_PduBuffer::pdec`

Definition at line 335 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeStart(), CF\_CFDP\_RecvAck(), CF\_CFDP\_RecvEof(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvFin(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvNak(), and CF\_CFDP\_RecvPh().

**11.74.2.5 pdu\_header** `CF_Logical_PduHeader_t CF_Logical_PduBuffer::pdu_header`

Data in PDU header is applicable to all packets.

Definition at line 340 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_ReceivePdu(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvPh(), CF\_CFDP\_RxStateDispatch(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), and CF\_CFDP\_SetupRxTransaction().

**11.74.2.6 penc** `struct CF_EncoderState* CF_Logical_PduBuffer::penc`

Definition at line 334 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeStart(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_SendNak(), and CF\_CFDP\_SetPduLength().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src\(cf\\_logical\\_pdu.h\)](#)

## 11.75 CF\_Logical\_PduEof Struct Reference

Structure representing logical End of file PDU.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- [CF\\_CFDP\\_ConditionCode\\_t cc](#)
- [uint32 crc](#)
- [CF\\_FileSize\\_t size](#)
- [CF\\_Logical\\_TlvList\\_t tlv\\_list](#)

*Set of all TLV blobs in this PDU.*

### 11.75.1 Detailed Description

Structure representing logical End of file PDU.

See also

[CF\\_CFDP\\_PduEof\\_t](#) for encoded form

Definition at line 219 of file cf\_logical\_pdu.h.

### 11.75.2 Field Documentation

#### 11.75.2.1 cc [CF\\_CFDP\\_ConditionCode\\_t](#) CF\_Logical\_PduEof::cc

Definition at line 221 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeEof(), CF\_CFDP\_EncodeEof(), CF\_CFDP\_R\_SubstateRecvEof(), and CF\_CFDP\_SendEof().

#### 11.75.2.2 crc [uint32](#) CF\_Logical\_PduEof::crc

Definition at line 222 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeEof(), CF\_CFDP\_EncodeEof(), CF\_CFDP\_R\_SubstateRecvEof(), and CF\_CFDP\_SendEof().

#### 11.75.2.3 size [CF\\_FileSize\\_t](#) CF\_Logical\_PduEof::size

Definition at line 223 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeEof(), CF\_CFDP\_EncodeEof(), CF\_CFDP\_R\_SubstateRecvEof(), and CF\_CFDP\_SendEof().

#### 11.75.2.4 tlv\_list [CF\\_Logical\\_TlvList\\_t](#) CF\_Logical\_PduEof::tlv\_list

Set of all TLV blobs in this PDU.

Definition at line 228 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeEof(), CF\_CFDP\_EncodeEof(), and CF\_CFDP\_SendEof().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf\\_logical\\_pdu.h](#)

## 11.76 CF\_Logical\_PduFileDataHeader Struct Reference

```
#include <cf_logical_pdu.h>
```

### Data Fields

- [uint8 continuation\\_state](#)
- [CF\\_Logical\\_SegmentList\\_t segment\\_list](#)  
*the segment\_meta\_length value will be stored in the segment\_list.num\_segments field below*
- [CF\\_FileSize\\_t offset](#)  
*Offset of data in file.*
- [const void \\* data\\_ptr](#)  
*pointer to read-only data blob within encoded PDU*
- [size\\_t data\\_len](#)  
*Length of data blob within encoded PDU (derived field)*

### 11.76.1 Detailed Description

Definition at line 291 of file cf\_logical\_pdu.h.

### 11.76.2 Field Documentation

#### 11.76.2.1 continuation\_state `uint8 CF_Logical_PduFileDataHeader::continuation_state`

Definition at line 293 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), and CF\_CFDP\_EncodeFileDataHeader().

#### 11.76.2.2 data\_len `size_t CF_Logical_PduFileDataHeader::data_len`

Length of data blob within encoded PDU (derived field)

Definition at line 304 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_RecvFd(), and CF\_CFDP\_S\_SendFileData().

#### 11.76.2.3 data\_ptr `const void* CF_Logical_PduFileDataHeader::data_ptr`

pointer to read-only data blob within encoded PDU

Definition at line 303 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), CF\_CFDP\_R\_ProcessFd(), and CF\_CFDP\_S\_SendFileData().

#### 11.76.2.4 offset `CF_FileSize_t CF_Logical_PduFileDataHeader::offset`

Offset of data in file.

Definition at line 301 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), CF\_CFDP\_EncodeFileDataHeader(), CF\_CFDP\_R\_ProcessFd(), and CF\_CFDP\_S\_SendFileData().

#### 11.76.2.5 segment\_list `CF_Logical_SegmentList_t CF_Logical_PduFileDataHeader::segment_list`

the segment\_meta\_length value will be stored in the segment\_list.num\_segments field below

Definition at line 299 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), and CF\_CFDP\_EncodeFileDataHeader().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf\\_logical\\_pdu.h](#)

## 11.77 CF\_Logical\_PduFileDirectiveHeader Struct Reference

Structure representing logical File Directive header.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- [CF\\_CFDP\\_FileDirective\\_t directive\\_code](#)

### 11.77.1 Detailed Description

Structure representing logical File Directive header.

This contains the file directive code from the PDUs for which it applies. The codes are mapped directly to the CFDP protocol values, but converted to a native value (enum) for direct use by software.

Definition at line 130 of file cf\_logical\_pdu.h.

### 11.77.2 Field Documentation

#### 11.77.2.1 directive\_code [CF\\_CFDP\\_FileDirective\\_t](#) CF\_Logical\_PduFileDirectiveHeader::directive\_code

Definition at line 132 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_DecodeFileDirectiveHeader(), CF\_CFDP\_EncodeFileDirectiveHeader(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_RecvHold(), and CF\_CFDP\_S\_DispatchRecv().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src\(cf\\_logical\\_pdu.h\)](#)

## 11.78 CF\_Logical\_PduFin Struct Reference

Structure representing logical Finished PDU.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- [CF\\_CFDP\\_ConditionCode\\_t cc](#)  
*complete file indicated by '0'. Nonzero means incomplete.*
- [CF\\_CFDP\\_FinFileStatus\\_t file\\_status](#)
- [uint8 delivery\\_code](#)  
*Set of all TLV blobs in this PDU.*

### 11.78.1 Detailed Description

Structure representing logical Finished PDU.

### See also

[CF\\_CFDP\\_PduFin\\_t](#) for encoded form

Definition at line 236 of file cf\_logical\_pdu.h.

### 11.78.2 Field Documentation

#### 11.78.2.1 cc [CF\\_CFDP\\_ConditionCode\\_t](#) CF\_Logical\_PduFin::cc

Definition at line 238 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFin(), CF\_CFDP\_EncodeFin(), CF\_CFDP\_S\_SubstateRecvFin(), and CF\_CFDP\_SendFin().

**11.78.2.2 delivery\_code** `uint8 CF_Logical_PduFin::delivery_code`  
 complete file indicated by '0'. Nonzero means incomplete.

Definition at line 240 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFin(), CF\_CFDP\_EncodeFin(), CF\_CFDP\_S\_SubstateRecvFin(), and CF\_CFDP\_SendFin().

**11.78.2.3 file\_status** `CF_CFDP_FinFileStatus_t CF_Logical_PduFin::file_status`

Definition at line 239 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFin(), CF\_CFDP\_EncodeFin(), CF\_CFDP\_S\_SubstateRecvFin(), and CF\_CFDP\_SendFin().

**11.78.2.4 tlv\_list** `CF_Logical_TlvList_t CF_Logical_PduFin::tlv_list`

Set of all TLV blobs in this PDU.

Definition at line 245 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeFin(), CF\_CFDP\_EncodeFin(), and CF\_CFDP\_SendFin().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src\(cf\\_logical\\_pdu.h\)](#)

## 11.79 CF\_Logical\_PduHeader Struct Reference

Structure representing base CFDP PDU header.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- **uint8 version**  
*Version of the protocol.*
- **uint8 pdu\_type**  
*File Directive (0) or File Data (1)*
- **uint8 direction**  
*Toward Receiver (0) or Toward Sender (1)*
- **uint8 txm\_mode**  
*Acknowledged (0) or Unacknowledged (1)*
- **uint8 crc\_flag**  
*CRC not present (0) or CRC present (1)*
- **uint8 large\_flag**  
*Small/32-bit size (0) or Large/64-bit size (1)*
- **uint8 segmentation\_control**  
*Record boundaries not preserved (0) or preserved (1)*
- **uint8 eid\_length**  
*Length of encoded entity IDs, in octets (NOT size of logical value)*
- **uint8 segment\_meta\_flag**  
*Segment Metadata not present (0) or Present (1)*
- **uint8 txn\_seq\_length**  
*Length of encoded sequence number, in octets (NOT size of logical value)*
- **uint16 header\_encoded\_length**  
*Length of the encoded PDU header, in octets (NOT sizeof struct)*
- **uint16 data\_encoded\_length**

*Length of the encoded PDU data, in octets.*

- **CF\_EntityId\_t source\_eid**  
*Source entity ID (normalized)*
- **CF\_EntityId\_t destination\_eid**  
*Destination entity ID (normalized)*
- **CF\_TransactionSeq\_t sequence\_num**  
*Sequence number (normalized)*

### 11.79.1 Detailed Description

Structure representing base CFDP PDU header.

Reflects the common content at the beginning of all CFDP PDUs, of all types.

See also

[CF\\_CFDP\\_PduHeader\\_t](#) for encoded form

Definition at line 101 of file cf\_logical\_pdu.h.

### 11.79.2 Field Documentation

#### 11.79.2.1 **crc\_flag** `uint8` `CF_Logical_PduHeader::crc_flag`

CRC not present (0) or CRC present (1)

Definition at line 107 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_RecvFd()`.

#### 11.79.2.2 **data\_encoded\_length** `uint16` `CF_Logical_PduHeader::data_encoded_length`

Length of the encoded PDU data, in octets.

Definition at line 116 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderFinalSize()`, `CF_CFDP_Send()`, and `CF_CFDP_SetPduLength()`.

#### 11.79.2.3 **destination\_eid** `CF_EntityId_t` `CF_Logical_PduHeader::destination_eid`

Destination entity ID (normalized)

Definition at line 119 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, and `CF_CFDP_ReceivePdu()`.

#### 11.79.2.4 **direction** `uint8` `CF_Logical_PduHeader::direction`

Toward Receiver (0) or Toward Sender (1)

Definition at line 105 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderWithoutSize()`.

**11.79.2.5 eid\_length** `uint8` `CF_Logical_PduHeader::eid_length`

Length of encoded entity IDs, in octets (NOT size of logical value)

Definition at line 111 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderWithoutSize()`.

**11.79.2.6 header\_encoded\_length** `uint16` `CF_Logical_PduHeader::header_encoded_length`

Length of the encoded PDU header, in octets (NOT sizeof struct)

Definition at line 115 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_Send()`, and `CF_CFDP_SetPduLength()`.

**11.79.2.7 large\_flag** `uint8` `CF_Logical_PduHeader::large_flag`

Small/32-bit size (0) or Large/64-bit size (1)

Definition at line 108 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_RecvPh()`.

**11.79.2.8 pdu\_type** `uint8` `CF_Logical_PduHeader::pdu_type`

File Directive (0) or File Data (1)

Definition at line 104 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_R_DispatchRecv()`, `CF_CFDP_RecvPh()`, and `CF_CFDP_S_DispatchRecv()`.

**11.79.2.9 segment\_meta\_flag** `uint8` `CF_Logical_PduHeader::segment_meta_flag`

Segment Metatdata not present (0) or Present (1)

Definition at line 112 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_RecvFd()`, and `CF_CFDP_S_SendFileData()`.

**11.79.2.10 segmentation\_control** `uint8` `CF_Logical_PduHeader::segmentation_control`

Record boundaries not preserved (0) or preserved (1)

Definition at line 110 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeHeader()`, and `CF_CFDP_EncodeHeaderWithoutSize()`.

**11.79.2.11 sequence\_num** `CF_TransactionSeq_t` `CF_Logical_PduHeader::sequence_num`

Sequence number (normalized)

Definition at line 120 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_ReceivePdu()`, and `CF_CFDP_SetupRxTransaction()`.

**11.79.2.12 source\_eid** `CF_EntityId_t` `CF_Logical_PduHeader::source_eid`

Source entity ID (normalized)

Definition at line 118 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_ConstructPduHeader()`, `CF_CFDP_DecodeHeader()`, `CF_CFDP_EncodeHeaderWithoutSize()`, `CF_CFDP_ReceivePdu()`, and `CF_CFDP_SetupRxTransaction()`.

**11.79.2.13 txm\_mode** `uint8` CF\_Logical\_PduHeader::txm\_mode

Acknowledged (0) or Unacknowledged (1)

Definition at line 106 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_DecodeHeader(), CF\_CFDP\_EncodeHeaderWithoutSize(), CF\_CFDP\_RxStateDispatch(), and CF\_CFDP\_SetupRxTransaction().

**11.79.2.14 txn\_seq\_length** `uint8` CF\_Logical\_PduHeader::txn\_seq\_length

Length of encoded sequence number, in octets (NOT size of logical value)

Definition at line 113 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**11.79.2.15 version** `uint8` CF\_Logical\_PduHeader::version

Version of the protocol.

Definition at line 103 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_logical_pdu.h`

## 11.80 CF\_Logical\_PduMd Struct Reference

Structure representing CFDP Metadata PDU.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `uint8 close_req`  
*transaction closure not requested (0) or requested (1)*
- `uint8 checksum_type`  
*0 indicates legacy modular checksum*
- `CF_FileSize_t size`
- `CF_Logical_Lv_t source_filename`
- `CF_Logical_Lv_t dest_filename`

### 11.80.1 Detailed Description

Structure representing CFDP Metadata PDU.

Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5

Definition at line 266 of file cf\_logical\_pdu.h.

### 11.80.2 Field Documentation

**11.80.2.1 checksum\_type** `uint8` CF\_Logical\_PduMd::checksum\_type

0 indicates legacy modular checksum

Definition at line 269 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeMd(), CF\_CFDP\_EncodeMd(), and CF\_CFDP\_SendMd().

**11.80.2.2 close\_req** `uint8 CF_Logical_PduMd::close_req`

transaction closure not requested (0) or requested (1)

Definition at line 268 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeMd(), CF\_CFDP\_EncodeMd(), CF\_CFDP\_RecvMd(), and CF\_CFDP\_SendMd().

**11.80.2.3 dest\_filename** `CF_Logical_Lv_t CF_Logical_PduMd::dest_filename`

Definition at line 274 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeMd(), CF\_CFDP\_EncodeMd(), CF\_CFDP\_RecvMd(), and CF\_CFDP\_SendMd().

**11.80.2.4 size** `CF_FileSize_t CF_Logical_PduMd::size`

Definition at line 271 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeMd(), CF\_CFDP\_EncodeMd(), CF\_CFDP\_RecvMd(), and CF\_CFDP\_SendMd().

**11.80.2.5 source\_filename** `CF_Logical_Lv_t CF_Logical_PduMd::source_filename`

Definition at line 273 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeMd(), CF\_CFDP\_EncodeMd(), CF\_CFDP\_RecvMd(), and CF\_CFDP\_SendMd().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src(cf_logical_pdu.h)`

## 11.81 CF\_Logical\_PduNak Struct Reference

Structure representing logical Non-Acknowledge PDU.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `CF_FileSize_t scope_start`
- `CF_FileSize_t scope_end`
- `CF_Logical_SegmentList_t segment_list`

*Set of all segments in this PDU.*

### 11.81.1 Detailed Description

Structure representing logical Non-Acknowledge PDU.

Definition at line 280 of file cf\_logical\_pdu.h.

### 11.81.2 Field Documentation

**11.81.2.1 scope\_end** `CF_FileSize_t CF_Logical_PduNak::scope_end`

Definition at line 283 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeNak(), CF\_CFDP\_EncodeNak(), and CF\_CFDP\_R\_SendNak().

**11.81.2.2 scope\_start** `CF_FileSize_t CF_Logical_PduNak::scope_start`

Definition at line 282 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeNak(), CF\_CFDP\_EncodeNak(), CF\_CFDP\_R2\_GapCompute(), and CF\_CFDP\_R\_SendNak().

**11.81.2.3 segment\_list** `CF_Logical_SegmentList_t` `CF_Logical_PduNak::segment_list`

Set of all segments in this PDU.

Definition at line 288 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeNak()`, `CF_CFDP_EncodeNak()`, `CF_CFDP_R2_GapCompute()`, `CF_CFDP_R_SendNak()`, `CF_CFDP_S2_SubstateNak()`, and `CF_CFDP_SendNak()`.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_logical\\_pdu.h](#)

## 11.82 CF\_Logical\_SegmentList Struct Reference

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `uint8 num_segments`  
*number of valid entries in the segment list*
- `CF_Logical_SegmentRequest_t segments [CF_PDU_MAX_SEGMENTS]`  
*Set of all segment requests in this PDU.*

### 11.82.1 Detailed Description

Definition at line 194 of file cf\_logical\_pdu.h.

### 11.82.2 Field Documentation

#### 11.82.2.1 num\_segments

`uint8 CF_Logical_SegmentList::num_segments`

number of valid entries in the segment list

Definition at line 196 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeAllSegments()`, `CF_CFDP_DecodeFileDataHeader()`, `CF_CFDP_EncodeAllSegments()`, `CF_CFDP_EncodeFileDataHeader()`, `CF_CFDP_R2_GapCompute()`, `CF_CFDP_R_SendNak()`, `CF_CFDP_S2_SubstateNak()`, and `CF_CFDP_SendNak()`.

#### 11.82.2.2 segments

`CF_Logical_SegmentRequest_t CF_Logical_SegmentList::segments [CF_PDU_MAX_SEGMENTS]`

Set of all segment requests in this PDU.

Number of valid entries is indicated by `num_segments`, and may be 0 if the PDU does not contain any such fields.

Definition at line 204 of file cf\_logical\_pdu.h.

Referenced by `CF_CFDP_DecodeAllSegments()`, `CF_CFDP_DecodeFileDataHeader()`, `CF_CFDP_EncodeAllSegments()`, `CF_CFDP_R2_GapCompute()`, `CF_CFDP_R_SendNak()`, and `CF_CFDP_S2_SubstateNak()`.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_logical\\_pdu.h](#)

## 11.83 CF\_Logical\_SegmentRequest Struct Reference

Structure representing logical Segment Request data.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `CF_FileSize_t offset_start`
- `CF_FileSize_t offset_end`

### 11.83.1 Detailed Description

Structure representing logical Segment Request data.  
Definition at line 188 of file cf\_logical\_pdu.h.

### 11.83.2 Field Documentation

#### 11.83.2.1 offset\_end [CF\\_FileSize\\_t](#) CF\_Logical\_SegmentRequest::offset\_end

Definition at line 191 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeSegmentRequest(), CF\_CFDP\_EncodeSegmentRequest(), CF\_CFDP\_R2\_GapCompute(), CF\_CFDP\_R\_SendNak(), and CF\_CFDP\_S2\_SubstateNak().

#### 11.83.2.2 offset\_start [CF\\_FileSize\\_t](#) CF\_Logical\_SegmentRequest::offset\_start

Definition at line 190 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_DecodeSegmentRequest(), CF\_CFDP\_EncodeSegmentRequest(), CF\_CFDP\_R2\_GapCompute(), CF\_CFDP\_R\_SendNak(), and CF\_CFDP\_S2\_SubstateNak().

The documentation for this struct was generated from the following file:

- [apps/cf/fsd/src/cf\\_logical\\_pdu.h](#)

## 11.84 CF\_Logical\_Tlv Struct Reference

Structure representing logical TLV Object format.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- [CF\\_CFDP\\_TlvType\\_t type](#)  
*Nature of data field.*
- [uint8 length](#)  
*Length of data field (encoded length, not local storage size)*
- [CF\\_Logical\\_TlvData\\_t data](#)

### 11.84.1 Detailed Description

Structure representing logical TLV Object format.

In the current implementation of CF, only entity IDs are currently encoded in this form where indicated in the spec. This may change in a future version.

#### See also

[CF\\_CFDP\\_tlv\\_t](#) for encoded form

Definition at line 178 of file cf\_logical\_pdu.h.

### 11.84.2 Field Documentation

#### 11.84.2.1 data [CF\\_Logical\\_TlvData\\_t](#) CF\_Logical\_Tlv::data

Definition at line 182 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_DecodeTLV(), and CF\_CFDP\_EncodeTLV().

**11.84.2.2 length** `uint8 CF_Logical_Tlv::length`

Length of data field (encoded length, not local storage size)

Definition at line 181 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_DecodeTLV(), and CF\_CFDP\_EncodeTLV().

**11.84.2.3 type** `CF_CFDP_TlvType_t CF_Logical_Tlv::type`

Nature of data field.

Definition at line 180 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_DecodeTLV(), and CF\_CFDP\_EncodeTLV().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_logical\\_pdu.h](#)

## 11.85 CF\_Logical\_TlvData Union Reference

Union of various data items that may occur in a TLV item.

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `CF_EntityId_t eid`  
*Valid when type=ENTITY\_ID (6)*
- `const void * data_ptr`  
*Source of actual data in original location (other string/binary types)*

### 11.85.1 Detailed Description

Union of various data items that may occur in a TLV item.

The actual type is identified by the "type" field in the enclosing TLV

Currently filestore requests are not implemented in CF, so the TLV use is limited. This may change in the future.

Numeric data needs to actually be copied to this buffer, because it needs to be normalized in length and byte-order. But string data (e.g. filenames, messages) can reside in the original encoded form.

Definition at line 163 of file cf\_logical\_pdu.h.

### 11.85.2 Field Documentation

**11.85.2.1 data\_ptr** `const void* CF_Logical_TlvData::data_ptr`

Source of actual data in original location (other string/binary types)

Definition at line 166 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_DecodeTLV(), and CF\_CFDP\_EncodeTLV().

**11.85.2.2 eid** `CF_EntityId_t CF_Logical_TlvData::eid`

Valid when type=ENTITY\_ID (6)

Definition at line 165 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_DecodeTLV(), and CF\_CFDP\_EncodeTLV().

The documentation for this union was generated from the following file:

- [apps/cf/fsw/src/cf\\_logical\\_pdu.h](#)

## 11.86 CF\_Logical\_TlvList Struct Reference

```
#include <cf_logical_pdu.h>
```

### Data Fields

- `uint8 num_tlv`  
*number of valid entries in the TLV list*
- `CF_Logical_Tlv_t tlv [CF_PDU_MAX_TLV]`

#### 11.86.1 Detailed Description

Definition at line 207 of file cf\_logical\_pdu.h.

#### 11.86.2 Field Documentation

##### 11.86.2.1 num\_tlv `uint8 CF_Logical_TlvList::num_tlv`

number of valid entries in the TLV list

Definition at line 209 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_DecodeAllTlv(), and CF\_CFDP\_EncodeAllTlv().

##### 11.86.2.2 tlv `CF_Logical_Tlv_t CF_Logical_TlvList::tlv[CF_PDU_MAX_TLV]`

Definition at line 211 of file cf\_logical\_pdu.h.

Referenced by CF\_CFDP\_AppendTlv(), CF\_CFDP\_DecodeAllTlv(), and CF\_CFDP\_EncodeAllTlv().

The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_logical_pdu.h`

## 11.87 CF\_NoopCmd Struct Reference

Noop command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

#### 11.87.1 Detailed Description

Noop command structure.

For command details see `CF_NOOP_CC`

Definition at line 74 of file default\_cf\_msgstruct.h.

#### 11.87.2 Field Documentation

**11.87.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_NoopCmd::CommandHeader  
Command header.

Definition at line 76 of file default\_cf\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.88 CF\_Output Struct Reference

CF engine output state.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- [CFE\\_SB\\_Buffer\\_t \\* msg](#)  
*Binary message to be sent to underlying transport.*
- [CF\\_EncoderState\\_t encode](#)  
*Encoding state (while building message)*
- [CF\\_Logical\\_PduBuffer\\_t tx\\_pdudata](#)  
*Tx PDU logical values.*

### 11.88.1 Detailed Description

CF engine output state.

Keeps the state of the current output PDU in the CF engine

Definition at line 415 of file cf\_cfdp\_types.h.

### 11.88.2 Field Documentation

#### 11.88.2.1 encode [CF\\_EncoderState\\_t](#) CF\_Output::encode

Encoding state (while building message)

Definition at line 418 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_MsgOutGet().

#### 11.88.2.2 msg [CFE\\_SB\\_Buffer\\_t\\*](#) CF\_Output::msg

Binary message to be sent to underlying transport.

Definition at line 417 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_MsgOutGet(), and CF\_CFDP\_Send().

#### 11.88.2.3 tx\_pdudata [CF\\_Logical\\_PduBuffer\\_t](#) CF\_Output::tx\_pdudata

Tx PDU logical values.

Definition at line 419 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_MsgOutGet().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.89 CF\_PduCmdMsg Struct Reference

PDU command encapsulation structure.

```
#include <cf_cfdp_sbintf.h>
```

## Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) `hdr`  
*software bus headers, not really used by CF*
- [CF\\_CFDP\\_PduHeader\\_t](#) `ph`  
*Beginning of CFDP headers.*

### 11.89.1 Detailed Description

PDU command encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "command" encapsulation (even though these are not really commands).

#### Note

this is only the definition of the header. In reality all messages are larger than this, up to CF\_MAX\_PDU\_SIZE.

Definition at line 44 of file cf\_cfdp\_sbintf.h.

### 11.89.2 Field Documentation

#### 11.89.2.1 `hdr` [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_PduCmdMsg::`hdr`

software bus headers, not really used by CF

Definition at line 46 of file cf\_cfdp\_sbintf.h.

#### 11.89.2.2 `ph` [CF\\_CFDP\\_PduHeader\\_t](#) CF\_PduCmdMsg::`ph`

Beginning of CFDP headers.

Definition at line 47 of file cf\_cfdp\_sbintf.h.

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_sbintf.h](#)

## 11.90 CF\_PduTlmMsg Struct Reference

PDU send encapsulation structure.

```
#include <cf_cfdp_sbintf.h>
```

## Data Fields

- [CFE\\_MSG\\_TelemetryHeader\\_t](#) `hdr`  
*software bus headers, not really used by CF*
- [CF\\_CFDP\\_PduHeader\\_t](#) `ph`  
*Beginning of CFDP headers.*

### 11.90.1 Detailed Description

PDU send encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "telemetry" encapsulation (even though these are not really telemetry items).

#### Note

this is only the definition of the header. In reality all messages are larger than this, up to CF\_MAX\_PDU\_SIZE.

Definition at line 60 of file cf\_cfdp\_sbintf.h.

### 11.90.2 Field Documentation

**11.90.2.1 `hdr`** `CFE_MSG_TelemetryHeader_t` `CF_PduTlmMsg::hdr`  
software bus headers, not really used by CF  
Definition at line 62 of file cf\_cfdp\_sbintf.h.

**11.90.2.2 `ph`** `CF_CFDP_PduHeader_t` `CF_PduTlmMsg::ph`  
Beginning of CFDP headers.  
Definition at line 63 of file cf\_cfdp\_sbintf.h.  
The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_cfdp_sbintf.h`

## 11.91 CF\_Playback Struct Reference

CF Playback entry.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- `osal_id_t dir_id`
- `CF_CFDP_Class_t cfdp_class`
- `CF_TxnFilenames_t fnames`
- `uint16 num_ts`  
*number of transactions*
- `uint8 priority`
- `CF_EntityId_t dest_id`
- `char pending_file [OS_MAX_FILE_NAME]`
- `bool busy`
- `bool diropen`
- `bool keep`
- `bool counted`

### 11.91.1 Detailed Description

CF Playback entry.

Keeps the state of CF playback requests

Definition at line 205 of file cf\_cfdp\_types.h.

### 11.91.2 Field Documentation

**11.91.2.1 `busy`** `bool` `CF_Playback::busy`  
Definition at line 215 of file cf\_cfdp\_types.h.

Referenced by `CF_CFDP_DisableEngine()`, `CF_CFDP_PlaybackDir()`, `CF_CFDP_PlaybackDir_Initiate()`, `CF_CFDP_ProcessPlaybackDirectories()`, `CF_CFDP_ProcessPlaybackDirectory()`, and `CF_CFDP_ProcessPollingDirectories()`.

**11.91.2.2 cfdp\_class** `CF_CFDP_Class_t` CF\_Playback::cfdp\_class

Definition at line 208 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), and CF\_CFDP\_ProcessPlaybackDirectory().

**11.91.2.3 counted** `bool` CF\_Playback::counted

Definition at line 218 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_UpdatePollPbCounted().

**11.91.2.4 dest\_id** `CF_EntityId_t` CF\_Playback::dest\_id

Definition at line 212 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), and CF\_CFDP\_ProcessPlaybackDirectory().

**11.91.2.5 dir\_id** `osal_id_t` CF\_Playback::dir\_id

Definition at line 207 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_DisableEngine(), CF\_CFDP\_PlaybackDir\_Initiate(), and CF\_CFDP\_ProcessPlaybackDirectory().

**11.91.2.6 diropen** `bool` CF\_Playback::diropen

Definition at line 216 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), and CF\_CFDP\_ProcessPlaybackDirectory().

**11.91.2.7 fnames** `CF_TxnFilenames_t` CF\_Playback::fnames

Definition at line 209 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), and CF\_CFDP\_ProcessPlaybackDirectory().

**11.91.2.8 keep** `bool` CF\_Playback::keep

Definition at line 217 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), and CF\_CFDP\_ProcessPlaybackDirectory().

**11.91.2.9 num\_ts** `uint16` CF\_Playback::num\_ts

number of transactions

Definition at line 210 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), CF\_CFDP\_ProcessPlaybackDirectory(), and CF\_CFDP\_ProcessPollingDirectories().

**11.91.2.10 pending\_file** `char` CF\_Playback::pending\_file[`OS_MAX_FILE_NAME`]

Definition at line 213 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ProcessPlaybackDirectory().

**11.91.2.11 priority** `uint8` CF\_Playback::priority

Definition at line 211 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), and CF\_CFDP\_ProcessPlaybackDirectory().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.92 CF\_PlaybackDirCmd Struct Reference

Playback directory command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_TxFile\\_Payload\\_t Payload](#)

#### 11.92.1 Detailed Description

Playback directory command structure.

For command details see [CF\\_PLAYBACK\\_DIR\\_CC](#)

Definition at line 236 of file default\_cf\_msgstruct.h.

#### 11.92.2 Field Documentation

##### 11.92.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_PlaybackDirCmd::CommandHeader

Command header.

Definition at line 238 of file default\_cf\_msgstruct.h.

##### 11.92.2.2 Payload [CF\\_TxFile\\_Payload\\_t](#) CF\_PlaybackDirCmd::Payload

Definition at line 239 of file default\_cf\_msgstruct.h.

Referenced by [CF\\_PlaybackDirCmd\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.93 CF\_Poll Struct Reference

CF Poll entry.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- [CF\\_Playback\\_t pb](#)
- [CF\\_Timer\\_t interval\\_timer](#)
- bool [timer\\_set](#)

#### 11.93.1 Detailed Description

CF Poll entry.

Keeps the state of CF directory polling

Definition at line 226 of file cf\_cfdp\_types.h.

#### 11.93.2 Field Documentation

**11.93.2.1 interval\_timer** [CF\\_Timer\\_t](#) CF\_Poll::interval\_timer

Definition at line 229 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

**11.93.2.2 pb** [CF\\_Playback\\_t](#) CF\_Poll::pb

Definition at line 228 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_DisableEngine(), and CF\_CFDP\_ProcessPollingDirectories().

**11.93.2.3 timer\_set** bool CF\_Poll::timer\_set

Definition at line 230 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.94 CF\_PollDir Struct Reference

Configuration entry for directory polling.

#include &lt;default\_cf\_tbldefs.h&gt;

### Data Fields

- [uint32 interval\\_sec](#)  
*number of seconds to wait before trying a new directory*
- [uint8 priority](#)  
*priority to use when placing transactions on the pending queue*
- [CF\\_CFDP\\_Class\\_t cfdp\\_class](#)  
*the CFDP class to send*
- [CF\\_EntityId\\_t dest\\_eid](#)  
*destination entity id*
- [char src\\_dir \[CF\\_FILENAME\\_MAX\\_PATH\]](#)  
*path to source dir*
- [char dst\\_dir \[CF\\_FILENAME\\_MAX\\_PATH\]](#)  
*path to destination dir*
- [uint8 enabled](#)  
*Enabled flag.*

### 11.94.1 Detailed Description

Configuration entry for directory polling.

Definition at line 40 of file default\_cf\_tbldefs.h.

### 11.94.2 Field Documentation

**11.94.2.1 cfdp\_class** [CF\\_CFDP\\_Class\\_t](#) CF\_PollDir::cfdp\_class

the CFDP class to send

Definition at line 45 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

**11.94.2.2 dest\_eid** `CF_EntityId_t` CF\_PollDir::dest\_eid  
destination entity id

Definition at line 46 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

**11.94.2.3 dst\_dir** `char` CF\_PollDir::dst\_dir[`CF_FILENAME_MAX_PATH`]  
path to destination dir

Definition at line 49 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

**11.94.2.4 enabled** `uint8` CF\_PollDir::enabled

Enabled flag.

Definition at line 51 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories(), and CF\_DoEnableDisablePolldir().

**11.94.2.5 interval\_sec** `uint32` CF\_PollDir::interval\_sec

number of seconds to wait before trying a new directory

Definition at line 42 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

**11.94.2.6 priority** `uint8` CF\_PollDir::priority

priority to use when placing transactions on the pending queue

Definition at line 44 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

**11.94.2.7 src\_dir** `char` CF\_PollDir::src\_dir[`CF_FILENAME_MAX_PATH`]

path to source dir

Definition at line 48 of file default\_cf\_tbldefs.h.

Referenced by CF\_CFDP\_ProcessPollingDirectories().

The documentation for this struct was generated from the following file:

- apps/cf/config/default\_cf\_tbldefs.h

## 11.95 CF\_PurgeQueueCmd Struct Reference

PurgeQueue command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t` CommandHeader
  - Command header.*
- `CF_UnionArgs_Payload_t` Payload
  - Generic command arguments.*

### 11.95.1 Detailed Description

PurgeQueue command structure.

For command details see [CF\\_PURGE\\_QUEUE\\_CC](#)

Definition at line 181 of file default\_cf\_msgstruct.h.

### 11.95.2 Field Documentation

#### 11.95.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_PurgeQueueCmd::CommandHeader

Command header.

Definition at line 183 of file default\_cf\_msgstruct.h.

#### 11.95.2.2 Payload [CF\\_UnionArgs\\_Payload\\_t](#) CF\_PurgeQueueCmd::Payload

Generic command arguments.

Definition at line 184 of file default\_cf\_msgstruct.h.

Referenced by CF\_PurgeQueueCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.96 CF\_ResetCountersCmd Struct Reference

Reset command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CF\\_UnionArgs\\_Payload\\_t](#) Payload  
*Generic command arguments.*

### 11.96.1 Detailed Description

Reset command structure.

For command details see [CF\\_RESET\\_CC](#)

Definition at line 104 of file default\_cf\_msgstruct.h.

### 11.96.2 Field Documentation

#### 11.96.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_ResetCountersCmd::CommandHeader

Command header.

Definition at line 106 of file default\_cf\_msgstruct.h.

**11.96.2.2 Payload** [CF\\_UnionArgs\\_Payload\\_t](#) CF\_ResetCountersCmd::Payload  
Generic command arguments.

Definition at line 107 of file default\_cf\_msgstruct.h.  
Referenced by CF\_ResetCountersCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.97 CF\_ResumeCmd Struct Reference

Resume command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_Transaction\\_Payload\\_t Payload](#)

#### 11.97.1 Detailed Description

Resume command structure.

For command details see [CF\\_RESUME\\_CC](#)

Definition at line 258 of file default\_cf\_msgstruct.h.

#### 11.97.2 Field Documentation

**11.97.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_ResumeCmd::CommandHeader  
Command header.

Definition at line 260 of file default\_cf\_msgstruct.h.

**11.97.2.2 Payload** [CF\\_Transaction\\_Payload\\_t](#) CF\_ResumeCmd::Payload

Definition at line 261 of file default\_cf\_msgstruct.h.

Referenced by CF\_ResumeCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.98 CF\_SendHkCmd Struct Reference

Send Housekeeping Command.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*

#### 11.98.1 Detailed Description

Send Housekeeping Command.

Internal notification from SCH with no payload

Definition at line 291 of file default\_cf\_msgstruct.h.

## 11.98.2 Field Documentation

**11.98.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_SendHkCmd::CommandHeader  
Command header.

Definition at line 293 of file default\_cf\_msgstruct.h.

The documentation for this struct was generated from the following file:

- apps/cf/config/default\_cf\_msgstruct.h

## 11.99 CF\_SetParam\_Payload Struct Reference

Set parameter command structure.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- **uint32 value**  
*Parameter value to set.*
- **uint8 key**  
*Parameter key, see [CF\\_GetSet\\_ValueID\\_t](#).*
- **uint8 chan\_num**  
*Channel number.*
- **uint8 spare [2]**  
*Alignment spare, uint32 multiple.*

### 11.99.1 Detailed Description

Set parameter command structure.

For command details see [CF\\_SET\\_PARAM\\_CC](#)

Definition at line 228 of file default\_cf\_msgdefs.h.

## 11.99.2 Field Documentation

**11.99.2.1 chan\_num** [uint8](#) CF\_SetParam\_Payload::chan\_num

Channel number.

Definition at line 232 of file default\_cf\_msgdefs.h.

Referenced by CF\_SetParamCmd().

**11.99.2.2 key** [uint8](#) CF\_SetParam\_Payload::key

Parameter key, see [CF\\_GetSet\\_ValueID\\_t](#).

Definition at line 231 of file default\_cf\_msgdefs.h.

Referenced by CF\_SetParamCmd().

**11.99.2.3 spare** [uint8](#) CF\_SetParam\_Payload::spare[2]

Alignment spare, uint32 multiple.

Definition at line 233 of file default\_cf\_msgdefs.h.

**11.99.2.4 value** `uint32 CF_SetParam_Payload::value`

Parameter value to set.

Definition at line 230 of file default\_cf\_msgdefs.h.

Referenced by CF\_SetParamCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.100 CF\_SetParamCmd Struct Reference

Set parameter command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_SetParam\\_Payload\\_t Payload](#)

### 11.100.1 Detailed Description

Set parameter command structure.

For command details see [CF\\_SET\\_PARAM\\_CC](#)

Definition at line 203 of file default\_cf\_msgstruct.h.

### 11.100.2 Field Documentation

#### 11.100.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t CF\\_SetParamCmd::CommandHeader](#)

Command header.

Definition at line 205 of file default\_cf\_msgstruct.h.

#### 11.100.2.2 Payload [CF\\_SetParam\\_Payload\\_t CF\\_SetParamCmd::Payload](#)

Definition at line 206 of file default\_cf\_msgstruct.h.

Referenced by CF\_SetParamCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.101 CF\_StateData Struct Reference

Summary of all possible transaction state information (tx and rx)

```
#include <cf_cfdp_types.h>
```

### Data Fields

- `uint8 sub_state`
- `uint8 acknak_count`
- `uint8 peer_cc`  
*the peer cc from the received FIN or EOF PDU*
- `uint8 fin_dc`  
*the dc in FIN PDU*

- `uint8 fin_fs`  
*the fs in FIN PDU*
- `CF_FileSize_t cached_pos`
- `uint32 eof_crc`  
*remember the crc in the received EOF PDU*
- `CF_FileSize_t eof_size`  
*remember the size in the received EOF PDU*

### 11.101.1 Detailed Description

Summary of all possible transaction state information (tx and rx)

Definition at line 300 of file cf\_cfdp\_types.h.

### 11.101.2 Field Documentation

#### 11.101.2.1 `acknak_count uint8 CF_StateData::acknak_count`

Number of times the ack\_timer expired and reset

Definition at line 303 of file cf\_cfdp\_types.h.

Referenced by `CF_CFDP_R_CheckState()`, `CF_CFDP_R_CheckState_DATA_EOF()`, `CF_CFDP_R_CheckState_<FINACK()`, `CF_CFDP_R_SubstateRecvFileData()`, `CF_CFDP_S2_CheckState_DATA_EOF()`, and `CF_CFDP_S_CheckState()`.

#### 11.101.2.2 `cached_pos CF_FileSize_t CF_StateData::cached_pos`

Definition at line 309 of file cf\_cfdp\_types.h.

Referenced by `CF_CFDP_R_CalcCrcChunk()`, `CF_CFDP_R_CalcCrcStart()`, `CF_CFDP_R_CheckState_VALIDATE()`, `CF_CFDP_R_ProcessFd()`, and `CF_CFDP_S_SendFileData()`.

#### 11.101.2.3 `eof_crc uint32 CF_StateData::eof_crc`

remember the crc in the received EOF PDU

Definition at line 310 of file cf\_cfdp\_types.h.

Referenced by `CF_CFDP_R_CheckCrc()`, and `CF_CFDP_R_SubstateRecvEof()`.

#### 11.101.2.4 `eof_size CF_FileSize_t CF_StateData::eof_size`

remember the size in the received EOF PDU

Definition at line 311 of file cf\_cfdp\_types.h.

Referenced by `CF_CFDP_R_CalcCrcStart()`, and `CF_CFDP_R_SubstateRecvEof()`.

#### 11.101.2.5 `fin_dc uint8 CF_StateData::fin_dc`

the dc in FIN PDU

Definition at line 306 of file cf\_cfdp\_types.h.

Referenced by `CF_CFDP_R_HandleFileRetention()`, `CF_CFDP_R_Init()`, `CF_CFDP_S1_CheckState_DATA_EOF()`, `CF_CFDP_S_HandleFileRetention()`, `CF_CFDP_S_SubstateRecvFin()`, and `CF_CFDP_SendFin()`.

**11.101.2.6 fin\_fs** `uint8 CF_StateData::fin_fs`

the fs in FIN PDU

Definition at line 307 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_SubstateRecvFin(), and CF\_CFDP\_SendFin().

**11.101.2.7 peer\_cc** `uint8 CF_StateData::peer_cc`

the peer cc from the received FIN or EOF PDU

Definition at line 305 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_S\_SubstateRecvFin(), and CF\_CFDP\_SendAck().

**11.101.2.8 sub\_state** `uint8 CF_StateData::sub_state`

Definition at line 302 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_CheckState\_FILESTORE(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_CheckState\_VALIDATE(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_CheckState\_FILESTORE(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_Tick(), and CF\_CFDP\_S\_Tick\_NewData().

The documentation for this struct was generated from the following file:

- `apps/cf/fsd/src/cf_cfdp_types.h`

## 11.102 CF\_StateFlags Union Reference

Summary of all possible transaction flags (tx and rx)

```
#include <cf_cfdp_types.h>
```

### Data Fields

- `CF_Flags_Common_t com`  
*applies to all transactions*
- `CF_Flags_Rx_t rx`  
*applies to only receive file transactions*
- `CF_Flags_Tx_t tx`  
*applies to only send file transactions*

### 11.102.1 Detailed Description

Summary of all possible transaction flags (tx and rx)

Definition at line 290 of file cf\_cfdp\_types.h.

### 11.102.2 Field Documentation

**11.102.2.1 com** `CF_Flags_Common_t CF_StateFlags::com`

applies to all transactions

Definition at line 292 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_CancelTransaction(), CF\_CFDP\_DoTick(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_GetClass(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_R2\_Recv(), CF\_CFDP\_R\_AckTimerTick(),

CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_FILESTORE(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_AckTimerTick(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_S\_Tick\_NewData(), CF\_CFDP\_SendMd(), CF\_CFDP\_StartRxTransaction(), CF\_DequeueTransaction(), CF\_DoSuspRes\_Txn(), CF\_InsertSortPrio(), and CF\_MoveTransaction().

### 11.102.2.2 rx [CF\\_Flags\\_Rx\\_t](#) CF\_StateFlags::rx

applies to only receive file transactions

Definition at line 293 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_SubstateRecvMd(), and CF\_CFDP\_R\_Tick\_Maintenance().

### 11.102.2.3 tx [CF\\_Flags\\_Tx\\_t](#) CF\_StateFlags::tx

applies to only send file transactions

Definition at line 294 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_S\_Tick\_Maintenance(), CF\_CFDP\_S\_Tick\_Nak(), and CF\_CFDP\_TxFile().

The documentation for this union was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.103 CF\_SuspendCmd Struct Reference

Suspend command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CF\\_Transaction\\_Payload\\_t](#) Payload

### 11.103.1 Detailed Description

Suspend command structure.

For command details see [CF\\_SUSPEND\\_CC](#)

Definition at line 247 of file default\_cf\_msgstruct.h.

### 11.103.2 Field Documentation

#### 11.103.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_SuspendCmd::CommandHeader

Command header.

Definition at line 249 of file default\_cf\_msgstruct.h.

**11.103.2.2 Payload** [CF\\_Transaction\\_Payload\\_t](#) CF\_SuspendCmd::Payload

Definition at line 250 of file default\_cf\_msgstruct.h.

Referenced by CF\_SuspendCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.104 CF\_ThawCmd Struct Reference

Thaw command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*
- [CF\\_UnionArgs\\_Payload\\_t Payload](#)  
*Generic command arguments.*

### 11.104.1 Detailed Description

Thaw command structure.

For command details see [CF\\_THAW\\_CC](#)

Definition at line 126 of file default\_cf\_msgstruct.h.

### 11.104.2 Field Documentation

#### 11.104.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_ThawCmd::CommandHeader

Command header.

Definition at line 128 of file default\_cf\_msgstruct.h.

#### 11.104.2.2 Payload [CF\\_UnionArgs\\_Payload\\_t](#) CF\_ThawCmd::Payload

Generic command arguments.

Definition at line 129 of file default\_cf\_msgstruct.h.

Referenced by CF\_ThawCmd().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.105 CF\_Timer Struct Reference

Basic CF timer object.

```
#include <cf_timer.h>
```

### Data Fields

- [CF\\_Timer\\_Ticks\\_t tick](#)  
*expires when reaches 0*

### 11.105.1 Detailed Description

Basic CF timer object.

Definition at line 48 of file cf\_timer.h.

### 11.105.2 Field Documentation

#### 11.105.2.1 tick `CF_Timer_Ticks_t` CF\_Timer::tick

expires when reaches 0

Definition at line 50 of file cf\_timer.h.

Referenced by CF\_Timer\_Expired(), CF\_Timer\_InitRelSec(), and CF\_Timer\_Tick().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_timer.h](#)

## 11.106 CF\_Transaction Struct Reference

Transaction state object.

```
#include <cf_cfdp_types.h>
```

### Data Fields

- `CF_TxnState_t state`  
*each engine is commanded to do something, which is the overall state*
- `CF_History_t * history`  
*weird, holds active filenames and possibly other info*
- `CF_ChunkWrapper_t * chunks`  
*for gap tracking, only used on class 2*
- `CF_Timer_t inactivity_timer`  
*set to the overall inactivity timer of a remote*
- `CF_Timer_t ack_timer`  
*called ack\_timer, but is also nak\_timer*
- `CF_FileSize_t fsize`  
*lseek() should be 64-bit on 64-bit system, but osal limits to 32-bit*
- `CF_FileSize_t foofs`  
*offset into file for next read*
- `osal_id_t fd`
- `CF_Crc_t crc`
- `bool reliable_mode`
- `uint8 keep`
- `uint8 chan_num`  
*if ever more than one engine, this may need to change to pointer*
- `uint8 priority`
- `CF_CListNode_t cl_node`
- `CF_Playback_t * pb`  
*NULL if transaction does not belong to a playback.*
- `CF_StateData_t state_data`
- `CF_StateFlags_t flags`  
*State flags.*

### 11.106.1 Detailed Description

Transaction state object.

This keeps the state of CF file transactions

Definition at line 320 of file cf\_cfdp\_types.h.

### 11.106.2 Field Documentation

#### 11.106.2.1 ack\_timer `CF_Timer_t` CF\_Transaction::ack\_timer

called ack\_timer, but is also nak\_timer

Definition at line 327 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_R\_AckTimerTick(), and CF\_CFDP\_S\_AckTimerTick().

#### 11.106.2.2 chan\_num `uint8` CF\_Transaction::chan\_num

if ever more than one engine, this may need to change to pointer

Definition at line 337 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_InitTxnTxFile(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvDrop(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvHold(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_S\_Tick\_NewData(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendEotPkt(), CF\_CFDP\_SendFd(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_SendNak(), CF\_DequeueTransaction(), CF\_FreeTransaction(), CF\_GetChannelFromTxn(), CF\_InsertSortPrio(), and CF\_MoveTransaction().

#### 11.106.2.3 chunks `CF_ChunkWrapper_t*` CF\_Transaction::chunks

for gap tracking, only used on class 2

Definition at line 325 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_AllocChunkList(), CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_Tick\_Nak(), CF\_CFDP\_SetupRxTransaction(), and CF\_CFDP\_SetupTxTransaction().

#### 11.106.2.4 cl\_node `CF_CListNode_t` CF\_Transaction::cl\_node

Definition at line 340 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_StartRxTransaction(), CF\_DequeueTransaction(), CF\_FindUnusedTransaction(), CF\_FreeTransaction(), CF\_InsertSortPrio(), and CF\_MoveTransaction().

#### 11.106.2.5 crc `CF_Crc_t` CF\_Transaction::crc

Definition at line 333 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SendEof(), and CF\_CFDP\_SendEotPkt().

**11.106.2.6 fd osal\_id\_t CF\_Transaction::fd**

Definition at line 331 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_CloseFiles(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S\_Init(), and CF\_CFDP\_S\_SendFileData().

**11.106.2.7 flags CF\_StateFlags\_t CF\_Transaction::flags**

State flags.

**Note**

The flags here look a little strange, because there are different flags for TX and RX. Both types share the same type of flag, though. Since RX flags plus the global flags is over one byte, storing them this way allows 2 bytes to cover all possible flags. Please ignore the duplicate declarations of the "all" flags.

Definition at line 354 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_CancelTransaction(), CF\_CFDP\_DoTick(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_GetClass(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_R2\_Recv(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_CheckState\_FILESTORE(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_SubstateRecvMd(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_R\_Tick\_Maintenance(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_AckTimerTick(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_S\_Tick\_Maintenance(), CF\_CFDP\_S\_Tick\_Nak(), CF\_CFDP\_S\_Tick\_NewData(), CF\_CFDP\_SendMd(), CF\_CFDP\_StartRxTransaction(), CF\_CFDP\_TxFile(), CF\_DequeueTransaction(), CF\_DoSuspRes\_Txn(), CF\_InsertSortPrio(), and CF\_MoveTransaction().

**11.106.2.8 foofs CF\_FileSize\_t CF\_Transaction::foofs**

offset into file for next read

Definition at line 330 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), and CF\_CFDP\_S\_SubstateSendFileData().

**11.106.2.9 fsize CF\_FileSize\_t CF\_Transaction::fsize**

Iseek() should be 64-bit on 64-bit system, but osal limits to 32-bit

Definition at line 329 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_R\_CheckState\_VALIDATE(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_RecvMd(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SubstateSendFileData(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendEotPkt(), and CF\_CFDP\_SendMd().

**11.106.2.10 history CF\_History\_t\* CF\_Transaction::history**

weird, holds active filenames and possibly other info

Definition at line 324 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_AllocChunkList(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_GetTxnStatus(), CF\_CFDP\_IsSender(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_

R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_RecvHold(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendEotPkt(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_SetTxnStatus(), CF\_CFDP\_SetupRxTransaction(), CF\_CFDP\_TxFile(), CF\_CFDP\_TxFile\_Initiate(), CF\_FindTransactionBySequenceNumber\_Impl(), CF\_FindUnusedTransaction(), and CF\_Traverse\_WriteTxnQueueEntryToFile().

#### 11.106.2.11 **inactivity\_timer** `CF_Timer_t` CF\_Transaction::inactivity\_timer

set to the overall inactivity timer of a remote

Definition at line 326 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_ArmlactTimer(), CF\_CFDP\_R\_Tick(), and CF\_CFDP\_S\_Tick().

#### 11.106.2.12 **keep** `uint8` CF\_Transaction::keep

Definition at line 336 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_InitTxnTxFile(), and CF\_CFDP\_S\_HandleFileRetention().

#### 11.106.2.13 **pb** `CF_Playback_t*` CF\_Transaction::pb

NULL if transaction does not belong to a playback.

Definition at line 342 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), and CF\_CFDP\_ProcessPlaybackDirectory().

#### 11.106.2.14 **priority** `uint8` CF\_Transaction::priority

Definition at line 338 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_InitTxnTxFile(), CF\_InsertSortPrio(), and CF\_PrioSearch().

#### 11.106.2.15 **reliable\_mode** `bool` CF\_Transaction::reliable\_mode

Set true if class 2, false in class 1

Definition at line 335 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_GetPrintClass(), CF\_CFDP\_InitTxnTxFile(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_FILESTORE(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_Tick\_Maintenance(), CF\_CFDP\_RxStateDispatch(), CF\_CFDP\_S\_AckTimerTick(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Tick\_Maintenance(), and CF\_CFDP\_SetupRxTransaction().

#### 11.106.2.16 **state** `CF_TxnState_t` CF\_Transaction::state

each engine is commanded to do something, which is the overall state

Definition at line 322 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_FinishTransaction(), CF\_CFDP\_GetAckTxnStatus(), CF\_CFDP\_GetClass(), CF\_CFDP\_InitTxnTxFile(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_ReceivePdu(), CF\_CFDP\_RxStateDispatch(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_SendEotPkt(), CF\_CFDP\_SendMd(), CF\_CFDP\_SetupRxTransaction(), and CF\_FindUnusedTransaction().

#### 11.106.2.17 **state\_data** `CF_StateData_t` CF\_Transaction::state\_data

Definition at line 344 of file cf\_cfdp\_types.h.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_CheckState\_FILESTORE(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_CheckState\_VALIDATE(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_SubstateRecvFileData(), CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_CheckState\_FILESTORE(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_S\_Tick\_NewData(), CF\_CFDP\_SendAck(), and CF\_CFDP\_SendFin().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_cfdp\\_types.h](#)

## 11.107 CF\_Transaction\_Payload Struct Reference

Transaction command structure.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- **[CF\\_TransactionSeq\\_t ts](#)**  
*Transaction sequence number.*
- **[CF\\_EntityId\\_t eid](#)**  
*Entity id.*
- **[uint8 chan](#)**  
*Channel number: 254=use ts, 255=all channels, else channel.*
- **[uint8 spare \[3\]](#)**  
*Alignment spare for 32-bit multiple.*

### 11.107.1 Detailed Description

Transaction command structure.

For command details see [CF\\_SUSPEND\\_CC](#), [CF\\_RESUME\\_CC](#), [CF\\_CANCEL\\_CC](#), [CF\\_ABANDON\\_CC](#)

Definition at line 272 of file default\_cf\_msgdefs.h.

### 11.107.2 Field Documentation

#### 11.107.2.1 **chan** [uint8 CF\\_Transaction\\_Payload::chan](#)

Channel number: 254=use ts, 255=all channels, else channel.

Definition at line 276 of file default\_cf\_msgdefs.h.

Referenced by [CF\\_TsnChanAction\(\)](#).

#### 11.107.2.2 **eid** [CF\\_EntityId\\_t CF\\_Transaction\\_Payload::eid](#)

Entity id.

Definition at line 275 of file default\_cf\_msgdefs.h.

Referenced by [CF\\_TsnChanAction\(\)](#).

**11.107.2.3 spare** `uint8 CF_Transaction_Payload::spare[3]`

Alignment spare for 32-bit multiple.

Definition at line 277 of file default\_cf\_msgdefs.h.

**11.107.2.4 ts** `CF_TransactionSeq_t CF_Transaction_Payload::ts`

Transaction sequence number.

Definition at line 274 of file default\_cf\_msgdefs.h.

Referenced by CF\_TsnChanAction().

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgdefs.h](#)

## 11.108 CF\_Traverse\_PriorityArg Struct Reference

Argument structure for use with [CF\\_CList\\_Traverse\\_R\(\)](#)

```
#include <cf_utils.h>
```

### Data Fields

- `CF_Transaction_t * txn`  
*OUT: holds value of transaction with which to call CF\_CList\_InsertAfter on.*
- `uint8 priority`  
*seeking this priority*

### 11.108.1 Detailed Description

Argument structure for use with [CF\\_CList\\_Traverse\\_R\(\)](#)

This is for searching for transactions of a specific priority

Definition at line 107 of file cf\_utils.h.

### 11.108.2 Field Documentation

**11.108.2.1 priority** `uint8 CF_Traverse_PriorityArg::priority`

seeking this priority

Definition at line 110 of file cf\_utils.h.

Referenced by CF\_PrioSearch().

**11.108.2.2 txn** `CF_Transaction_t* CF_Traverse_PriorityArg::txn`

OUT: holds value of transaction with which to call CF\_CList\_InsertAfter on.

Definition at line 109 of file cf\_utils.h.

Referenced by CF\_InsertSortPrio(), and CF\_PrioSearch().

The documentation for this struct was generated from the following file:

- [apps/cf/fsw/src/cf\\_utils.h](#)

## 11.109 CF\_Traverse\_TransSeqArg Struct Reference

Argument structure for use with CList\_Traverse()

```
#include <cf_utils.h>
```

**Data Fields**

- `CF_TransactionSeq_t transaction_sequence_number`
- `CF_EntityId_t src_eid`
- `CF_Transaction_t * txn`  
*output transaction pointer*

**11.109.1 Detailed Description**

Argument structure for use with `CList_Traverse()`

This identifies a specific transaction sequence number and entity ID. The transaction pointer is set by the implementation. Definition at line 39 of file `cf_utils.h`.

**11.109.2 Field Documentation****11.109.2.1 src\_eid `CF_EntityId_t` CF\_Traverse\_TransSeqArg::src\_eid**

Definition at line 42 of file `cf_utils.h`.

Referenced by `CF_FindTransactionBySequenceNumber_Impl()`.

**11.109.2.2 transaction\_sequence\_number `CF_TransactionSeq_t` CF\_Traverse\_TransSeqArg::transaction→\_sequence\_number**

Definition at line 41 of file `cf_utils.h`.

Referenced by `CF_FindTransactionBySequenceNumber_Impl()`.

**11.109.2.3 txn `CF_Transaction_t*` CF\_Traverse\_TransSeqArg::txn**

output transaction pointer

Definition at line 43 of file `cf_utils.h`.

Referenced by `CF_FindTransactionBySequenceNumber()`, and `CF_FindTransactionBySequenceNumber_Impl()`.

The documentation for this struct was generated from the following file:

- `apps/cf/fsu/src/cf_utils.h`

**11.110 CF\_Traverse\_WriteHistoryFileArg Struct Reference**

Argument structure for use with `CF_Traverse_WriteHistoryQueueEntryToFile()`

```
#include <cf_utils.h>
```

**Data Fields**

- `osal_id_t fd`
- `CF_Direction_t filter_dir`
- `bool error`  
*Will be set to true if any write failed.*
- `uint32 counter`  
*Total number of entries written.*

### 11.110.1 Detailed Description

Argument structure for use with [CF\\_Traverse\\_WriteHistoryQueueEntryToFile\(\)](#)

This is used for writing status files. It contains a designated file descriptor for output and counters.

When traversing history, the list contains all entries, and may need additional filtering for direction (TX/RX) depending on what information the user has requested.

Definition at line 55 of file cf\_utils.h.

### 11.110.2 Field Documentation

#### 11.110.2.1 counter `uint32 CF_Traverse_WriteHistoryFileArg::counter`

Total number of entries written.

Definition at line 61 of file cf\_utils.h.

Referenced by [CF\\_Traverse\\_WriteHistoryQueueEntryToFile\(\)](#), and [CF\\_WriteHistoryQueueDataToFile\(\)](#).

#### 11.110.2.2 error `bool CF_Traverse_WriteHistoryFileArg::error`

Will be set to true if any write failed.

Definition at line 60 of file cf\_utils.h.

Referenced by [CF\\_Traverse\\_WriteHistoryQueueEntryToFile\(\)](#), and [CF\\_WriteHistoryQueueDataToFile\(\)](#).

#### 11.110.2.3 fd `osal_id_t CF_Traverse_WriteHistoryFileArg::fd`

Definition at line 57 of file cf\_utils.h.

Referenced by [CF\\_Traverse\\_WriteHistoryQueueEntryToFile\(\)](#), and [CF\\_WriteHistoryQueueDataToFile\(\)](#).

#### 11.110.2.4 filter\_dir `CF_Direction_t CF_Traverse_WriteHistoryFileArg::filter_dir`

Definition at line 58 of file cf\_utils.h.

Referenced by [CF\\_Traverse\\_WriteHistoryQueueEntryToFile\(\)](#), and [CF\\_WriteHistoryQueueDataToFile\(\)](#).

The documentation for this struct was generated from the following file:

- [apps/cf/fsu/src/cf\\_utils.h](#)

## 11.111 CF\_Traverse\_WriteTxnFileArg Struct Reference

Argument structure for use with [CF\\_Traverse\\_WriteTxnQueueEntryToFile\(\)](#)

```
#include <cf_utils.h>
```

### Data Fields

- `osal_id_t fd`
- `bool error`

*Will be set to true if any write failed.*

- `uint32 counter`

*Total number of entries written.*

### 11.111.1 Detailed Description

Argument structure for use with [CF\\_Traverse\\_WriteTxnQueueEntryToFile\(\)](#)

This is used for writing status files. It contains a designated file descriptor for output and counters.

When traversing transactions, the entire list is written to the file. No additional filtering is necessary, because the queues themselves are limited in what they contain (therefore "pre-filtered" to some degree).

Definition at line 74 of file cf\_utils.h.

### 11.111.2 Field Documentation

#### 11.111.2.1 counter `uint32` CF\_Traverse\_WriteTxnFileArg::counter

Total number of entries written.

Definition at line 79 of file cf\_utils.h.

Referenced by CF\_Traverse\_WriteTxnQueueEntryToFile(), and CF\_WriteTxnQueueDataToFile().

#### 11.111.2.2 error `bool` CF\_Traverse\_WriteTxnFileArg::error

Will be set to true if any write failed.

Definition at line 78 of file cf\_utils.h.

Referenced by CF\_Traverse\_WriteTxnQueueEntryToFile(), and CF\_WriteTxnQueueDataToFile().

#### 11.111.2.3 fd `osal_id_t` CF\_Traverse\_WriteTxnFileArg::fd

Definition at line 76 of file cf\_utils.h.

Referenced by CF\_Traverse\_WriteTxnQueueEntryToFile(), and CF\_WriteTxnQueueDataToFile().

The documentation for this struct was generated from the following file:

- apps/cf/fsw/src/[cf\\_utils.h](#)

## 11.112 CF\_TraverseAll\_Arg Struct Reference

Argument structure for use with [CF\\_TraverseAllTransactions\(\)](#)

```
#include <cf_utils.h>
```

### Data Fields

- [CF\\_TraverseAllTransactions\\_fn\\_t fn](#)  
*internal callback to use for each CList\_Traverse*
- `void *` [context](#)  
*opaque object to pass to internal callback*
- `int32` [counter](#)  
*Running tally of all nodes traversed from all lists.*

### 11.112.1 Detailed Description

Argument structure for use with [CF\\_TraverseAllTransactions\(\)](#)

This basically allows for running a CF\_Traverse on several lists at once

Definition at line 95 of file cf\_utils.h.

### 11.112.2 Field Documentation

**11.112.2.1 context** `void* CF_TraverseAll_Arg::context`  
opaque object to pass to internal callback  
Definition at line 98 of file cf\_utils.h.  
Referenced by CF\_TraverseAllTransactions\_Impl().

**11.112.2.2 counter** `int32 CF_TraverseAll_Arg::counter`  
Running tally of all nodes traversed from all lists.  
Definition at line 99 of file cf\_utils.h.  
Referenced by CF\_TraverseAllTransactions(), and CF\_TraverseAllTransactions\_Impl().

**11.112.2.3 fn** `CF_TraverseAllTransactions_fn_t CF_TraverseAll_Arg::fn`  
internal callback to use for each CList\_Traverse  
Definition at line 97 of file cf\_utils.h.  
Referenced by CF\_TraverseAllTransactions\_Impl().  
The documentation for this struct was generated from the following file:

- `apps/cf/fsw/src/cf_utils.h`

## 11.113 CF\_TxFile\_Payload Struct Reference

Transmit file command structure.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- `uint8 cfdp_class`  
*CFDP class: 0=class 1, 1=class 2.*
- `uint8 keep`  
*Keep file flag: 1=keep, else delete.*
- `uint8 chan_num`  
*Channel number.*
- `uint8 priority`  
*Priority: 0=highest priority.*
- `CF_EntityId_t dest_id`  
*Destination entity id.*
- `char src_filename [CF_FILENAME_MAX_LEN]`  
*Source file/directory name.*
- `char dst_filename [CF_FILENAME_MAX_LEN]`  
*Destination file/directory name.*

### 11.113.1 Detailed Description

Transmit file command structure.

For command details see [CF\\_TX\\_FILE\\_CC](#)

Definition at line 241 of file default\_cf\_msgdefs.h.

### 11.113.2 Field Documentation

**11.113.2.1 cfdp\_class** `uint8` CF\_TxFile\_Payload::cfdp\_class  
CFDP class: 0=class 1, 1=class 2.

Definition at line 243 of file default\_cf\_msgdefs.h.

Referenced by CF\_PlaybackDirCmd(), and CF\_TxFileCmd().

**11.113.2.2 chan\_num** `uint8` CF\_TxFile\_Payload::chan\_num

Channel number.

Definition at line 245 of file default\_cf\_msgdefs.h.

Referenced by CF\_PlaybackDirCmd(), and CF\_TxFileCmd().

**11.113.2.3 dest\_id** `CF_EntityId_t` CF\_TxFile\_Payload::dest\_id

Destination entity id.

Definition at line 247 of file default\_cf\_msgdefs.h.

Referenced by CF\_PlaybackDirCmd(), and CF\_TxFileCmd().

**11.113.2.4 dst\_filename** `char` CF\_TxFile\_Payload::dst\_filename[`CF_FILENAME_MAX_LEN`]

Destination file/directory name.

Definition at line 249 of file default\_cf\_msgdefs.h.

Referenced by CF\_PlaybackDirCmd(), and CF\_TxFileCmd().

**11.113.2.5 keep** `uint8` CF\_TxFile\_Payload::keep

Keep file flag: 1=keep, else delete.

Definition at line 244 of file default\_cf\_msgdefs.h.

Referenced by CF\_PlaybackDirCmd(), and CF\_TxFileCmd().

**11.113.2.6 priority** `uint8` CF\_TxFile\_Payload::priority

Priority: 0=highest priority.

Definition at line 246 of file default\_cf\_msgdefs.h.

Referenced by CF\_PlaybackDirCmd(), and CF\_TxFileCmd().

**11.113.2.7 src\_filename** `char` CF\_TxFile\_Payload::src\_filename[`CF_FILENAME_MAX_LEN`]

Source file/directory name.

Definition at line 248 of file default\_cf\_msgdefs.h.

Referenced by CF\_PlaybackDirCmd(), and CF\_TxFileCmd().

The documentation for this struct was generated from the following file:

- `apps/cf/config/default_cf_msgdefs.h`

## 11.114 CF\_TxFileCmd Struct Reference

Transmit file command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t` CommandHeader
  - Command header.*
- `CF_TxFile_Payload_t` Payload

### 11.114.1 Detailed Description

Transmit file command structure.

For command details see [CF\\_TX\\_FILE\\_CC](#)

Definition at line 214 of file default\_cf\_msgstruct.h.

### 11.114.2 Field Documentation

#### 11.114.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_TxFileCmd::CommandHeader

Command header.

Definition at line 216 of file default\_cf\_msgstruct.h.

#### 11.114.2.2 Payload [CF\\_TxFile\\_Payload\\_t](#) CF\_TxFileCmd::Payload

Definition at line 217 of file default\_cf\_msgstruct.h.

Referenced by CF\_TxFileCmd().

The documentation for this struct was generated from the following file:

- apps/cf/config/default\_cf\_msgstruct.h

## 11.115 CF\_TxnFilenames Struct Reference

Cache of source and destination filename.

```
#include <default_cf_extern_typedefs.h>
```

### Data Fields

- char [src\\_filename](#) [[CF\\_FILENAME\\_MAX\\_LEN](#)]
- char [dst\\_filename](#) [[CF\\_FILENAME\\_MAX\\_LEN](#)]

### 11.115.1 Detailed Description

Cache of source and destination filename.

This pairs a source and destination file name together to be retained for future reference in the transaction/history

Definition at line 67 of file default\_cf\_extern\_typedefs.h.

### 11.115.2 Field Documentation

#### 11.115.2.1 dst\_filename char CF\_TxnFilenames::dst\_filename [[CF\\_FILENAME\\_MAX\\_LEN](#)]

Definition at line 70 of file default\_cf\_extern\_typedefs.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_R\_HandleFile<  
Retention(), CF\_CFDP\_RecvMd(), CF\_CFDP\_SendMd(), CF\_CFDP\_TxFile(), CF\_CFDP\_TxFile\_Initiate(), and CF\_<  
WriteHistoryEntryToFile().

#### 11.115.2.2 src\_filename char CF\_TxnFilenames::src\_filename [[CF\\_FILENAME\\_MAX\\_LEN](#)]

Definition at line 69 of file default\_cf\_extern\_typedefs.h.

Referenced by CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_RecvMd(), CF\_<  
\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_SendMd(), CF\_CFDP\_TxFile(), CF\_CFDP\_TxFile<  
\_Initiate(), and CF\_WriteHistoryEntryToFile().

The documentation for this struct was generated from the following file:

- apps/cf/config/default\_cf\_extern\_typedefs.h

## 11.116 CF\_UnionArgs\_Payload Union Reference

Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- **uint32 dword**  
*Generic uint32 argument.*
- **uint16 hword [2]**  
*Generic uint16 array of arguments.*
- **uint8 byte [4]**  
*Generic uint8 array of arguments.*

### 11.116.1 Detailed Description

Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.

Definition at line 151 of file default\_cf\_msgdefs.h.

### 11.116.2 Field Documentation

#### 11.116.2.1 **byte uint8** CF\_UnionArgs\_Payload::byte[4]

Generic uint8 array of arguments.

Definition at line 155 of file default\_cf\_msgdefs.h.

Referenced by CF\_DoChanAction(), CF\_DoEnableDisablePolldir(), CF\_DoPurgeQueue(), and CF\_ResetCountersCmd().

#### 11.116.2.2 **dword uint32** CF\_UnionArgs\_Payload::dword

Generic uint32 argument.

Definition at line 153 of file default\_cf\_msgdefs.h.

#### 11.116.2.3 **hword uint16** CF\_UnionArgs\_Payload::hword[2]

Generic uint16 array of arguments.

Definition at line 154 of file default\_cf\_msgdefs.h.

The documentation for this union was generated from the following file:

- apps/cf/config/default\_cf\_msgdefs.h

## 11.117 CF\_WakeupCmd Struct Reference

Wake Up Command.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t CommandHeader**  
*Command header.*

### 11.117.1 Detailed Description

Wake Up Command.

Internal notification from SCH with no payload

Definition at line 301 of file default\_cf\_msgstruct.h.

### 11.117.2 Field Documentation

#### 11.117.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CF\_WakeupCmd::CommandHeader

Command header.

Definition at line 303 of file default\_cf\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.118 CF\_WriteQueue\_Payload Struct Reference

Write Queue command structure.

```
#include <default_cf_msgdefs.h>
```

### Data Fields

- [uint8 type](#)  
*Transaction direction: all=0, up=1, down=2.*
- [uint8 chan](#)  
*Channel number.*
- [uint8 queue](#)  
*Queue type: 0=pending, 1=active, 2=history, 3=all.*
- [uint8 spare](#)  
*Alignment spare, puts filename on 32-bit boundary.*
- [char filename \[CF\\_FILENAME\\_MAX\\_LEN\]](#)  
*Filename written to.*

### 11.118.1 Detailed Description

Write Queue command structure.

For command details see [CF\\_WRITE\\_QUEUE\\_CC](#)

Definition at line 257 of file default\_cf\_msgdefs.h.

### 11.118.2 Field Documentation

#### 11.118.2.1 chan [uint8](#) CF\_WriteQueue\_Payload::chan

Channel number.

Definition at line 260 of file default\_cf\_msgdefs.h.

Referenced by CF\_WriteQueueCmd().

**11.118.2.2 filename** `char CF_WriteQueue_Payload::filename[CF_FILENAME_MAX_LEN]`

Filename written to.

Definition at line 264 of file default\_cf\_msgdefs.h.

Referenced by CF\_WriteQueueCmd().

**11.118.2.3 queue** `uint8 CF_WriteQueue_Payload::queue`

Queue type: 0=pending, 1=active, 2=history, 3=all.

Definition at line 261 of file default\_cf\_msgdefs.h.

Referenced by CF\_WriteQueueCmd().

**11.118.2.4 spare** `uint8 CF_WriteQueue_Payload::spare`

Alignment spare, puts filename on 32-bit boundary.

Definition at line 262 of file default\_cf\_msgdefs.h.

**11.118.2.5 type** `uint8 CF_WriteQueue_Payload::type`

Transaction direction: all=0, up=1, down=2.

Definition at line 259 of file default\_cf\_msgdefs.h.

Referenced by CF\_WriteQueueCmd().

The documentation for this struct was generated from the following file:

- apps/cf/config/default\_cf\_msgdefs.h

## 11.119 CF\_WriteQueueCmd Struct Reference

Write Queue command structure.

```
#include <default_cf_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CF_WriteQueue_Payload_t Payload`

### 11.119.1 Detailed Description

Write Queue command structure.

For command details see [CF\\_WRITE\\_QUEUE\\_CC](#)

Definition at line 225 of file default\_cf\_msgstruct.h.

### 11.119.2 Field Documentation

**11.119.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CF_WriteQueueCmd::CommandHeader`

Command header.

Definition at line 227 of file default\_cf\_msgstruct.h.

**11.119.2.2 Payload** `CF_WriteQueue_Payload_t` `CF_WriteQueueCmd::Payload`

Definition at line 228 of file `default_cf_msgstruct.h`.

Referenced by `CF_WriteQueueCmd()`.

The documentation for this struct was generated from the following file:

- [apps/cf/config/default\\_cf\\_msgstruct.h](#)

## 11.120 CFE\_Config\_ArrayValue Struct Reference

Wrapper type for array configuration.

```
#include <cfecfg_api_typedefs.h>
```

### Data Fields

- `size_t NumElements`
- `const void * ElementPtr`

### 11.120.1 Detailed Description

Wrapper type for array configuration.

This is a pair containing a size and pointer that is get/set via a single config table entry

Definition at line 59 of file `cfecfg_api_typedefs.h`.

### 11.120.2 Field Documentation

**11.120.2.1 ElementPtr** `const void* CFE_Config_ArrayValue::ElementPtr`

Definition at line 62 of file `cfecfg_api_typedefs.h`.

**11.120.2.2 NumElements** `size_t CFE_Config_ArrayValue::NumElements`

Definition at line 61 of file `cfecfg_api_typedefs.h`.

The documentation for this struct was generated from the following file:

- [cfe/modules/core\\_api/fsw/inc/cfecfg\\_api\\_typedefs.h](#)

## 11.121 CFE\_Config\_IdNameEntry Struct Reference

```
#include <cfecfg_nametable.h>
```

### Data Fields

- `const char * Name`

### 11.121.1 Detailed Description

Definition at line 33 of file `cfecfg_nametable.h`.

### 11.121.2 Field Documentation

**11.121.2.1 Name** `const char* CFE_Config_IdNameEntry::Name`  
Definition at line 35 of file `cfe_config_nametable.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/config/fsw/inc/cfe_config_nametable.h`

## 11.122 CFE\_Config\_ValueBuffer Union Reference

```
#include <cfe_config_table.h>
```

### Data Fields

- `uint32 AsInteger`
- `const void * AsPointer`

### 11.122.1 Detailed Description

Definition at line 43 of file `cfe_config_table.h`.

### 11.122.2 Field Documentation

**11.122.2.1 AsInteger** `uint32 CFE_Config_ValueBuffer::AsInteger`

Definition at line 45 of file `cfe_config_table.h`.

**11.122.2.2 AsPointer** `const void* CFE_Config_ValueBuffer::AsPointer`

Definition at line 46 of file `cfe_config_table.h`.

The documentation for this union was generated from the following file:

- `cfe/modules/config/fsw/inc/cfe_config_table.h`

## 11.123 CFE\_Config\_ValueEntry Struct Reference

```
#include <cfe_config_table.h>
```

### Data Fields

- `CFE_ConfigType_t ActualType`
- `CFE_Config_ValueBuffer_t Datum`

### 11.123.1 Detailed Description

Definition at line 49 of file `cfe_config_table.h`.

### 11.123.2 Field Documentation

**11.123.2.1 ActualType** `CFE_ConfigType_t CFE_Config_ValueEntry::ActualType`

Definition at line 51 of file `cfe_config_table.h`.

**11.123.2.2 Datum** `CFE_Config_ValueBuffer_t` `CFE_Config_ValueEntry::Datum`

Definition at line 52 of file `cfe_config_table.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/config/fsw/inc/cfe_config_table.h`

## 11.124 CFE\_ES\_AppInfo Struct Reference

Application Information.

```
#include <default_cfe_es_extern_typedefs.h>
```

### Data Fields

- `CFE_ResourceId_t ResourceId`  
*Application or Library ID for this resource.*
- `uint32 Type`  
*The type of App: CORE or EXTERNAL.*
- `char Name [CFE_MISSION_MAX_API_LEN]`  
*The Registered Name of the Application.*
- `char EntryPoint [CFE_MISSION_MAX_API_LEN]`  
*The Entry Point label for the Application.*
- `char FileName [CFE_MISSION_MAX_PATH_LEN]`  
*The Filename of the file containing the Application.*
- `CFE_ES_MemOffset_t StackSize`  
*The Stack Size of the Application.*
- `uint32 AddressesAreValid`  
*Indicates that the Code, Data, and BSS addresses/sizes are valid.*
- `CFE_ES_MemAddress_t CodeAddress`  
*The Address of the Application Code Segment.*
- `CFE_ES_MemOffset_t CodeSize`  
*The Code Size of the Application.*
- `CFE_ES_MemAddress_t DataAddress`  
*The Address of the Application Data Segment.*
- `CFE_ES_MemOffset_t DataSize`  
*The Data Size of the Application.*
- `CFE_ES_MemAddress_t BSSAddress`  
*The Address of the Application BSS Segment.*
- `CFE_ES_MemOffset_t BSSSize`  
*The BSS Size of the Application.*
- `CFE_ES_MemAddress_t StartAddress`  
*The Start Address of the Application.*
- `CFE_ES_ExceptionAction_Enum_t ExceptionAction`  
*What should occur if Application has an exception (Restart Application OR Restart Processor)*
- `CFE_ES_TaskPriority_Atom_t Priority`  
*The Priority of the Application.*
- `CFE_ES_TaskId_t MainTaskId`  
*The Application's Main Task ID.*
- `uint32 ExecutionCounter`  
*The Application's Main Task Execution Counter.*

- char **MainTaskName** [CFE\_MISSION\_MAX\_API\_LEN]  
*The Application's Main Task ID.*
- uint32 **NumOfChildTasks**  
*Number of Child tasks for an App.*

### 11.124.1 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 376 of file default\_cfe\_es\_extern\_typedefs.h.

### 11.124.2 Field Documentation

#### 11.124.2.1 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 392 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.124.2.2 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 402 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.124.2.3 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 404 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.124.2.4 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 394 of file default\_cfe\_es\_extern\_typedefs.h.

#### 11.124.2.5 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 396 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.6 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 398 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.7 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 400 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.8 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 385 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.9 ExceptionAction** `CFE_ES_ExceptionAction_Enum_t` `CFE_ES_AppInfo::ExceptionAction`  
What should occur if Application has an exception (Restart Application OR Restart Processor)

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_ExceptnActn`

Definition at line 408 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.10 ExecutionCounter** `uint32` `CFE_ES_AppInfo::ExecutionCounter`  
The Application's Main Task Execution Counter.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_ExecutionCtr`

Definition at line 415 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.11 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 387 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.12 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 413 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.13 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 417 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.14 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 383 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.15 NumOfChildTasks** `uint32 CFE_ES_AppInfo::NumOfChildTasks`  
Number of Child tasks for an App.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_ChildTasks`

Definition at line 419 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.16 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 411 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.17 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 378 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.18 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 390 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.19 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 406 of file default\_cfe\_es\_extern\_typedefs.h.

**11.124.2.20 Type** `uint32 CFE_ES_AppInfo::Type`

The type of App: CORE or EXTERNAL.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_AppType`

Definition at line 380 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

## 11.125 CFE\_ES\_AppNameCmd\_Payload Struct Reference

Generic application name command payload.

```
#include <default_cfe_es_msgdefs.h>
```

**Data Fields**

- char `Application [CFE_MISSION_MAX_API_LEN]`

*ASCII text string containing Application or Library Name.*

### 11.125.1 Detailed Description

Generic application name command payload.

For command details, see [CFE\\_ES\\_STOP\\_APP\\_CC](#), [CFE\\_ES\\_RESTART\\_APP\\_CC](#), [CFE\\_ES\\_QUERY\\_ONE\\_CC](#)

Definition at line 104 of file default\_cfe\_es\_msgdefs.h.

### 11.125.2 Field Documentation

#### 11.125.2.1 Application `char CFE_ES_AppNameCmd_Payload::Application [CFE_MISSION_MAX_API_LEN]`

ASCII text string containing Application or Library Name.

Definition at line 106 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.126 CFE\_ES\_AppReloadCmd\_Payload Struct Reference

Reload Application Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

**Data Fields**

- char `Application [CFE_MISSION_MAX_API_LEN]`

*ASCII text string containing Application Name.*

- char `AppFileName [CFE_MISSION_MAX_PATH_LEN]`

*Full path and filename of Application's executable image.*

### 11.126.1 Detailed Description

Reload Application Command Payload.

For command details, see [CFE\\_ES\\_RELOAD\\_APP\\_CC](#)

Definition at line 115 of file default\_cfe\_es\_msgdefs.h.

## 11.126.2 Field Documentation

### 11.126.2.1 AppFileName `char CFE_ES_AppReloadCmd_Payload::AppFileName[CFE_MISSION_MAX_PATH_LEN]`

Full path and filename of Application's executable image.

Definition at line 118 of file default\_cfe\_es\_msgdefs.h.

### 11.126.2.2 Application `char CFE_ES_AppReloadCmd_Payload::Application[CFE_MISSION_MAX_API_LEN]`

ASCII text string containing Application Name.

Definition at line 117 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.127 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.*

### 11.127.1 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 473 of file default\_cfe\_es\_extern\_typedefs.h.

## 11.127.2 Field Documentation

### 11.127.2.1 BlockSize `CFE_ES_MemOffset_t CFE_ES_BlockStats::BlockSize`

Number of bytes in each of these blocks.

Definition at line 475 of file default\_cfe\_es\_extern\_typedefs.h.

### 11.127.2.2 NumCreated `uint32 CFE_ES_BlockStats::NumCreated`

Number of Memory Blocks of this size created.

Definition at line 476 of file default\_cfe\_es\_extern\_typedefs.h.

**11.127.2.3 NumFree** `uint32 CFE_ES_BlockStats::NumFree`

Number of Memory Blocks of this size that are free.

Definition at line 477 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`

## 11.128 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.*

### 11.128.1 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 458 of file `default_cfe_es_extern_typedefs.h`.

### 11.128.2 Field Documentation

#### 11.128.2.1 ByteAlignSpare

**uint8 CFE\_ES\_CDSRegDumpRec::ByteAlignSpare[3]**

Spare bytes to ensure structure size is multiple of 4 bytes.

Definition at line 464 of file `default_cfe_es_extern_typedefs.h`.

#### 11.128.2.2 Handle

**CFE\_ES\_CDSHandle\_t CFE\_ES\_CDSRegDumpRec::Handle**

Handle of CDS.

Definition at line 460 of file `default_cfe_es_extern_typedefs.h`.

**11.128.2.3 Name** `char CFE_ES_CDSRegDumpRec::Name[CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`  
Processor Unique Name of CDS.  
Definition at line 463 of file default\_cfe\_es\_extern\_typedefs.h.

**11.128.2.4 Size** `CFE_ES_MemOffset_t CFE_ES_CDSRegDumpRec::Size`  
Size, in bytes, of the CDS memory block.  
Definition at line 461 of file default\_cfe\_es\_extern\_typedefs.h.

**11.128.2.5 Table** `bool CFE_ES_CDSRegDumpRec::Table`  
Flag that indicates whether CDS contains a Critical Table.  
Definition at line 462 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

## 11.129 CFE\_ES\_ClearERLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

### 11.129.1 Detailed Description

Definition at line 61 of file default\_cfe\_es\_msgstruct.h.

### 11.129.2 Field Documentation

**11.129.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_ES_ClearERLogCmd::CommandHeader`  
Command header.  
Definition at line 63 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

## 11.130 CFE\_ES\_ClearSysLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

### 11.130.1 Detailed Description

Definition at line 56 of file default\_cfe\_es\_msgstruct.h.

### 11.130.2 Field Documentation

**11.130.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_ES_ClearSysLogCmd::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

## 11.131 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.*

### 11.131.1 Detailed Description

Delete Critical Data Store Command.

Definition at line 185 of file default\_cfe\_es\_msgstruct.h.

### 11.131.2 Field Documentation

**11.131.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_ES_DeleteCDSCmd::CommandHeader`  
Command header.

Definition at line 187 of file default\_cfe\_es\_msgstruct.h.

**11.131.2.2 Payload** `CFE_ES_DeleteCDSCmd_Payload_t CFE_ES_DeleteCDSCmd::Payload`  
Command payload.

Definition at line 188 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

## 11.132 CFE\_ES\_DeleteCDSCmd\_Payload Struct Reference

Delete Critical Data Store Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- char `CdsName [CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`  
*ASCII text string containing name of CDS to delete.*

### 11.132.1 Detailed Description

Delete Critical Data Store Command Payload.

For command details, see [CFE\\_ES\\_DELETE\\_CDS\\_CC](#)

Definition at line 140 of file default\_cfe\_es\_msgdefs.h.

### 11.132.2 Field Documentation

#### 11.132.2.1 CdsName `char CFE_ES_DeleteCDSCmd_Payload::CdsName[CFE_MISSION_ES_CDS_MAX_FULL_NAME_LEN]`

ASCII text string containing name of CDS to delete.

Definition at line 143 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.133 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.*

### 11.133.1 Detailed Description

Dump CDS Registry Command.

Definition at line 239 of file default\_cfe\_es\_msgstruct.h.

### 11.133.2 Field Documentation

#### 11.133.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_ES_DumpCDSRegistryCmd::CommandHeader`

Command header.

Definition at line 241 of file default\_cfe\_es\_msgstruct.h.

#### 11.133.2.2 Payload `CFE_ES_DumpCDSRegistryCmd_Payload_t CFE_ES_DumpCDSRegistryCmd::Payload`

Command payload.

Definition at line 242 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

## 11.134 CFE\_ES\_DumpCDSRegistryCmd\_Payload Struct Reference

Dump CDS Registry Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

## Data Fields

- char [DumpFilename \[CFE\\_MISSION\\_MAX\\_PATH\\_LEN\]](#)  
*ASCII text string of full path and filename of file CDS Registry is to be written.*

### 11.134.1 Detailed Description

Dump CDS Registry Command Payload.

For command details, see [CFE\\_ES\\_DUMP\\_CDS\\_REGISTRY\\_CC](#)

Definition at line 225 of file default\_cfe\_es\_msgdefs.h.

### 11.134.2 Field Documentation

#### 11.134.2.1 DumpFilename [char CFE\\_ES\\_DumpCDSRegistryCmd\\_Payload::DumpFilename \[CFE\\_MISSION\\_MAX\\_PATH\\_LEN\]](#)

ASCII text string of full path and filename of file CDS Registry is to be written.

Definition at line 227 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.135 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.*

### 11.135.1 Detailed Description

Generic file name command.

Definition at line 89 of file default\_cfe\_es\_msgstruct.h.

### 11.135.2 Field Documentation

#### 11.135.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t CFE\\_ES\\_FileNameCmd::CommandHeader](#)

Command header.

Definition at line 91 of file default\_cfe\_es\_msgstruct.h.

#### 11.135.2.2 Payload [CFE\\_ES\\_FileNameCmd\\_Payload\\_t CFE\\_ES\\_FileNameCmd::Payload](#)

Command payload.

Definition at line 92 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.136 CFE\_ES\_FileNameCmd\_Payload Struct Reference

Generic file name command payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- char **FileName** [CFE\_MISSION\_MAX\_PATH\_LEN]

*ASCII text string containing full path and filename of file in which Application data is to be dumped.*

### 11.136.1 Detailed Description

Generic file name command payload.

This format is shared by several executive services commands. For command details, see [CFE\\_ES\\_QUERY\\_ALL\\_CC](#),

[CFE\\_ES\\_QUERY\\_ALL\\_TASKS\\_CC](#), [CFE\\_ES\\_WRITE\\_SYS\\_LOG\\_CC](#), and [CFE\\_ES\\_WRITE\\_ER\\_LOG\\_CC](#)

Definition at line 58 of file default\_cfe\_es\_msgdefs.h.

### 11.136.2 Field Documentation

#### 11.136.2.1 **FileName** char CFE\_ES\_FileNameCmd\_Payload::FileName[CFE\_MISSION\_MAX\_PATH\_LEN]

ASCII text string containing full path and filename of file in which Application data is to be dumped.

Definition at line 60 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.137 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.*

### 11.137.1 Detailed Description

**Name** Executive Services Housekeeping Packet

Definition at line 272 of file default\_cfe\_es\_msgstruct.h.

### 11.137.2 Field Documentation

#### 11.137.2.1 **Payload** [CFE\\_ES\\_HousekeepingTlm\\_Payload\\_t](#) CFE\_ES\_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 275 of file default\_cfe\_es\_msgstruct.h.

**11.137.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_ES_HousekeepingTlm::TelemetryHeader`  
Telemetry header.

Definition at line 274 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`

## 11.138 CFE\_ES\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- `uint8 CommandCounter`  
*The ES Application Command Counter.*
- `uint8 CommandErrorCounter`  
*The ES Application Command Error Counter.*
- `uint16 CFECOREChecksum`  
*Checksum of cFE Core Code.*
- `uint8 CFEMajorVersion`  
*Major Version Number of cFE.*
- `uint8 CFEMinorVersion`  
*Minor Version Number of cFE.*
- `uint8 CFERevision`  
*Sub-Minor Version Number of cFE.*
- `uint8 CFEMissionRevision`  
*Mission Version Number of cFE.*
- `uint8 OSALMajorVersion`  
*OS Abstraction Layer Major Version Number.*
- `uint8 OSALMinorVersion`  
*OS Abstraction Layer Minor Version Number.*
- `uint8 OSALRevision`  
*OS Abstraction Layer Revision Number.*
- `uint8 OSALMissionRevision`  
*OS Abstraction Layer MissionRevision Number.*
- `uint8 PSPMajorVersion`  
*Platform Support Package Major Version Number.*
- `uint8 PSPMinorVersion`  
*Platform Support Package Minor Version Number.*
- `uint8 PSPRevision`  
*Platform Support Package Revision Number.*
- `uint8 PSPMissionRevision`  
*Platform Support Package MissionRevision Number.*
- `CFE_ES_MemOffset_t SysLogBytesUsed`  
*Total number of bytes used in system log.*
- `CFE_ES_MemOffset_t SysLogSize`  
*Total size of the system log.*
- `uint32 SysLogEntries`  
*Number of entries in the system log.*

- **uint32 SysLogMode**  
*Write/Overwrite Mode.*
- **uint32 ERLogIndex**  
*Current index of the ER Log (wraps around)*
- **uint32 ERLogEntries**  
*Number of entries made in the ER Log since the power on.*
- **uint32 RegisteredCoreApps**  
*Number of Applications registered with ES.*
- **uint32 RegisteredExternalApps**  
*Number of Applications registered with ES.*
- **uint32 RegisteredTasks**  
*Number of Tasks ( main AND child tasks ) registered with ES.*
- **uint32 RegisteredLibs**  
*Number of Libraries registered with ES.*
- **uint32 ResetType**  
*Reset type ( PROCESSOR or POWERON )*
- **uint32 ResetSubtype**  
*Reset Sub Type.*
- **uint32 ProcessorResets**  
*Number of processor resets since last power on.*
- **uint32 MaxProcessorResets**  
*Max processor resets before a power on is done.*
- **uint32 BootSource**  
*Boot source ( as provided from BSP )*
- **uint32 PerfState**  
*Current state of Performance Analyzer.*
- **uint32 PerfMode**  
*Current mode of Performance Analyzer.*
- **uint32 PerfTriggerCount**  
*Number of Times Performance Analyzer has Triggered.*
- **uint32 PerfFilterMask [CFE\_MISSION\_ES\_PERF\_MAX\_IDS/32]**  
*Current Setting of Performance Analyzer Filter Masks.*
- **uint32 PerfTriggerMask [CFE\_MISSION\_ES\_PERF\_MAX\_IDS/32]**  
*Current Setting of Performance Analyzer Trigger Masks.*
- **uint32 PerfDataStart**  
*Identifies First Stored Entry in Performance Analyzer Log.*
- **uint32 PerfDataEnd**  
*Identifies Last Stored Entry in Performance Analyzer Log.*
- **uint32 PerfDataCount**  
*Number of Entries Put Into the Performance Analyzer Log.*
- **uint32 PerfDataToWrite**  
*Number of Performance Analyzer Log Entries Left to be Written to Log Dump File.*
- **CFE\_ES\_MemOffset\_t HeapBytesFree**  
*Number of free bytes remaining in the OS heap.*
- **CFE\_ES\_MemOffset\_t HeapBlocksFree**  
*Number of free blocks remaining in the OS heap.*
- **CFE\_ES\_MemOffset\_t HeapMaxBlockSize**  
*Number of bytes in the largest free block.*

### 11.138.1 Detailed Description

**Name** Executive Services Housekeeping Packet

Definition at line 263 of file default\_cfe\_es\_msgdefs.h.

### 11.138.2 Field Documentation

**11.138.2.1 BootSource** `uint32 CFE_ES_HousekeepingTlm_Payload::BootSource`  
Boot source ( as provided from BSP )

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_BootSource

Definition at line 329 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.2 CFECOREChecksum** `uint16 CFE_ES_HousekeepingTlm_Payload::CFECOREChecksum`  
Checksum of cFE Core Code.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_CKSUM

Definition at line 270 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.3 CFEMajorVersion** `uint8 CFE_ES_HousekeepingTlm_Payload::CFEMajorVersion`  
Major Version Number of cFE.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_CFEMAJORVER

Definition at line 272 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.4 CFEMinorVersion** `uint8 CFE_ES_HousekeepingTlm_Payload::CFEMinorVersion`  
Minor Version Number of cFE.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_CFEMINORVER

Definition at line 274 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.5 CFEMissionRevision** `uint8 CFE_ES_HousekeepingTlm_Payload::CFEMissionRevision`  
Mission Version Number of cFE.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_CFEMISSIONREV

Definition at line 278 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.6 CFERevision** `uint8 CFE_ES_HousekeepingTlm_Payload::CFERevision`  
Sub-Minor Version Number of cFE.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_CFEREVISION

Definition at line 276 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.7 CommandCounter** `uint8` `CFE_ES_HousekeepingTlm_Payload::CommandCounter`  
The ES Application Command Counter.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_CMDPC`

Definition at line 265 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.8 CommandErrorCounter** `uint8` `CFE_ES_HousekeepingTlm_Payload::CommandErrorCounter`  
The ES Application Command Error Counter.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_CMDEC`

Definition at line 267 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.9 ERLogEntries** `uint32` `CFE_ES_HousekeepingTlm_Payload::ERLogEntries`  
Number of entries made in the ER Log since the power on.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_ERLOGENTRIES`

Definition at line 309 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.10 ERLogIndex** `uint32` `CFE_ES_HousekeepingTlm_Payload::ERLogIndex`  
Current index of the ER Log (wraps around)

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_ERLOGINDEX`

Definition at line 307 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.11 HeapBlocksFree** `CFE_ES_MemOffset_t` `CFE_ES_HousekeepingTlm_Payload::HeapBlocksFree`  
Number of free blocks remaining in the OS heap.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_HeapBlocksFree`

Definition at line 354 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.12 HeapBytesFree** `CFE_ES_MemOffset_t` `CFE_ES_HousekeepingTlm_Payload::HeapBytesFree`  
Number of free bytes remaining in the OS heap.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_HeapBytesFree`

Definition at line 352 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.13 HeapMaxBlockSize** `CFE_ES_MemOffset_t` `CFE_ES_HousekeepingTlm_Payload::HeapMaxBlockSize`  
Number of bytes in the largest free block.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_HeapMaxBlkSize`

Definition at line 356 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.14 MaxProcessorResets** `uint32` `CFE_ES_HousekeepingTlm_Payload::MaxProcessorResets`  
Max processor resets before a power on is done.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_MaxProcResets`

Definition at line 327 of file `default_cfe_es_msgdefs.h`.

**11.138.2.15 OSALMajorVersion** `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALMajorVersion`  
OS Abstraction Layer Major Version Number.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_OSMajorVer`

Definition at line 280 of file `default_cfe_es_msgdefs.h`.

**11.138.2.16 OSALMinorVersion** `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALMinorVersion`  
OS Abstraction Layer Minor Version Number.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_Osminorver`

Definition at line 282 of file `default_cfe_es_msgdefs.h`.

**11.138.2.17 OSALMissionRevision** `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALMissionRevision`  
OS Abstraction Layer MissionRevision Number.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_Osmissionrev`

Definition at line 286 of file `default_cfe_es_msgdefs.h`.

**11.138.2.18 OSALRevision** `uint8` `CFE_ES_HousekeepingTlm_Payload::OSALRevision`  
OS Abstraction Layer Revision Number.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_Osrevision`

Definition at line 284 of file `default_cfe_es_msgdefs.h`.

**11.138.2.19 PerfDataCount** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfDataCount`  
Number of Entries Put Into the Performance Analyzer Log.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfDataCnt`

Definition at line 347 of file `default_cfe_es_msgdefs.h`.

**11.138.2.20 PerfDataEnd** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfDataEnd`  
Identifies Last Stored Entry in Performance Analyzer Log.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfDataEnd`

Definition at line 345 of file `default_cfe_es_msgdefs.h`.

**11.138.2.21 PerfDataStart** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfDataStart`  
Identifies First Stored Entry in Performance Analyzer Log.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfDataStart`

Definition at line 343 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.22 PerfDataToWrite** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfDataToWrite`  
Number of Performance Analyzer Log Entries Left to be Written to Log Dump File.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfData2Write`

Definition at line 350 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.23 PerfFilterMask** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfFilterMask` [`CFE_MISSION_ES_PERF_MAX_IDS/32`]  
Current Setting of Performance Analyzer Filter Masks.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfFltrMask[MaskCnt]`

Definition at line 338 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.24 PerfMode** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfMode`  
Current mode of Performance Analyzer.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfMode`

Definition at line 334 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.25 PerfState** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfState`  
Current state of Performance Analyzer.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfState`

Definition at line 332 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.26 PerfTriggerCount** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfTriggerCount`  
Number of Times Performance Analyzer has Triggered.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfTrigCnt`

Definition at line 336 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.27 PerfTriggerMask** `uint32` `CFE_ES_HousekeepingTlm_Payload::PerfTriggerMask` [`CFE_MISSION_ES_PERF_MAX_IDS/32`]  
Current Setting of Performance Analyzer Trigger Masks.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_PerfTrigMask[MaskCnt]`

Definition at line 341 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.28 ProcessorResets** `uint32 CFE_ES_HousekeepingTlm_Payload::ProcessorResets`  
Number of processor resets since last power on.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_ProcResetCnt

Definition at line 325 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.29 PSPMajorVersion** `uint8 CFE_ES_HousekeepingTlm_Payload::PSPMajorVersion`  
Platform Support Package Major Version Number.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_PSPMAJORVER

Definition at line 289 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.30 PSPMinorVersion** `uint8 CFE_ES_HousekeepingTlm_Payload::PSPMinorVersion`  
Platform Support Package Minor Version Number.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_PSPMINORVER

Definition at line 291 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.31 PSPMissionRevision** `uint8 CFE_ES_HousekeepingTlm_Payload::PSPMissionRevision`  
Platform Support Package MissionRevision Number.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_PSPMISSIONREV

Definition at line 295 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.32 PSPRevision** `uint8 CFE_ES_HousekeepingTlm_Payload::PSPRevision`  
Platform Support Package Revision Number.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_PSPREVISION

Definition at line 293 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.33 RegisteredCoreApps** `uint32 CFE_ES_HousekeepingTlm_Payload::RegisteredCoreApps`  
Number of Applications registered with ES.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_RegCoreApps

Definition at line 312 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.34 RegisteredExternalApps** `uint32 CFE_ES_HousekeepingTlm_Payload::RegisteredExternalApps`  
Number of Applications registered with ES.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_RegExtApps

Definition at line 314 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.35 RegisteredLibs** `uint32` `CFE_ES_HousekeepingTlm_Payload::RegisteredLibs`  
Number of Libraries registered with ES.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_RegLibs`

Definition at line 318 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.36 RegisteredTasks** `uint32` `CFE_ES_HousekeepingTlm_Payload::RegisteredTasks`  
Number of Tasks ( main AND child tasks ) registered with ES.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_RegTasks`

Definition at line 316 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.37 ResetSubtype** `uint32` `CFE_ES_HousekeepingTlm_Payload::ResetSubtype`  
Reset Sub Type.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_ResetSubtype`

Definition at line 323 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.38 ResetType** `uint32` `CFE_ES_HousekeepingTlm_Payload::ResetType`  
Reset type ( PROCESSOR or POWERON )

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_ResetType`

Definition at line 321 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.39 SysLogBytesUsed** `CFE_ES_MemOffset_t` `CFE_ES_HousekeepingTlm_Payload::SysLogBytesUsed`  
Total number of bytes used in system log.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_SYSLOGBYTEUSED`

Definition at line 298 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.40 SysLogEntries** `uint32` `CFE_ES_HousekeepingTlm_Payload::SysLogEntries`  
Number of entries in the system log.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_SYSLOGENTRIES`

Definition at line 302 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.41 SysLogMode** `uint32` `CFE_ES_HousekeepingTlm_Payload::SysLogMode`  
Write/Overwrite Mode.

**Telemetry Mnemonic(s)** `$sc_$cpu_ES_SYSLOGMODE`

Definition at line 304 of file default\_cfe\_es\_msgdefs.h.

**11.138.2.42 SysLogSize** `CFE_ES_MemOffset_t` `CFE_ES_HousekeepingTlm_Payload::SysLogSize`  
Total size of the system log.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_SYSLOGSIZE

Definition at line 300 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.139 CFE\_ES\_MemAddress\_t Struct Reference

Type used for memory addresses in command and telemetry messages.

```
#include <example_2x32bit_cfe_es_memaddress.h>
```

### Data Fields

- `uint32 bits [2]`

#### 11.139.1 Detailed Description

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 117 of file example\_2x32bit\_cfe\_es\_memaddress.h.

#### 11.139.2 Field Documentation

##### 11.139.2.1 bits `uint32 CFE_ES_MemAddress_t::bits[2]`

Definition at line 119 of file example\_2x32bit\_cfe\_es\_memaddress.h.

Referenced by `CFE_ES_MemAddress_FromNative()`, and `CFE_ES_MemAddress_ToNative()`.

The documentation for this struct was generated from the following file:

- cfe/cmake/sample\_defs/example\_2x32bit\_cfe\_es\_memaddress.h

## 11.140 CFE\_ES\_MemOffset\_t Struct Reference

Type used for memory sizes and offsets in commands and telemetry.

```
#include <example_2x32bit_cfe_es_memaddress.h>
```

### Data Fields

- `uint32 bits [2]`

### 11.140.1 Detailed Description

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.

#### Note

It is defined as two uint32s rather than a uint64 in case it is not aligned in the parent structure.

Definition at line 60 of file example\_2x32bit\_cfe\_es\_memaddress.h.

### 11.140.2 Field Documentation

#### 11.140.2.1 bits uint32 CFE\_ES\_MemOffset\_t::bits[2]

Definition at line 62 of file example\_2x32bit\_cfe\_es\_memaddress.h.

Referenced by CFE\_ES\_MemOffset\_FromNative(), and CFE\_ES\_MemOffset\_ToNative().

The documentation for this struct was generated from the following file:

- cfe/cmake/sample\_defs/example\_2x32bit\_cfe\_es\_memaddress.h

## 11.141 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.*

### 11.141.1 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 488 of file default\_cfe\_es\_extern\_typedefs.h.

### 11.141.2 Field Documentation

**11.141.2.1 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 498 of file `default_cfe_es_extern_typedefs.h`.

**11.141.2.2 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 494 of file `default_cfe_es_extern_typedefs.h`.

**11.141.2.3 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 492 of file `default_cfe_es_extern_typedefs.h`.

**11.141.2.4 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 496 of file `default_cfe_es_extern_typedefs.h`.

**11.141.2.5 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 490 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`

### 11.142 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.*

### 11.142.1 Detailed Description

**Name** Memory Pool Statistics Packet

Definition at line 263 of file default\_cfe\_es\_msgstruct.h.

### 11.142.2 Field Documentation

**11.142.2.1 Payload** `CFE_ES_PoolStatsTlm_Payload_t` `CFE_ES_MemStatsTlm::Payload`  
Telemetry payload.

Definition at line 266 of file default\_cfe\_es\_msgstruct.h.

**11.142.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_ES_MemStatsTlm::TelemetryHeader`  
Telemetry header.

Definition at line 265 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default\\_cfe\\_es\\_msgstruct.h](#)

## 11.143 CFE\_ES\_NoopCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

### 11.143.1 Detailed Description

Definition at line 46 of file default\_cfe\_es\_msgstruct.h.

### 11.143.2 Field Documentation

**11.143.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_NoopCmd::CommandHeader`  
Command header.

Definition at line 48 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](#)

## 11.144 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.*

### 11.144.1 Detailed Description

**Name** Single Application Information Packet

Definition at line 254 of file default\_cfe\_es\_msgstruct.h.

### 11.144.2 Field Documentation

#### 11.144.2.1 Payload [CFE\\_ES\\_OneAppTlm\\_Payload\\_t](#) CFE\_ES\_OneAppTlm::Payload

Telemetry payload.

Definition at line 257 of file default\_cfe\_es\_msgstruct.h.

#### 11.144.2.2 TelemetryHeader [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_ES\_OneAppTlm::TelemetryHeader

Telemetry header.

Definition at line 256 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

## 11.145 CFE\_ES\_OneAppTlm\_Payload Struct Reference

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- [CFE\\_ES\\_AppInfo\\_t](#) AppInfo

*For more information, see [CFE\\_ES\\_AppInfo\\_t](#).*

### 11.145.1 Detailed Description

**Name** Single Application Information Packet

Definition at line 243 of file default\_cfe\_es\_msgdefs.h.

### 11.145.2 Field Documentation

#### 11.145.2.1 AppInfo [CFE\\_ES\\_AppInfo\\_t](#) CFE\_ES\_OneAppTlm\_Payload::AppInfo

For more information, see [CFE\\_ES\\_AppInfo\\_t](#).

Definition at line 245 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.146 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.*

**11.146.1 Detailed Description**

Overwrite/Discard System Log Configuration Command Payload.  
Definition at line 126 of file default\_cfe\_es\_msgstruct.h.

**11.146.2 Field Documentation****11.146.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_OverWriteSysLogCmd::CommandHeader**  
Command header.

Definition at line 128 of file default\_cfe\_es\_msgstruct.h.

**11.146.2.2 Payload [CFE\\_ES\\_OverWriteSysLogCmd\\_Payload\\_t](#) CFE\_ES\_OverWriteSysLogCmd::Payload**  
Command payload.

Definition at line 129 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

**11.147 CFE\_ES\_OverWriteSysLogCmd\_Payload Struct Reference**

Overwrite/Discard System Log Configuration Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

**Data Fields**

- [uint32](#) Mode  
*CFE\_ES\_LogMode\_DISCARD=Throw away most recent messages, CFE\_ES\_LogMode\_OVERWRITE=Overwrite oldest with most recent*

**11.147.1 Detailed Description**

Overwrite/Discard System Log Configuration Command Payload.  
For command details, see [CFE\\_ES\\_OVER\\_WRITE\\_SYS\\_LOG\\_CC](#)  
Definition at line 70 of file default\_cfe\_es\_msgdefs.h.

**11.147.2 Field Documentation****11.147.2.1 Mode [uint32](#) CFE\_ES\_OverWriteSysLogCmd\_Payload::Mode**

*CFE\_ES\_LogMode\_DISCARD=Throw away most recent messages, CFE\_ES\_LogMode\_OVERWRITE=Overwrite oldest with most recent*

Definition at line 72 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.148 CFE\_ES\_PoolAlign Union Reference

Pool Alignment.

```
#include <cfes_api_typedefs.h>
```

### Data Fields

- void \* **Ptr**  
*Aligned pointer.*
- long long int **LongInt**  
*Aligned Long Integer.*
- long double **LongDouble**  
*Aligned Long Double.*

### 11.148.1 Detailed Description

Pool Alignment.

Union that can be used for minimum memory alignment of ES memory pools on the target. It contains the longest native data types such that the alignment of this structure should reflect the largest possible alignment requirements for any data on this processor.

Definition at line 145 of file cfe\_es\_api\_typedefs.h.

### 11.148.2 Field Documentation

#### 11.148.2.1 **LongDouble** long double CFE\_ES\_PoolAlign::LongDouble

Aligned Long Double.

Definition at line 150 of file cfe\_es\_api\_typedefs.h.

#### 11.148.2.2 **LongInt** long long int CFE\_ES\_PoolAlign::LongInt

Aligned Long Integer.

Definition at line 149 of file cfe\_es\_api\_typedefs.h.

#### 11.148.2.3 **Ptr** void\* CFE\_ES\_PoolAlign::Ptr

Aligned pointer.

Definition at line 147 of file cfe\_es\_api\_typedefs.h.

The documentation for this union was generated from the following file:

- cfe/modules/core\_api/fsw/inc/cfe\_es\_api\_typedefs.h

## 11.149 CFE\_ES\_PoolStatsTlm\_Payload Struct Reference

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- **CFE\_ES\_MemHandle\_t** PoolHandle  
*Handle of memory pool whose stats are being telemetered.*
- **CFE\_ES\_MemPoolStats\_t** PoolStats  
*For more info, see CFE\_ES\_MemPoolStats\_t.*

### 11.149.1 Detailed Description

**Name** Memory Pool Statistics Packet

Definition at line 251 of file default\_cfe\_es\_msgdefs.h.

### 11.149.2 Field Documentation

**11.149.2.1 PoolHandle** [CFE\\_ES\\_MemHandle\\_t](#) CFE\_ES\_PoolStatsTlm\_Payload::PoolHandle  
Handle of memory pool whose stats are being telemetered.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_ES\_PoolHandle

Definition at line 253 of file default\_cfe\_es\_msgdefs.h.

**11.149.2.2 PoolStats** [CFE\\_ES\\_MemPoolStats\\_t](#) CFE\_ES\_PoolStatsTlm\_Payload::PoolStats  
For more info, see [CFE\\_ES\\_MemPoolStats\\_t](#).

Definition at line 255 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.150 CFE\_ES\_QueryAllCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_FileNameCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.150.1 Detailed Description

Definition at line 99 of file default\_cfe\_es\_msgstruct.h.

### 11.150.2 Field Documentation

**11.150.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_QueryAllCmd::CommandHeader  
Command header.

Definition at line 101 of file default\_cfe\_es\_msgstruct.h.

**11.150.2.2 Payload** [CFE\\_ES\\_FileNameCmd\\_Payload\\_t](#) CFE\_ES\_QueryAllCmd::Payload  
Command payload.

Definition at line 102 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

## 11.151 CFE\_ES\_QueryAllTasksCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_FileNameCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.151.1 Detailed Description

Definition at line 105 of file default\_cfe\_es\_msgstruct.h.

### 11.151.2 Field Documentation

#### 11.151.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_QueryAllTasksCmd::CommandHeader

Command header.

Definition at line 107 of file default\_cfe\_es\_msgstruct.h.

#### 11.151.2.2 Payload [CFE\\_ES\\_FileNameCmd\\_Payload\\_t](#) CFE\_ES\_QueryAllTasksCmd::Payload

Command payload.

Definition at line 108 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

## 11.152 CFE\_ES\_QueryOneCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_AppNameCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.152.1 Detailed Description

Definition at line 158 of file default\_cfe\_es\_msgstruct.h.

### 11.152.2 Field Documentation

#### 11.152.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_QueryOneCmd::CommandHeader

Command header.

Definition at line 160 of file default\_cfe\_es\_msgstruct.h.

**11.152.2.2 Payload** `CFE_ES_AppNameCmd_Payload_t` `CFE_ES_QueryOneCmd::Payload`  
Command payload.

Definition at line 161 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](#)

## 11.153 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.*

### 11.153.1 Detailed Description

Reload Application Command.

Definition at line 167 of file default\_cfe\_es\_msgstruct.h.

### 11.153.2 Field Documentation

**11.153.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_ReloadAppCmd::CommandHeader`  
Command header.

Definition at line 169 of file default\_cfe\_es\_msgstruct.h.

**11.153.2.2 Payload** `CFE_ES_AppReloadCmd_Payload_t` `CFE_ES_ReloadAppCmd::Payload`  
Command payload.

Definition at line 170 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default\\_cfe\\_es\\_msgstruct.h](#)

## 11.154 CFE\_ES\_ResetCountersCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

### 11.154.1 Detailed Description

Definition at line 51 of file default\_cfe\_es\_msgstruct.h.

### 11.154.2 Field Documentation

**11.154.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_ResetCountersCmd::CommandHeader`  
Command header.

Definition at line 53 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.155 CFE\_ES\_ResetPRCountCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

### 11.155.1 Detailed Description

Definition at line 66 of file `default_cfe_es_msgstruct.h`.

### 11.155.2 Field Documentation

**11.155.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_ResetPRCountCmd::CommandHeader`  
Command header.

Definition at line 68 of file `default_cfe_es_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgstruct.h`

## 11.156 CFE\_ES\_RestartAppCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_ES_AppNameCmd_Payload_t Payload`  
*Command payload.*

### 11.156.1 Detailed Description

Definition at line 152 of file `default_cfe_es_msgstruct.h`.

### 11.156.2 Field Documentation

**11.156.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_RestartAppCmd::CommandHeader  
Command header.  
Definition at line 154 of file default\_cfe\_es\_msgstruct.h.

**11.156.2.2 Payload** [CFE\\_ES\\_AppNameCmd\\_Payload\\_t](#) CFE\_ES\_RestartAppCmd::Payload  
Command payload.  
Definition at line 155 of file default\_cfe\_es\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.157 CFE\_ES\_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.*

### 11.157.1 Detailed Description

Restart cFE Command.

Definition at line 79 of file default\_cfe\_es\_msgstruct.h.

### 11.157.2 Field Documentation

**11.157.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_RestartCmd::CommandHeader  
Command header.  
Definition at line 81 of file default\_cfe\_es\_msgstruct.h.

**11.157.2.2 Payload** [CFE\\_ES\\_RestartCmd\\_Payload\\_t](#) CFE\_ES\_RestartCmd::Payload  
Command payload.  
Definition at line 82 of file default\_cfe\_es\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.158 CFE\_ES\_RestartCmd\_Payload Struct Reference

Restart cFE Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- [uint16](#) RestartType  
*CFE\_PSP\_RST\_TYPE\_PROCESSOR=Processor Reset or CFE\_PSP\_RST\_TYPE\_POWERON=Power-On Reset*

### 11.158.1 Detailed Description

Restart cFE Command Payload.

For command details, see [CFE\\_ES\\_RESTART\\_CC](#)

Definition at line 44 of file default\_cfe\_es\_msgdefs.h.

### 11.158.2 Field Documentation

#### 11.158.2.1 RestartType `uint16 CFE_ES_RestartCmd_Payload::RestartType`

`CFE_PSP_RST_TYPE_PROCESSOR`=Processor Reset or `CFE_PSP_RST_TYPE_POWERON`=Power-On Reset

Definition at line 46 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default\\_cfe\\_es\\_msgdefs.h](#)

## 11.159 CFE\_ES\_SendHkCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)  
*Command header.*

### 11.159.1 Detailed Description

Definition at line 71 of file default\_cfe\_es\_msgstruct.h.

### 11.159.2 Field Documentation

#### 11.159.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_ES_SendHkCmd::CommandHeader`

Command header.

Definition at line 73 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/es/config/default\\_cfe\\_es\\_msgstruct.h](#)

## 11.160 CFE\_ES\_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.*

### 11.160.1 Detailed Description

Send Memory Pool Statistics Command.

Definition at line 230 of file default\_cfe\_es\_msgstruct.h.

### 11.160.2 Field Documentation

#### 11.160.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_SendMemPoolStatsCmd::CommandHeader

Command header.

Definition at line 232 of file default\_cfe\_es\_msgstruct.h.

#### 11.160.2.2 Payload [CFE\\_ES\\_SendMemPoolStatsCmd\\_Payload\\_t](#) CFE\_ES\_SendMemPoolStatsCmd::Payload

Command payload.

Definition at line 233 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.161 CFE\_ES\_SendMemPoolStatsCmd\_Payload Struct Reference

Send Memory Pool Statistics Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- char [Application](#) [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]
  - *RESERVED - should be all zeroes*
- [CFE\\_ES\\_MemHandle\\_t](#) [PoolHandle](#)

*Handle of Pool whose statistics are to be telemetered.*

### 11.161.1 Detailed Description

Send Memory Pool Statistics Command Payload.

For command details, see [CFE\\_ES\\_SEND\\_MEM\\_POOL\\_STATS\\_CC](#)

Definition at line 213 of file default\_cfe\_es\_msgdefs.h.

### 11.161.2 Field Documentation

#### 11.161.2.1 Application char CFE\_ES\_SendMemPoolStatsCmd\_Payload::Application [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]

- RESERVED - should be all zeroes

Definition at line 215 of file default\_cfe\_es\_msgdefs.h.

#### 11.161.2.2 PoolHandle [CFE\\_ES\\_MemHandle\\_t](#) CFE\_ES\_SendMemPoolStatsCmd\_Payload::PoolHandle

Handle of Pool whose statistics are to be telemetered.

Definition at line 216 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.162 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.*

### 11.162.1 Detailed Description

Set Maximum Processor Reset Count Command.

Definition at line 176 of file default\_cfe\_es\_msgstruct.h.

### 11.162.2 Field Documentation

#### 11.162.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_SetMaxPRCountCmd::CommandHeader

Command header.

Definition at line 178 of file default\_cfe\_es\_msgstruct.h.

#### 11.162.2.2 Payload [CFE\\_ES\\_SetMaxPRCountCmd\\_Payload\\_t](#) CFE\_ES\_SetMaxPRCountCmd::Payload

Command payload.

Definition at line 179 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.163 CFE\_ES\_SetMaxPRCountCmd\_Payload Struct Reference

Set Maximum Processor Reset Count Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- [uint16](#) MaxPRCount  
*New maximum number of Processor Resets before an automatic Power-On Reset is performed.*

### 11.163.1 Detailed Description

Set Maximum Processor Reset Count Command Payload.

For command details, see [CFE\\_ES\\_SET\\_MAX\\_PR\\_COUNT\\_CC](#)

Definition at line 128 of file default\_cfe\_es\_msgdefs.h.

### 11.163.2 Field Documentation

**11.163.2.1 MaxPRCount** `uint16 CFE_ES_SetMaxPRCountCmd_Payload::MaxPRCount`

New maximum number of Processor Resets before an automatic Power-On Reset is performed.

Definition at line 130 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/es/config/default_cfe_es_msgdefs.h`

## 11.164 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.*

### 11.164.1 Detailed Description

Set Performance Analyzer Filter Mask Command.

Definition at line 212 of file default\_cfe\_es\_msgstruct.h.

### 11.164.2 Field Documentation

#### 11.164.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_ES_SetPerfFilterMaskCmd::CommandHeader`

Command header.

Definition at line 214 of file default\_cfe\_es\_msgstruct.h.

#### 11.164.2.2 Payload

`CFE_ES_SetPerfFilterMaskCmd_Payload_t CFE_ES_SetPerfFilterMaskCmd::Payload`

Command payload.

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`

## 11.165 CFE\_ES\_SetPerfFilterMaskCmd\_Payload Struct Reference

Set Performance Analyzer Filter Mask Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- `uint32 FilterMaskNum`  
*Index into array of Filter Masks.*
- `uint32 FilterMask`  
*New Mask for specified entry in array of Filter Masks.*

### 11.165.1 Detailed Description

Set Performance Analyzer Filter Mask Command Payload.

For command details, see [CFE\\_ES\\_SET\\_PERF\\_FILTER\\_MASK\\_CC](#)

Definition at line 189 of file default\_cfe\_es\_msgdefs.h.

### 11.165.2 Field Documentation

#### 11.165.2.1 FilterMask `uint32` CFE\_ES\_SetPerfFilterMaskCmd\_Payload::FilterMask

New Mask for specified entry in array of Filter Masks.

Definition at line 192 of file default\_cfe\_es\_msgdefs.h.

#### 11.165.2.2 FilterMaskNum `uint32` CFE\_ES\_SetPerfFilterMaskCmd\_Payload::FilterMaskNum

Index into array of Filter Masks.

Definition at line 191 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.166 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.*

### 11.166.1 Detailed Description

Set Performance Analyzer Trigger Mask Command.

Definition at line 221 of file default\_cfe\_es\_msgstruct.h.

### 11.166.2 Field Documentation

#### 11.166.2.1 CommandHeader `CFE_MSG_CommandHeader_t` CFE\_ES\_SetPerfTriggerMaskCmd::CommandHeader

Command header.

Definition at line 223 of file default\_cfe\_es\_msgstruct.h.

#### 11.166.2.2 Payload `CFE_ES_SetPerfTrigMaskCmd_Payload_t` CFE\_ES\_SetPerfTriggerMaskCmd::Payload

Command payload.

Definition at line 224 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

## 11.167 CFE\_ES\_SetPerfTrigMaskCmd\_Payload Struct Reference

Set Performance Analyzer Trigger Mask Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- `uint32 TriggerMaskNum`  
*Index into array of Trigger Masks.*
- `uint32 TriggerMask`  
*New Mask for specified entry in array of Trigger Masks.*

### 11.167.1 Detailed Description

Set Performance Analyzer Trigger Mask Command Payload.

For command details, see [CFE\\_ES\\_SET\\_PERF\\_TRIGGER\\_MASK\\_CC](#)

Definition at line 201 of file default\_cfe\_es\_msgdefs.h.

### 11.167.2 Field Documentation

#### 11.167.2.1 TriggerMask `uint32 CFE_ES_SetPerfTrigMaskCmd_Payload::TriggerMask`

New Mask for specified entry in array of Trigger Masks.

Definition at line 204 of file default\_cfe\_es\_msgdefs.h.

#### 11.167.2.2 TriggerMaskNum `uint32 CFE_ES_SetPerfTrigMaskCmd_Payload::TriggerMaskNum`

Index into array of Trigger Masks.

Definition at line 203 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.168 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.*

### 11.168.1 Detailed Description

Start Application Command.

Definition at line 135 of file default\_cfe\_es\_msgstruct.h.

### 11.168.2 Field Documentation

**11.168.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_StartApp::CommandHeader  
Command header.  
Definition at line 137 of file default\_cfe\_es\_msgstruct.h.

**11.168.2.2 Payload** [CFE\\_ES\\_StartAppCmd\\_Payload\\_t](#) CFE\_ES\_StartApp::Payload  
Command payload.  
Definition at line 138 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

## 11.169 CFE\_ES\_StartAppCmd\_Payload Struct Reference

Start Application Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- char [Application](#) [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]  
*Name of Application to be started.*
- char [AppEntryPoint](#) [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]  
*Symbolic name of Application's entry point.*
- char [AppFileName](#) [[CFE\\_MISSION\\_MAX\\_PATH\\_LEN](#)]  
*Full path and filename of Application's executable image.*
- [CFE\\_ES\\_MemOffset\\_t](#) [StackSize](#)  
*Desired stack size for the new application.*
- [CFE\\_ES\\_ExceptionAction\\_Enum\\_t](#) [ExceptionAction](#)  
*CFE\_ES\_ExceptionAction\_RESTART\_APP=On exception, restart Application, CFE\_ES\_ExceptionAction\_PROC\_RESTART=On exception, perform a Processor Reset*
- [CFE\\_ES\\_TaskPriority\\_Atom\\_t](#) [Priority](#)  
*The new Applications runtime priority.*

### 11.169.1 Detailed Description

Start Application Command Payload.

For command details, see [CFE\\_ES\\_START\\_APP\\_CC](#)

Definition at line 82 of file default\_cfe\_es\_msgdefs.h.

### 11.169.2 Field Documentation

**11.169.2.1 AppEntryPoint** char CFE\_ES\_StartAppCmd\_Payload::AppEntryPoint [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]  
Symbolic name of Application's entry point.  
Definition at line 85 of file default\_cfe\_es\_msgdefs.h.

**11.169.2.2 AppFileName** char CFE\_ES\_StartAppCmd\_Payload::AppFileName [[CFE\\_MISSION\\_MAX\\_PATH\\_LEN](#)]  
Full path and filename of Application's executable image.  
Definition at line 86 of file default\_cfe\_es\_msgdefs.h.

**11.169.2.3 Application** `char CFE_ES_StartAppCmd_Payload::Application[CFE_MISSION_MAX_API_LEN]`

Name of Application to be started.

Definition at line 84 of file default\_cfe\_es\_msgdefs.h.

**11.169.2.4 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 91 of file default\_cfe\_es\_msgdefs.h.

**11.169.2.5 Priority** `CFE_ES_TaskPriority_Atom_t CFE_ES_StartAppCmd_Payload::Priority`

The new Applications runtime priority.

Definition at line 95 of file default\_cfe\_es\_msgdefs.h.

**11.169.2.6 StackSize** `CFE_ES_MemOffset_t CFE_ES_StartAppCmd_Payload::StackSize`

Desired stack size for the new application.

Definition at line 89 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.170 CFE\_ES\_StartPerfCmd\_Payload Struct Reference

Start Performance Analyzer Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- `CFE_ES_PerfMode_Enum_t TriggerMode`

*Desired trigger position (Start, Center, End). Values defined by CFE\_ES\_PerfMode.*

### 11.170.1 Detailed Description

Start Performance Analyzer Command Payload.

For command details, see `CFE_ES_START_PERF_DATA_CC`

Definition at line 165 of file default\_cfe\_es\_msgdefs.h.

### 11.170.2 Field Documentation

**11.170.2.1 TriggerMode** `CFE_ES_PerfMode_Enum_t CFE_ES_StartPerfCmd_Payload::TriggerMode`

Desired trigger position (Start, Center, End). Values defined by `CFE_ES_PerfMode`.

Definition at line 168 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.171 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.*

**11.171.1 Detailed Description**

Start Performance Analyzer Command.

Definition at line 194 of file default\_cfe\_es\_msgstruct.h.

**11.171.2 Field Documentation****11.171.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_StartPerfDataCmd::CommandHeader  
Command header.

Definition at line 196 of file default\_cfe\_es\_msgstruct.h.

**11.171.2.2 Payload** [CFE\\_ES\\_StartPerfCmd\\_Payload\\_t](#) CFE\_ES\_StartPerfDataCmd::Payload  
Command payload.

Definition at line 197 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

**11.172 CFE\_ES\_StopAppCmd Struct Reference**

```
#include <default_cfe_es_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_ES\\_AppNameCmd\\_Payload\\_t](#) Payload  
*Command payload.*

**11.172.1 Detailed Description**

Definition at line 146 of file default\_cfe\_es\_msgstruct.h.

**11.172.2 Field Documentation****11.172.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_StopAppCmd::CommandHeader  
Command header.

Definition at line 148 of file default\_cfe\_es\_msgstruct.h.

**11.172.2.2 Payload** `CFE_ES_AppNameCmd_Payload_t` `CFE_ES_StopAppCmd::Payload`  
Command payload.

Definition at line 149 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

## 11.173 CFE\_ES\_StopPerfCmd\_Payload Struct Reference

Stop Performance Analyzer Command Payload.

```
#include <default_cfe_es_msgdefs.h>
```

### Data Fields

- char `DataFileName [CFE_MISSION_MAX_PATH_LEN]`  
*ASCII text string of full path and filename of file Performance Analyzer data is to be written.*

### 11.173.1 Detailed Description

Stop Performance Analyzer Command Payload.

For command details, see [CFE\\_ES\\_STOP\\_PERF\\_DATA\\_CC](#)

Definition at line 177 of file default\_cfe\_es\_msgdefs.h.

### 11.173.2 Field Documentation

**11.173.2.1 DataFileName** `char CFE_ES_StopPerfCmd_Payload::DataFileName [CFE_MISSION_MAX_PATH_LEN]`  
ASCII text string of full path and filename of file Performance Analyzer data is to be written.

Definition at line 179 of file default\_cfe\_es\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgdefs.h

## 11.174 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.*

### 11.174.1 Detailed Description

Stop Performance Analyzer Command.

Definition at line 203 of file default\_cfe\_es\_msgstruct.h.

### 11.174.2 Field Documentation

**11.174.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_ES\_StopPerfDataCmd::CommandHeader  
Command header.  
Definition at line 205 of file default\_cfe\_es\_msgstruct.h.

**11.174.2.2 Payload** [CFE\\_ES\\_StopPerfCmd\\_Payload\\_t](#) CFE\_ES\_StopPerfDataCmd::Payload  
Command payload.  
Definition at line 206 of file default\_cfe\_es\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/es/config/default\_cfe\_es\_msgstruct.h

## 11.175 CFE\_ES\_TaskInfo Struct Reference

Task Information.

```
#include <default_cfe_es_extern_typedefs.h>
```

### Data Fields

- [CFE\\_ES\\_TaskId\\_t TaskId](#)  
*Task Id.*
- [uint32 ExecutionCounter](#)  
*Task Execution Counter.*
- [char TaskName \[CFE\\_MISSION\\_MAX\\_API\\_LEN\]](#)  
*Task Name.*
- [CFE\\_ES\\_AppId\\_t AppId](#)  
*Parent Application ID.*
- [charAppName \[CFE\\_MISSION\\_MAX\\_API\\_LEN\]](#)  
*Parent Application Name.*
- [CFE\\_ES\\_MemOffset\\_t StackSize](#)
- [CFE\\_ES\\_TaskPriority\\_Atom\\_t Priority](#)
- [uint8 Spare \[2\]](#)

### 11.175.1 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 434 of file default\_cfe\_es\_extern\_typedefs.h.

### 11.175.2 Field Documentation

**11.175.2.1 AppId** [CFE\\_ES\\_AppId\\_t](#) CFE\_ES\_TaskInfo::AppId  
Parent Application ID.  
Definition at line 439 of file default\_cfe\_es\_extern\_typedefs.h.

**11.175.2.2 AppName** `char CFE_ES_TaskInfo::AppName[CFE_MISSION_MAX_API_LEN]`  
Parent Application Name.  
Definition at line 440 of file default\_cfe\_es\_extern\_typedefs.h.

**11.175.2.3 ExecutionCounter** `uint32 CFE_ES_TaskInfo::ExecutionCounter`  
Task Execution Counter.  
Definition at line 437 of file default\_cfe\_es\_extern\_typedefs.h.

**11.175.2.4 Priority** `CFE_ES_TaskPriority_Atom_t CFE_ES_TaskInfo::Priority`  
Priority of task  
Definition at line 442 of file default\_cfe\_es\_extern\_typedefs.h.

**11.175.2.5 Spare** `uint8 CFE_ES_TaskInfo::Spare[2]`  
Spare bytes for alignment  
Definition at line 443 of file default\_cfe\_es\_extern\_typedefs.h.

**11.175.2.6 StackSize** `CFE_ES_MemOffset_t CFE_ES_TaskInfo::StackSize`  
Size of task stack  
Definition at line 441 of file default\_cfe\_es\_extern\_typedefs.h.

**11.175.2.7 TaskId** `CFE_ES_TaskId_t CFE_ES_TaskInfo::TaskId`  
Task Id.  
Definition at line 436 of file default\_cfe\_es\_extern\_typedefs.h.

**11.175.2.8 TaskName** `char CFE_ES_TaskInfo::TaskName[CFE_MISSION_MAX_API_LEN]`  
Task Name.  
Definition at line 438 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

## 11.176 CFE\_ES\_WriteERLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t CommandHeader**  
*Command header.*
- **CFE\_ES\_FileNameCmd\_Payload\_t Payload**  
*Command payload.*

### 11.176.1 Detailed Description

Definition at line 117 of file default\_cfe\_es\_msgstruct.h.

## 11.176.2 Field Documentation

**11.176.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_WriteERLogCmd::CommandHeader`  
Command header.

Definition at line 119 of file default\_cfe\_es\_msgstruct.h.

**11.176.2.2 Payload** `CFE_ES_FileNameCmd_Payload_t` `CFE_ES_WriteERLogCmd::Payload`  
Command payload.

Definition at line 120 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

## 11.177 CFE\_ES\_WriteSysLogCmd Struct Reference

```
#include <default_cfe_es_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_ES_FileNameCmd_Payload_t Payload`  
*Command payload.*

### 11.177.1 Detailed Description

Definition at line 111 of file default\_cfe\_es\_msgstruct.h.

### 11.177.2 Field Documentation

**11.177.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_ES_WriteSysLogCmd::CommandHeader`  
Command header.

Definition at line 113 of file default\_cfe\_es\_msgstruct.h.

**11.177.2.2 Payload** `CFE_ES_FileNameCmd_Payload_t` `CFE_ES_WriteSysLogCmd::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

## 11.178 CFE\_EVS\_AddEventFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload\\_t](#) Payload  
*Command payload.*

**11.178.1 Detailed Description**

Definition at line 195 of file default\_cfe\_evs\_msgstruct.h.

**11.178.2 Field Documentation****11.178.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_AddEventFilterCmd::CommandHeader**  
Command header.

Definition at line 197 of file default\_cfe\_evs\_msgstruct.h.

**11.178.2.2 Payload [CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload\\_t](#) CFE\_EVS\_AddEventFilterCmd::Payload**  
Command payload.

Definition at line 198 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

**11.179 CFE\_EVS\_AppDataCmd\_Payload Struct Reference**

Write Event Services Application Information to File Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

**Data Fields**

- char [AppDataFilename \[CFE\\_MISSION\\_MAX\\_PATH\\_LEN\]](#)  
*Filename where application data is to be written.*

**11.179.1 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 66 of file default\_cfe\_evs\_msgdefs.h.

**11.179.2 Field Documentation****11.179.2.1 AppDataFilename char CFE\_EVS\_AppDataCmd\_Payload::AppDataFilename [CFE\_MISSION\_MAX\_PATH\_LEN]**  
Filename where application data is to be written.

Definition at line 68 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.180 CFE\_EVS\_AppNameBitMaskCmd\_Payload Struct Reference

Generic App Name and Bitmask Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- char **AppName** [CFE\_MISSION\_MAX\_API\_LEN]  
*Application name to use in the command.*
- uint8 **BitMask**  
*BitMask to use in the command.*
- uint8 **Spare**  
*Pad to even byte.*

### 11.180.1 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 138 of file default\_cfe\_evs\_msgdefs.h.

### 11.180.2 Field Documentation

#### 11.180.2.1 **AppName** char CFE\_EVS\_AppNameBitMaskCmd\_Payload::AppName [CFE\_MISSION\_MAX\_API\_LEN]

Application name to use in the command.

Definition at line 140 of file default\_cfe\_evs\_msgdefs.h.

#### 11.180.2.2 **BitMask** uint8 CFE\_EVS\_AppNameBitMaskCmd\_Payload::BitMask

BitMask to use in the command.

Definition at line 141 of file default\_cfe\_evs\_msgdefs.h.

#### 11.180.2.3 **Spare** uint8 CFE\_EVS\_AppNameBitMaskCmd\_Payload::Spare

Pad to even byte.

Definition at line 142 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.181 CFE\_EVS\_AppNameCmd\_Payload Struct Reference

Generic App Name Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- char **AppName** [CFE\_MISSION\_MAX\_API\_LEN]  
*Application name to use in the command.*

### 11.181.1 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 115 of file default\_cfe\_evs\_msgdefs.h.

### 11.181.2 Field Documentation

#### 11.181.2.1 AppName `char CFE_EVS_AppNameCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]`

Application name to use in the command.

Definition at line 117 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.182 CFE\_EVS\_AppNameEventIDCmd\_Payload Struct Reference

Generic App Name and Event ID Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- `char AppName [CFE_MISSION_MAX_API_LEN]`  
*Application name to use in the command.*
- `uint16 EventID`  
*Event ID to use in the command.*

### 11.182.1 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 126 of file default\_cfe\_evs\_msgdefs.h.

### 11.182.2 Field Documentation

#### 11.182.2.1 AppName `char CFE_EVS_AppNameEventIDCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]`

Application name to use in the command.

Definition at line 128 of file default\_cfe\_evs\_msgdefs.h.

#### 11.182.2.2 EventID `uint16 CFE_EVS_AppNameEventIDCmd_Payload::EventID`

Event ID to use in the command.

Definition at line 129 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.183 CFE\_EVS\_AppNameEventIDMaskCmd\_Payload Struct Reference

Generic App Name, Event ID, Mask Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

## Data Fields

- `charAppName [CFE_MISSION_MAX_API_LEN]`  
*Application name to use in the command.*
- `uint16EventID`  
*Event ID to use in the command.*
- `uint16Mask`  
*Mask to use in the command.*

### 11.183.1 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 152 of file `default_cfe_evs_msgdefs.h`.

### 11.183.2 Field Documentation

#### 11.183.2.1 `AppName` `char CFE_EVS_AppNameEventIDMaskCmd_Payload::AppName [CFE_MISSION_MAX_API_LEN]`

Application name to use in the command.

Definition at line 154 of file `default_cfe_evs_msgdefs.h`.

#### 11.183.2.2 `EventID` `uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload::EventID`

Event ID to use in the command.

Definition at line 155 of file `default_cfe_evs_msgdefs.h`.

#### 11.183.2.3 `Mask` `uint16 CFE_EVS_AppNameEventIDMaskCmd_Payload::Mask`

Mask to use in the command.

Definition at line 156 of file `default_cfe_evs_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgdefs.h`

## 11.184 CFE\_EVS\_AppTlmData Struct Reference

```
#include <default_cfe_evs_msgdefs.h>
```

## Data Fields

- `CFE_ES_AppId_t AppID`  
*Numerical application identifier.*
- `uint16AppMessageSentCounter`  
*Application message sent counter.*
- `uint8AppEnableStatus`  
*Application event service enable status.*
- `uint8AppMessageSquelchedCounter`  
*Number of events squelched.*

### 11.184.1 Detailed Description

Definition at line 165 of file default\_cfe\_evs\_msgdefs.h.

### 11.184.2 Field Documentation

#### 11.184.2.1 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 171 of file default\_cfe\_evs\_msgdefs.h.

#### 11.184.2.2 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 167 of file default\_cfe\_evs\_msgdefs.h.

#### 11.184.2.3 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 169 of file default\_cfe\_evs\_msgdefs.h.

#### 11.184.2.4 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 173 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.185 CFE\_EVS\_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.*

### 11.185.1 Detailed Description

Event message filter definition structure.

Definition at line 60 of file `cfe_evs_api_typedefs.h`.

### 11.185.2 Field Documentation

#### 11.185.2.1 EventID `uint16` `CFE_EVS_BinFilter::EventID`

Numerical event identifier.

Definition at line 62 of file `cfe_evs_api_typedefs.h`.

#### 11.185.2.2 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`

## 11.186 CFE\_EVS\_BitMaskCmd\_Payload Struct Reference

Generic Bitmask Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- `uint8 BitMask`  
*BitMask to use in the command.*
- `uint8 Spare`  
*Pad to even byte.*

### 11.186.1 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 102 of file `default_cfe_evs_msgdefs.h`.

### 11.186.2 Field Documentation

#### 11.186.2.1 BitMask `uint8` `CFE_EVS_BitMaskCmd_Payload::BitMask`

BitMask to use in the command.

Definition at line 104 of file `default_cfe_evs_msgdefs.h`.

#### 11.186.2.2 Spare `uint8` `CFE_EVS_BitMaskCmd_Payload::Spare`

Pad to even byte.

Definition at line 105 of file `default_cfe_evs_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgdefs.h`

## 11.187 CFE\_EVS\_ClearLogCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)

*Command header.*

#### 11.187.1 Detailed Description

Definition at line 52 of file default\_cfe\_evs\_msgstruct.h.

#### 11.187.2 Field Documentation

##### 11.187.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_ClearLogCmd::CommandHeader

Command header.

Definition at line 54 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/[default\\_cfe\\_evs\\_msgstruct.h](#)

## 11.188 CFE\_EVS\_DeleteEventFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)

*Command header.*

- [CFE\\_EVS\\_AppNameEventIDCmd\\_Payload\\_t Payload](#)

*Command payload.*

#### 11.188.1 Detailed Description

Definition at line 167 of file default\_cfe\_evs\_msgstruct.h.

#### 11.188.2 Field Documentation

##### 11.188.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_DeleteEventFilterCmd::CommandHeader

Command header.

Definition at line 169 of file default\_cfe\_evs\_msgstruct.h.

##### 11.188.2.2 Payload [CFE\\_EVS\\_AppNameEventIDCmd\\_Payload\\_t](#) CFE\_EVS\_DeleteEventFilterCmd::Payload

Command payload.

Definition at line 170 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](#)

## 11.189 CFE\_EVS\_DisableAppEventsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_AppNameCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.189.1 Detailed Description

Definition at line 138 of file default\_cfe\_evs\_msgstruct.h.

### 11.189.2 Field Documentation

#### 11.189.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_DisableAppEventsCmd::CommandHeader

Command header.

Definition at line 140 of file default\_cfe\_evs\_msgstruct.h.

#### 11.189.2.2 Payload [CFE\\_EVS\\_AppNameCmd\\_Payload\\_t](#) CFE\_EVS\_DisableAppEventsCmd::Payload

Command payload.

Definition at line 141 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

## 11.190 CFE\_EVS\_DisableEventTypeCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.190.1 Detailed Description

Definition at line 184 of file default\_cfe\_evs\_msgstruct.h.

### 11.190.2 Field Documentation

#### 11.190.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_DisableEventTypeCmd::CommandHeader

Command header.

Definition at line 186 of file default\_cfe\_evs\_msgstruct.h.

**11.190.2.2 Payload** `CFE_EVS_AppNameBitMaskCmd_Payload_t` `CFE_EVS_DisableAppEventTypeCmd::Payload`  
Command payload.

Definition at line 187 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.191 CFE\_EVS\_DisableEventTypeCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_EVS_BitMaskCmd_Payload_t Payload`  
*Command payload.*

### 11.191.1 Detailed Description

Definition at line 121 of file default\_cfe\_evs\_msgstruct.h.

### 11.191.2 Field Documentation

**11.191.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_EVS_DisableEventTypeCmd::CommandHeader`  
Command header.

Definition at line 123 of file default\_cfe\_evs\_msgstruct.h.

**11.191.2.2 Payload** `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_DisableEventTypeCmd::Payload`  
Command payload.

Definition at line 124 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.192 CFE\_EVS\_DisablePortsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_EVS_BitMaskCmd_Payload_t Payload`  
*Command payload.*

### 11.192.1 Detailed Description

Definition at line 109 of file default\_cfe\_evs\_msgstruct.h.

### 11.192.2 Field Documentation

**11.192.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_EVS_DisablePortsCmd::CommandHeader`  
Command header.

Definition at line 111 of file `default_cfe_evs_msgstruct.h`.

**11.192.2.2 Payload** `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_DisablePortsCmd::Payload`  
Command payload.

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`

## 11.193 CFE\_EVS\_EnableAppEventsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_EVS_AppNameCmd_Payload_t Payload`  
*Command payload.*

### 11.193.1 Detailed Description

Definition at line 132 of file `default_cfe_evs_msgstruct.h`.

### 11.193.2 Field Documentation

**11.193.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_EVS_EnableAppEventsCmd::CommandHeader`  
Command header.

Definition at line 134 of file `default_cfe_evs_msgstruct.h`.

**11.193.2.2 Payload** `CFE_EVS_AppNameCmd_Payload_t` `CFE_EVS_EnableAppEventsCmd::Payload`  
Command payload.

Definition at line 135 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.194 CFE\_EVS\_EnableEventTypeCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload\\_t](#) Payload  
*Command payload.*

**11.194.1 Detailed Description**

Definition at line 178 of file default\_cfe\_evs\_msgstruct.h.

**11.194.2 Field Documentation****11.194.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_EnableAppEventTypeCmd::CommandHeader  
Command header.

Definition at line 180 of file default\_cfe\_evs\_msgstruct.h.

**11.194.2.2 Payload** [CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload\\_t](#) CFE\_EVS\_EnableAppEventTypeCmd::Payload  
Command payload.

Definition at line 181 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

**11.195 CFE\_EVS\_EnableEventTypeCmd Struct Reference**

```
#include <default_cfe_evs_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_BitMaskCmd\\_Payload\\_t](#) Payload  
*Command payload.*

**11.195.1 Detailed Description**

Definition at line 115 of file default\_cfe\_evs\_msgstruct.h.

**11.195.2 Field Documentation****11.195.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_EnableEventTypeCmd::CommandHeader  
Command header.

Definition at line 117 of file default\_cfe\_evs\_msgstruct.h.

**11.195.2.2 Payload** `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_EnableEventTypeCmd::Payload`  
Command payload.

Definition at line 118 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.196 CFE\_EVS\_EnablePortsCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_EVS_BitMaskCmd_Payload_t Payload`  
*Command payload.*

### 11.196.1 Detailed Description

Definition at line 103 of file default\_cfe\_evs\_msgstruct.h.

### 11.196.2 Field Documentation

**11.196.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_EVS_EnablePortsCmd::CommandHeader`  
Command header.

Definition at line 105 of file default\_cfe\_evs\_msgstruct.h.

**11.196.2.2 Payload** `CFE_EVS_BitMaskCmd_Payload_t` `CFE_EVS_EnablePortsCmd::Payload`  
Command payload.

Definition at line 106 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.197 CFE\_EVS\_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.*

### 11.197.1 Detailed Description

Definition at line 213 of file default\_cfe\_evs\_msgstruct.h.

## 11.197.2 Field Documentation

**11.197.2.1 Payload** `CFE_EVS_HousekeepingTlm_Payload_t` `CFE_EVS_HousekeepingTlm::Payload`  
Telemetry payload.

Definition at line 216 of file `default_cfe_evs_msgstruct.h`.

**11.197.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_EVS_HousekeepingTlm::TelemetryHeader`  
Telemetry header.

Definition at line 215 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.198 CFE\_EVS\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- `uint8 CommandCounter`  
*EVS Command Counter.*
- `uint8 CommandErrorCounter`  
*EVS Command Error Counter.*
- `uint8 MessageFormatMode`  
*Event message format mode (short/long)*
- `uint8 MessageTruncCounter`  
*Event message truncation counter.*
- `uint8 UnregisteredAppCounter`  
*Unregistered application message send counter.*
- `uint8 OutputPort`  
*Output port mask.*
- `uint8 LogFullFlag`  
*Local event log full flag.*
- `uint8 LogMode`  
*Local event logging mode (overwrite/discard)*
- `uint16 MessageSendCounter`  
*Event message send counter.*
- `uint16 LogOverflowCounter`  
*Local event log overflow counter.*
- `uint8 LogEnabled`  
*Current event log enable/disable state.*
- `uint8 Spare1`  
*Padding for 32 bit boundary.*
- `uint8 Spare2`  
*Padding for 32 bit boundary.*
- `uint8 Spare3`  
*Padding for 32 bit boundary.*
- `CFE_EVS_AppTlmData_t AppData [CFE_MISSION_ES_MAX_APPLICATIONS]`  
*Array of registered application table data.*

### 11.198.1 Detailed Description

**Name** Event Services Housekeeping Telemetry Packet

Definition at line 180 of file default\_cfe\_evs\_msgdefs.h.

### 11.198.2 Field Documentation

**11.198.2.1 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 214 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.2 CommandCounter** `uint8 CFE_EVS_HousekeepingTlm_Payload::CommandCounter`  
EVS Command Counter.

**Telemetry Mnemonic(s)** `$sc_$cpu_EVS_CMDPC`

Definition at line 182 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.3 CommandErrorCounter** `uint8 CFE_EVS_HousekeepingTlm_Payload::CommandErrorCounter`  
EVS Command Error Counter.

**Telemetry Mnemonic(s)** `$sc_$cpu_EVS_CMDEC`

Definition at line 184 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.4 LogEnabled** `uint8 CFE_EVS_HousekeepingTlm_Payload::LogEnabled`  
Current event log enable/disable state.

**Telemetry Mnemonic(s)** `$sc_$cpu_EVS_LOGENABLED`

Definition at line 205 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.5 LogFullFlag** `uint8 CFE_EVS_HousekeepingTlm_Payload::LogFullFlag`  
Local event log full flag.

**Telemetry Mnemonic(s)** `$sc_$cpu_EVS_LOGFULL`

Definition at line 195 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.6 LogMode** `uint8 CFE_EVS_HousekeepingTlm_Payload::LogMode`  
Local event logging mode (overwrite/discard)

**Telemetry Mnemonic(s)** `$sc_$cpu_EVS_LOGMODE`

Definition at line 197 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.7 LogOverflowCounter** `uint16` CFE\_EVS\_HousekeepingTlm\_Payload::LogOverflowCounter  
Local event log overflow counter.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_LOGOVERFLOWC

Definition at line 202 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.8 MessageFormatMode** `uint8` CFE\_EVS\_HousekeepingTlm\_Payload::MessageFormatMode  
Event message format mode (short/long)

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_MSGFMTMODE

Definition at line 186 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.9 MessageSendCounter** `uint16` CFE\_EVS\_HousekeepingTlm\_Payload::MessageSendCounter  
Event message send counter.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_MSGSENTC

Definition at line 200 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.10 MessageTruncCounter** `uint8` CFE\_EVS\_HousekeepingTlm\_Payload::MessageTruncCounter  
Event message truncation counter.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_MSGTRUNC

Definition at line 188 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.11 OutputPort** `uint8` CFE\_EVS\_HousekeepingTlm\_Payload::OutputPort  
Output port mask.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_OUTPUTPORT

Definition at line 193 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.12 Spare1** `uint8` CFE\_EVS\_HousekeepingTlm\_Payload::Spare1  
Padding for 32 bit boundary.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_HK\_SPARE1

Definition at line 207 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.13 Spare2** `uint8` CFE\_EVS\_HousekeepingTlm\_Payload::Spare2  
Padding for 32 bit boundary.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_HK\_SPARE2

Definition at line 209 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.14 Spare3** `uint8 CFE_EVS_HousekeepingTlm_Payload::Spare3`  
Padding for 32 bit boundary.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_HK\_SPARE3

Definition at line 211 of file default\_cfe\_evs\_msgdefs.h.

**11.198.2.15 UnregisteredAppCounter** `uint8 CFE_EVS_HousekeepingTlm_Payload::UnregisteredAppCounter`  
Unregistered application message send counter.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_UNREGAPPC

Definition at line 191 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.199 CFE\_EVS\_LogFileCmd\_Payload Struct Reference

Write Event Log to File Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- char **LogFileFilename** [`CFE_MISSION_MAX_PATH_LEN`]  
*Filename where log data is to be written.*

#### 11.199.1 Detailed Description

Write Event Log to File Command Payload.

For command details, see [CFE\\_EVS\\_WRITE\\_LOG\\_DATA\\_FILE\\_CC](#)

Definition at line 55 of file default\_cfe\_evs\_msgdefs.h.

#### 11.199.2 Field Documentation

**11.199.2.1 LogFilename** `char CFE_EVS_LogFileCmd_Payload::LogFileFilename[CFE_MISSION_MAX_PATH_LEN]`  
Filename where log data is to be written.

Definition at line 57 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.200 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.*

### 11.200.1 Detailed Description

Definition at line 219 of file default\_cfe\_evs\_msgstruct.h.

### 11.200.2 Field Documentation

#### 11.200.2.1 Payload `CFE_EVS_LongEventTlm_Payload_t` `CFE_EVS_LongEventTlm::Payload`

Telemetry payload.

Definition at line 222 of file default\_cfe\_evs\_msgstruct.h.

#### 11.200.2.2 TelemetryHeader `CFE_MSG_TelemetryHeader_t` `CFE_EVS_LongEventTlm::TelemetryHeader`

Telemetry header.

Definition at line 221 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.201 CFE\_EVS\_LongEventTlm\_Payload Struct Reference

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- `CFE_EVS_PacketID_t PacketID`  
*Event packet information.*
- char `Message [CFE_MISSION_EVS_MAX_MESSAGE_LENGTH]`  
*Event message string.*
- `uint8 Spare1`  
*Structure padding.*
- `uint8 Spare2`  
*Structure padding.*

### 11.201.1 Detailed Description

**Name** Event Message Telemetry Packet (Long format)

Definition at line 237 of file default\_cfe\_evs\_msgdefs.h.

### 11.201.2 Field Documentation

#### 11.201.2.1 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 240 of file default\_cfe\_evs\_msgdefs.h.

**11.201.2.2 PacketID** `CFE_EVS_PacketID_t CFE_EVS_LongEventTlm_Payload::PacketID`  
Event packet information.  
Definition at line 239 of file default\_cfe\_evs\_msgdefs.h.

**11.201.2.3 Spare1** `uint8 CFE_EVS_LongEventTlm_Payload::Spare1`  
Structure padding.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_SPARE1

Definition at line 242 of file default\_cfe\_evs\_msgdefs.h.

**11.201.2.4 Spare2** `uint8 CFE_EVS_LongEventTlm_Payload::Spare2`  
Structure padding.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_SPARE2

Definition at line 244 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.202 CFE\_EVS\_NoopCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t CommandHeader**  
*Command header.*

### 11.202.1 Detailed Description

Definition at line 42 of file default\_cfe\_evs\_msgstruct.h.

### 11.202.2 Field Documentation

**11.202.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_EVS_NoopCmd::CommandHeader`  
Command header.  
Definition at line 44 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

## 11.203 CFE\_EVS\_PacketID Struct Reference

```
#include <default_cfe_evs_msgdefs.h>
```

## Data Fields

- char `AppName [CFE_MISSION_MAX_API_LEN]`  
*Application name.*
- `uint16 EventID`  
*Numerical event identifier.*
- `CFE_EVS_EventType_Enum_t EventType`  
*Numerical event type identifier.*
- `uint32 SpacecraftID`  
*Spacecraft identifier.*
- `uint32 ProcessorID`  
*Numerical processor identifier.*

### 11.203.1 Detailed Description

Telemetry packet structures

Definition at line 220 of file default\_cfe\_evs\_msgdefs.h.

### 11.203.2 Field Documentation

#### 11.203.2.1 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 222 of file default\_cfe\_evs\_msgdefs.h.

#### 11.203.2.2 EventID `uint16 CFE_EVS_PacketID::EventID`

Numerical event identifier.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_EVENTID

Definition at line 224 of file default\_cfe\_evs\_msgdefs.h.

#### 11.203.2.3 EventType `CFE_EVS_EventType_Enum_t CFE_EVS_PacketID::EventType`

Numerical event type identifier.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_EVENTTYPE

Definition at line 226 of file default\_cfe\_evs\_msgdefs.h.

#### 11.203.2.4 ProcessorID `uint32 CFE_EVS_PacketID::ProcessorID`

Numerical processor identifier.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_PROCESSORID

Definition at line 230 of file default\_cfe\_evs\_msgdefs.h.

**11.203.2.5 SpacecraftID** `uint32 CFE_EVS_PacketID::SpacecraftID`  
Spacecraft identifier.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_EVS\_SCID

Definition at line 228 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.204 CFE\_EVS\_ResetAllFiltersCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t CommandHeader**  
*Command header.*
- **CFE\_EVS\_AppNameCmd\_Payload\_t Payload**  
*Command payload.*

#### 11.204.1 Detailed Description

Definition at line 150 of file default\_cfe\_evs\_msgstruct.h.

#### 11.204.2 Field Documentation

**11.204.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_EVS_ResetAllFiltersCmd::CommandHeader`  
Command header.

Definition at line 152 of file default\_cfe\_evs\_msgstruct.h.

**11.204.2.2 Payload** `CFE_EVS_AppNameCmd_Payload_t CFE_EVS_ResetAllFiltersCmd::Payload`  
Command payload.

Definition at line 153 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

## 11.205 CFE\_EVS\_ResetAppCounterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- **CFE\_MSG\_CommandHeader\_t CommandHeader**  
*Command header.*
- **CFE\_EVS\_AppNameCmd\_Payload\_t Payload**  
*Command payload.*

#### 11.205.1 Detailed Description

Definition at line 144 of file default\_cfe\_evs\_msgstruct.h.

## 11.205.2 Field Documentation

**11.205.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_ResetAppCounterCmd::CommandHeader  
Command header.

Definition at line 146 of file default\_cfe\_evs\_msgstruct.h.

**11.205.2.2 Payload** [CFE\\_EVS\\_AppNameCmd\\_Payload\\_t](#) CFE\_EVS\_ResetAppCounterCmd::Payload  
Command payload.

Definition at line 147 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.206 CFE\_EVS\_ResetCountersCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*

### 11.206.1 Detailed Description

Definition at line 47 of file default\_cfe\_evs\_msgstruct.h.

## 11.206.2 Field Documentation

**11.206.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_ResetCountersCmd::CommandHeader  
Command header.

Definition at line 49 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

## 11.207 CFE\_EVS\_ResetFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_AppNameEventIDCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.207.1 Detailed Description

Definition at line 161 of file default\_cfe\_evs\_msgstruct.h.

## 11.207.2 Field Documentation

**11.207.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_EVS_ResetFilterCmd::CommandHeader`  
Command header.

Definition at line 163 of file `default_cfe_evs_msgstruct.h`.

**11.207.2.2 Payload** `CFE_EVS_AppNameEventIDCmd_Payload_t` `CFE_EVS_ResetFilterCmd::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`

## 11.208 CFE\_EVS\_SendHkCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

### 11.208.1 Detailed Description

Definition at line 57 of file `default_cfe_evs_msgstruct.h`.

## 11.208.2 Field Documentation

**11.208.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_EVS_SendHkCmd::CommandHeader`  
Command header.

Definition at line 59 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.209 CFE\_EVS\_SetEventFormatCode\_Payload Struct Reference

Set Event Format Mode Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- `CFE_EVS_MsgFormat_Enum_t MsgFormat`  
*Mode to use in the command.*
- `uint8 Spare`  
*Pad to even byte.*

### 11.209.1 Detailed Description

Set Event Format Mode Command Payload.

For command details, see [CFE\\_EVS\\_SET\\_EVENT\\_FORMAT\\_MODE\\_CC](#)

Definition at line 89 of file default\_cfe\_evs\_msgdefs.h.

### 11.209.2 Field Documentation

**11.209.2.1 MsgFormat** [CFE\\_EVS\\_MsgFormat\\_Enum\\_t](#) CFE\_EVS\_SetEventFormatCode\_Payload::MsgFormat  
Mode to use in the command.

Definition at line 91 of file default\_cfe\_evs\_msgdefs.h.

**11.209.2.2 Spare** [uint8](#) CFE\_EVS\_SetEventFormatCode\_Payload::Spare  
Pad to even byte.

Definition at line 92 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.210 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.*

### 11.210.1 Detailed Description

Set Event Format Mode Command.

Definition at line 92 of file default\_cfe\_evs\_msgstruct.h.

### 11.210.2 Field Documentation

**11.210.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_SetEventFormatModeCmd::CommandHeader  
Command header.

Definition at line 94 of file default\_cfe\_evs\_msgstruct.h.

**11.210.2.2 Payload** [CFE\\_EVS\\_SetEventFormatMode\\_Payload\\_t](#) CFE\_EVS\_SetEventFormatModeCmd::Payload  
Command payload.

Definition at line 95 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.211 CFE\_EVS\_SetFilterCmd Struct Reference

```
#include <default_cfe_evs_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.211.1 Detailed Description

Definition at line 201 of file default\_cfe\_evs\_msgstruct.h.

### 11.211.2 Field Documentation

#### 11.211.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_SetFilterCmd::CommandHeader

Command header.

Definition at line 203 of file default\_cfe\_evs\_msgstruct.h.

#### 11.211.2.2 Payload [CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload\\_t](#) CFE\_EVS\_SetFilterCmd::Payload

Command payload.

Definition at line 204 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.212 CFE\_EVS\_SetLogMode\_Payload Struct Reference

Set Log Mode Command Payload.

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- [CFE\\_EVS\\_LogMode\\_Enum\\_t](#) LogMode  
*Mode to use in the command.*
- [uint8](#) Spare  
*Pad to even byte.*

### 11.212.1 Detailed Description

Set Log Mode Command Payload.

For command details, see [CFE\\_EVS\\_SET\\_LOG\\_MODE\\_CC](#)

Definition at line 77 of file default\_cfe\_evs\_msgdefs.h.

### 11.212.2 Field Documentation

**11.212.2.1 LogMode** [CFE\\_EVS\\_LogMode\\_Enum\\_t](#) CFE\_EVS\_SetLogMode\_Payload::LogMode  
Mode to use in the command.

Definition at line 79 of file default\_cfe\_evs\_msgdefs.h.

**11.212.2 Spare** [uint8](#) CFE\_EVS\_SetLogMode\_Payload::Spare

Pad to even byte.

Definition at line 80 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.213 CFE\_EVS\_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.*

### 11.213.1 Detailed Description

Set Log Mode Command.

Definition at line 83 of file default\_cfe\_evs\_msgstruct.h.

### 11.213.2 Field Documentation

**11.213.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_SetLogModeCmd::CommandHeader  
Command header.

Definition at line 85 of file default\_cfe\_evs\_msgstruct.h.

**11.213.2.2 Payload** [CFE\\_EVS\\_SetLogMode\\_Payload\\_t](#) CFE\_EVS\_SetLogModeCmd::Payload  
Command payload.

Definition at line 86 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.214 CFE\_EVS\_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.*

### 11.214.1 Detailed Description

Definition at line 225 of file default\_cfe\_evs\_msgstruct.h.

### 11.214.2 Field Documentation

**11.214.2.1 Payload** `CFE_EVS_ShortEventTlm_Payload_t` `CFE_EVS_ShortEventTlm::Payload`  
Telemetry payload.

Definition at line 228 of file default\_cfe\_evs\_msgstruct.h.

**11.214.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_EVS_ShortEventTlm::TelemetryHeader`  
Telemetry header.

Definition at line 227 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

## 11.215 CFE\_EVS\_ShortEventTlm\_Payload Struct Reference

```
#include <default_cfe_evs_msgdefs.h>
```

### Data Fields

- `CFE_EVS_PacketID_t PacketID`

*Event packet information.*

### 11.215.1 Detailed Description

**Name** Event Message Telemetry Packet (Short format)

Definition at line 251 of file default\_cfe\_evs\_msgdefs.h.

### 11.215.2 Field Documentation

**11.215.2.1 PacketID** `CFE_EVS_PacketID_t` `CFE_EVS_ShortEventTlm_Payload::PacketID`  
Event packet information.

Definition at line 253 of file default\_cfe\_evs\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h

## 11.216 CFE\_EVS\_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.*

**11.216.1 Detailed Description**

Write Event Services Application Information to File Command.  
Definition at line 74 of file default\_cfe\_evs\_msgstruct.h.

**11.216.2 Field Documentation****11.216.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_WriteAppDataFileCmd::CommandHeader  
Command header.

Definition at line 76 of file default\_cfe\_evs\_msgstruct.h.

**11.216.2.2 Payload** [CFE\\_EVS\\_AppDataCmd\\_Payload\\_t](#) CFE\_EVS\_WriteAppDataFileCmd::Payload  
Command payload.

Definition at line 77 of file default\_cfe\_evs\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h

**11.217 CFE\_EVS\_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.*

**11.217.1 Detailed Description**

Write Event Log to File Command.

Definition at line 65 of file default\_cfe\_evs\_msgstruct.h.

**11.217.2 Field Documentation****11.217.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_EVS\_WriteLogFileCmd::CommandHeader  
Command header.

Definition at line 67 of file default\_cfe\_evs\_msgstruct.h.

**11.217.2.2 Payload** `CFE_EVS_LogFileCmd_Payload_t` `CFE_EVS_WriteLogFileCmd::Payload`  
Command payload.

Definition at line 68 of file `default_cfe_evs_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/evs/config/default_cfe_evs_msgstruct.h`

## 11.218 CFE\_FS\_FileWriteMetaData Struct Reference

External Metadata/State object associated with background file writes.

```
#include <cfe_fs_api_typedefs.h>
```

### Data Fields

- volatile bool `IsPending`
- char `FileName` [`CFE_MISSION_MAX_PATH_LEN`]
- `uint32 FileSubType`
- char `Description` [`CFE_FS_HDR_DESC_MAX_LEN`]
- `CFE_FS_FileWriteGetData_t GetData`
- `CFE_FS_FileWriteOnEvent_t OnEvent`

### 11.218.1 Detailed Description

External Metadata/State object associated with background file writes.

Applications intending to schedule background file write jobs should instantiate this object in static/global data memory.

This keeps track of the state of the file write request(s).

Definition at line 124 of file `cfe_fs_api_typedefs.h`.

### 11.218.2 Field Documentation

#### 11.218.2.1 Description `char CFE_FS_FileWriteMetaData::Description[CFE_FS_HDR_DESC_MAX_LEN]`

Description of file (for FS header)

Definition at line 132 of file `cfe_fs_api_typedefs.h`.

#### 11.218.2.2 FileName `char CFE_FS_FileWriteMetaData::FileName[CFE_MISSION_MAX_PATH_LEN]`

Name of file to write

Definition at line 128 of file `cfe_fs_api_typedefs.h`.

#### 11.218.2.3 FileSubType `uint32 CFE_FS_FileWriteMetaData::FileSubType`

Type of file to write (for FS header)

Definition at line 131 of file `cfe_fs_api_typedefs.h`.

#### 11.218.2.4 GetData `CFE_FS_FileWriteGetData_t CFE_FS_FileWriteMetaData::GetData`

Application callback to get a data record

Definition at line 134 of file `cfe_fs_api_typedefs.h`.

**11.218.2.5 IsPending** `volatile bool CFE_FS_FileWriteMetaData::IsPending`  
Whether request is pending (volatile as it may be checked outside lock)  
Definition at line 126 of file cfe\_fs\_api\_typedefs.h.

**11.218.2.6 OnEvent** `CFE_FS_FileWriteOnEvent_t CFE_FS_FileWriteMetaData::OnEvent`  
Application callback for abstract event processing  
Definition at line 135 of file cfe\_fs\_api\_typedefs.h.  
The documentation for this struct was generated from the following file:

- [cfe/modules/core\\_api/fsw/inc/cfe\\_fs\\_api\\_typedefs.h](#)

## 11.219 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 Content Type, 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.*

### 11.219.1 Detailed Description

Standard cFE File header structure definition.

Definition at line 181 of file default\_cfe\_fs\_filedef.h.

### 11.219.2 Field Documentation

**11.219.2.1 ApplicationID** `uint32 CFE_FS_Header::ApplicationID`  
Application that generated the file.  
Definition at line 190 of file default\_cfe\_fs\_filedef.h.

**11.219.2.2 ContentType** `uint32 CFE_FS_Header::ContentType`  
Identifies the content type ('cFE1'=0x63464531)  
Definition at line 183 of file default\_cfe\_fs\_filedef.h.

**11.219.2.3 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.

**11.219.2.4 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.

**11.219.2.5 ProcessorID** `uint32 CFE_FS_Header::ProcessorID`  
Processor that generated the file.  
Definition at line 189 of file default\_cfe\_fs\_filedef.h.

**11.219.2.6 SpacecraftID** `uint32 CFE_FS_Header::SpacecraftID`  
Spacecraft that generated the file.  
Definition at line 188 of file default\_cfe\_fs\_filedef.h.

**11.219.2.7 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.

**11.219.2.8 TimeSeconds** `uint32 CFE_FS_Header::TimeSeconds`  
File creation timestamp (seconds)  
Definition at line 192 of file default\_cfe\_fs\_filedef.h.

**11.219.2.9 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

## 11.220 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.*

### 11.220.1 Detailed Description

Definition at line 138 of file default\_cfe\_sb\_msgstruct.h.

### 11.220.2 Field Documentation

**11.220.2.1 Payload** [CFE\\_SB\\_AllSubscriptionsTlm\\_Payload\\_t](#) CFE\_SB\_AllSubscriptionsTlm::Payload  
Telemetry payload.

Definition at line 141 of file default\_cfe\_sb\_msgstruct.h.

**11.220.2.2 TelemetryHeader** [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_SB\_AllSubscriptionsTlm::TelemetryHeader  
Telemetry header.

Definition at line 140 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

## 11.221 CFE\_SB\_AllSubscriptionsTlm\_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- [uint32 PktSegment](#)  
*Pkt number(starts at 1) in the series.*
- [uint32 TotalSegments](#)  
*Total number of pkts needed to complete the request.*
- [uint32 Entries](#)  
*Number of entries in the pkt.*
- [CFE\\_SB\\_SubEntries\\_t Entry \[CFE\\_MISSION\\_SB\\_SUB\\_ENTRIES\\_PER\\_PKT\]](#)  
*Array of [CFE\\_SB\\_SubEntries\\_t](#) entries.*

### 11.221.1 Detailed Description

**Name** SB Previous Subscriptions Packet

This structure defines the pkt(s) sent by SB that contains a list of all current subscriptions. This pkt is generated on cmd and intended to be used primarily by the Software Bus Networking Application (SBN). Typically, when the cmd is received there are more subscriptions than can fit in one pkt. The complete list of subscriptions is sent via a series of segmented pkts.

Definition at line 274 of file default\_cfe\_sb\_msgdefs.h.

### 11.221.2 Field Documentation

**11.221.2.1 Entries** [uint32](#) CFE\_SB\_AllSubscriptionsTlm\_Payload::Entries  
Number of entries in the pkt.

Definition at line 278 of file default\_cfe\_sb\_msgdefs.h.

**11.221.2.2 Entry** `CFE_SB_SubEntries_t` `CFE_SB_AllSubscriptionsTlm_Payload::Entry`[`CFE_MISSION_SB_SUB_ENTRIES_PER_PKT`]  
Array of `CFE_SB_SubEntries_t` entries.

Definition at line 279 of file `default_cfe_sb_msgdefs.h`.

**11.221.2.3 PktSegment** `uint32` `CFE_SB_AllSubscriptionsTlm_Payload::PktSegment`

Pkt number(starts at 1) in the series.

Definition at line 276 of file `default_cfe_sb_msgdefs.h`.

**11.221.2.4 TotalSegments** `uint32` `CFE_SB_AllSubscriptionsTlm_Payload::TotalSegments`

Total number of pkts needed to complete the request.

Definition at line 277 of file `default_cfe_sb_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/sb/config/default_cfe_sb_msgdefs.h`

## 11.222 CFE\_SB\_DisableRouteCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_SB_RouteCmd_Payload_t Payload`  
*Command payload.*

### 11.222.1 Detailed Description

Definition at line 110 of file `default_cfe_sb_msgstruct.h`.

### 11.222.2 Field Documentation

**11.222.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_SB_DisableRouteCmd::CommandHeader`  
Command header.

Definition at line 112 of file `default_cfe_sb_msgstruct.h`.

**11.222.2.2 Payload** `CFE_SB_RouteCmd_Payload_t` `CFE_SB_DisableRouteCmd::Payload`  
Command payload.

Definition at line 113 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`

## 11.223 CFE\_SB\_DisableSubReportingCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

### 11.223.1 Detailed Description

Definition at line 60 of file default\_cfe\_sb\_msgstruct.h.

### 11.223.2 Field Documentation

#### 11.223.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_DisableSubReportingCmd::CommandHeader

Definition at line 62 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.224 CFE\_SB\_EnableRouteCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_SB\\_RouteCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.224.1 Detailed Description

Definition at line 104 of file default\_cfe\_sb\_msgstruct.h.

### 11.224.2 Field Documentation

#### 11.224.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_EnableRouteCmd::CommandHeader

Command header.

Definition at line 106 of file default\_cfe\_sb\_msgstruct.h.

#### 11.224.2.2 Payload [CFE\\_SB\\_RouteCmd\\_Payload\\_t](#) CFE\_SB\_EnableRouteCmd::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

## 11.225 CFE\_SB\_EnableSubReportingCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader

### 11.225.1 Detailed Description

Definition at line 55 of file default\_cfe\_sb\_msgstruct.h.

### 11.225.2 Field Documentation

#### 11.225.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_EnableSubReportingCmd::CommandHeader

Definition at line 57 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.226 CFE\_SB\_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.*

### 11.226.1 Detailed Description

Definition at line 120 of file default\_cfe\_sb\_msgstruct.h.

### 11.226.2 Field Documentation

#### 11.226.2.1 Payload [CFE\\_SB\\_HousekeepingTlm\\_Payload\\_t](#) CFE\_SB\_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 123 of file default\_cfe\_sb\_msgstruct.h.

#### 11.226.2.2 TelemetryHeader [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_SB\_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 122 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.227 CFE\_SB\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- uint8 CommandCounter  
*Count of valid commands received.*
- uint8 CommandErrorCounter

- `uint8 NoSubscribersCounter`  
*Count of invalid commands received.*
- `uint8 MsgSendErrorCounter`  
*Count pkts sent with no subscribers.*
- `uint8 MsgReceiveErrorCounter`  
*Count of message send errors.*
- `uint8 InternalErrorCounter`  
*Count of message receive errors.*
- `uint8 CreatePipeErrorCounter`  
*Count of queue read or write errors.*
- `uint8 SubscribeErrorCounter`  
*Count of errors in create pipe API.*
- `uint8 PipeOptsErrorCounter`  
*Count of errors in set/get pipe options API.*
- `uint8 DuplicateSubscriptionsCounter`  
*Count of duplicate subscriptions.*
- `uint8 GetPipeldByNameErrorCounter`  
*Count of errors in get pipe id by name API.*
- `uint8 Spare2Align [1]`  
*Spare bytes to ensure alignment.*
- `uint16 PipeOverflowErrorCounter`  
*Count of pipe overflow errors.*
- `uint16 MsgLimitErrorCounter`  
*Count of msg id to pipe errors.*
- `CFE_ES_MemHandle_t MemPoolHandle`  
*Handle to SB's Memory Pool.*
- `uint32 MemInUse`  
*Memory in use.*
- `uint32 UnmarkedMem`  
*cfg param CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES minus Peak Memory in use*

### 11.227.1 Detailed Description

**Name** Software Bus task housekeeping Packet

Definition at line 67 of file default\_cfe\_sb\_msgdefs.h.

### 11.227.2 Field Documentation

**11.227.2.1 CommandCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::CommandCounter`  
Count of valid commands received.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_CMDPC

Definition at line 69 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.2 CommandErrorCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::CommandErrorCounter`  
Count of invalid commands received.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_CMDEC

Definition at line 71 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.3 CreatePipeErrorCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::CreatePipeErrorCounter`  
Count of errors in create pipe API.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_NewPipeEC

Definition at line 82 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.4 DuplicateSubscriptionsCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::DuplicateSubscriptionsCounter`  
Count of duplicate subscriptions.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_DupSubCnt

Definition at line 88 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.5 GetPipeIdByNameErrorCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::GetPipeIdByNameErrorCounter`  
Count of errors in get pipe id by name API.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_GetPipeIDByNameEC

Definition at line 90 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.6 InternalErrorCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::InternalErrorCounter`  
Count of queue read or write errors.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_InternalEC

Definition at line 80 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.7 MemInUse** `uint32 CFE_SB_HousekeepingTlm_Payload::MemInUse`  
Memory in use.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_MemInUse

Definition at line 103 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.8 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 100 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.9 MsgLimitErrorCounter** `uint16` CFE\_SB\_HousekeepingTlm\_Payload::MsgLimitErrorCounter  
Count of msg id to pipe errors.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_MsgLimEC

Definition at line 97 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.10 MsgReceiveErrorCounter** `uint8` CFE\_SB\_HousekeepingTlm\_Payload::MsgReceiveErrorCounter  
Count of message receive errors.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_MsgRecEC

Definition at line 78 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.11 MsgSendErrorCounter** `uint8` CFE\_SB\_HousekeepingTlm\_Payload::MsgSendErrorCounter  
Count of message send errors.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_MsgSndEC

Definition at line 75 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.12 NoSubscribersCounter** `uint8` CFE\_SB\_HousekeepingTlm\_Payload::NoSubscribersCounter  
Count pkts sent with no subscribers.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_NoSubEC

Definition at line 73 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.13 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 86 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.14 PipeOverflowErrorCounter** `uint16` CFE\_SB\_HousekeepingTlm\_Payload::PipeOverflowError←  
Counter  
Count of pipe overflow errors.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_PipeOvrEC

Definition at line 95 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.15 Spare2Align** `uint8` CFE\_SB\_HousekeepingTlm\_Payload::Spare2Align[1]  
Spare bytes to ensure alignment.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Spare2Align[2]

Definition at line 92 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.16 SubscribeErrorCounter** `uint8 CFE_SB_HousekeepingTlm_Payload::SubscribeErrorCounter`  
Count of errors in subscribe API.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_SubscrEC

Definition at line 84 of file default\_cfe\_sb\_msgdefs.h.

**11.227.2.17 UnmarkedMem** `uint32 CFE_SB_HousekeepingTlm_Payload::UnmarkedMem`  
cfg param CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES minus Peak Memory in use

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_UnMarkedMem

Definition at line 106 of file default\_cfe\_sb\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.228 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.*

### 11.228.1 Detailed Description

Software Bus generic message.

Definition at line 142 of file cfe\_sb\_api\_typedefs.h.

### 11.228.2 Field Documentation

**11.228.2.1 LongDouble** `long double CFE_SB_Msg::LongDouble`  
Align to support Long Double.  
Definition at line 146 of file cfe\_sb\_api\_typedefs.h.

**11.228.2.2 LongInt** `long long int CFE_SB_Msg::LongInt`  
Align to support Long Integer.  
Definition at line 145 of file cfe\_sb\_api\_typedefs.h.

**11.228.2.3 Msg** [CFE\\_MSG\\_Message\\_t](#) CFE\_SB\_Msg::Msg

Base message type without enforced alignment.

Definition at line 144 of file cfe\_sb\_api\_typedefs.h.

Referenced by CF\_AppPipe(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_ReceiveMessage(), CF\_CFDP\_Send(), and CF←\_ProcessGroundCommand().

The documentation for this union was generated from the following file:

- [cfe/modules/core\\_api/fsw/inc/cfe\\_sb\\_api\\_typedefs.h](#)

**11.229 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](#)

**11.229.1 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 102 of file default\_cfe\_sb\_extern\_typedefs.h.

**11.229.2 Field Documentation****11.229.2.1 Value** [CFE\\_SB\\_MsgId\\_Atom\\_t](#) CFE\_SB\_MsgId\_t::Value

Definition at line 104 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](#)

**11.230 CFE\_SB\_MsgMapFileEntry Struct Reference**

SB Map File Entry.

```
#include <default_cfe_sb_msgdefs.h>
```

**Data Fields**

- [CFE\\_SB\\_MsgId\\_t MsgId](#)

*Message Id which has been subscribed to.*

- [CFE\\_SB\\_RoutId\\_Atom\\_t Index](#)

*Routing raw index value (0 based, not Route ID)*

### 11.230.1 Detailed Description

SB Map File Entry.

Structure of one element of the map information in response to [CFE\\_SB\\_WRITE\\_MAP\\_INFO\\_CC](#)

Definition at line 226 of file default\_cfe\_sb\_msgdefs.h.

### 11.230.2 Field Documentation

#### 11.230.2.1 Index [CFE\\_SB\\_RouteId\\_Atom\\_t](#) CFE\_SB\_MsgMapFileEntry::Index

Routing raw index value (0 based, not Route ID)

Definition at line 229 of file default\_cfe\_sb\_msgdefs.h.

#### 11.230.2.2 MsgId [CFE\\_SB\\_MsgId\\_t](#) CFE\_SB\_MsgMapFileEntry::MsgId

Message Id which has been subscribed to.

Definition at line 228 of file default\_cfe\_sb\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.231 CFE\_SB\_NoopCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader

#### 11.231.1 Detailed Description

Definition at line 45 of file default\_cfe\_sb\_msgstruct.h.

### 11.231.2 Field Documentation

#### 11.231.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_NoopCmd::CommandHeader

Definition at line 47 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

## 11.232 CFE\_SB\_PipeDepthStats Struct Reference

SB Pipe Depth Statistics.

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- [CFE\\_SB\\_Pipeld\\_t](#) Pipeld

*Pipe Id associated with the stats below.*

- [uint16](#) MaxQueueDepth

*Number of messages the pipe can hold.*

- **uint16 CurrentQueueDepth**  
*Number of messages currently on the pipe.*
- **uint16 PeakQueueDepth**  
*Peak number of messages that have been on the pipe.*
- **uint16 Spare**  
*Spare word to ensure alignment.*

### 11.232.1 Detailed Description

SB Pipe Depth Statistics.

Used in SB Statistics Telemetry Packet [CFE\\_SB\\_StatsTlm\\_t](#)

Definition at line 115 of file default\_cfe\_sb\_msgdefs.h.

### 11.232.2 Field Documentation

#### 11.232.2.1 CurrentQueueDepth [uint16 CFE\\_SB\\_PipeDepthStats::CurrentQueueDepth](#)

Number of messages currently on the pipe.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDINUSE

Definition at line 121 of file default\_cfe\_sb\_msgdefs.h.

#### 11.232.2.2 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\_PDEPTH

Definition at line 119 of file default\_cfe\_sb\_msgdefs.h.

#### 11.232.2.3 PeakQueueDepth [uint16 CFE\\_SB\\_PipeDepthStats::PeakQueueDepth](#)

Peak number of messages that have been on the pipe.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDPKINUSE

Definition at line 123 of file default\_cfe\_sb\_msgdefs.h.

#### 11.232.2.4 PipeId [CFE\\_SB\\_PipeId\\_t CFE\\_SB\\_PipeDepthStats::PipeId](#)

Pipe Id associated with the stats below.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDPIPEID

Definition at line 117 of file default\_cfe\_sb\_msgdefs.h.

#### 11.232.2.5 Spare [uint16 CFE\\_SB\\_PipeDepthStats::Spare](#)

Spare word to ensure alignment.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES].SB\_PDSPARE

Definition at line 125 of file default\_cfe\_sb\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.233 CFE\_SB\_PipeInfoEntry Struct Reference

SB Pipe Information File Entry.

```
#include <default_cfe_sb_msgdefs.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]`

### 11.233.1 Detailed Description

SB Pipe Information File Entry.

This statistics structure is output as part of the CFE SB "Send Pipe Info" command (CFE\_SB\_SEND\_PIPE\_INFO\_CC). Previous versions of CFE simply wrote the internal CFE\_SB\_PipeD\_t object to the file, but this also contains information such as pointers which are not relevant outside the running CFE process.

By defining the pipe info structure separately, it also provides some independence, such that the internal CFE\_SB\_PipeD\_t definition can evolve without changing the binary format of the information file.

Definition at line 144 of file default\_cfe\_sb\_msgdefs.h.

### 11.233.2 Field Documentation

#### 11.233.2.1 **AppId** `CFE_ES_AppId_t CFE_SB_PipeInfoEntry::AppId`

The runtime ID of the application that owns the pipe

Definition at line 147 of file default\_cfe\_sb\_msgdefs.h.

#### 11.233.2.2 **AppName** `char CFE_SB_PipeInfoEntry::AppName [CFE_MISSION_MAX_API_LEN]`

The Name of the application that owns the pipe

Definition at line 149 of file default\_cfe\_sb\_msgdefs.h.

#### 11.233.2.3 **CurrentQueueDepth** `uint16 CFE_SB_PipeInfoEntry::CurrentQueueDepth`

The current depth of the pipe

Definition at line 151 of file default\_cfe\_sb\_msgdefs.h.

#### 11.233.2.4 **MaxQueueDepth** `uint16 CFE_SB_PipeInfoEntry::MaxQueueDepth`

The allocated depth of the pipe (max capacity)

Definition at line 150 of file default\_cfe\_sb\_msgdefs.h.

**11.233.2.5 Opts** `uint8` `CFE_SB_PipeInfoEntry::Opts`  
Pipe options set (bitmask)  
Definition at line 154 of file default\_cfe\_sb\_msgdefs.h.

**11.233.2.6 PeakQueueDepth** `uint16` `CFE_SB_PipeInfoEntry::PeakQueueDepth`  
The peak depth of the pipe (high watermark)  
Definition at line 152 of file default\_cfe\_sb\_msgdefs.h.

**11.233.2.7 PipeId** `CFE_SB_PipeId_t` `CFE_SB_PipeInfoEntry::PipeId`  
The runtime ID of the pipe  
Definition at line 146 of file default\_cfe\_sb\_msgdefs.h.

**11.233.2.8 PipeName** `char` `CFE_SB_PipeInfoEntry::PipeName[CFE_MISSION_MAX_API_LEN]`  
The Name of the pipe  
Definition at line 148 of file default\_cfe\_sb\_msgdefs.h.

**11.233.2.9 SendErrors** `uint16` `CFE_SB_PipeInfoEntry::SendErrors`  
Number of errors when writing to this pipe  
Definition at line 153 of file default\_cfe\_sb\_msgdefs.h.

**11.233.2.10 Spare** `uint8` `CFE_SB_PipeInfoEntry::Spare[3]`  
Padding to make this structure a multiple of 4 bytes  
Definition at line 155 of file default\_cfe\_sb\_msgdefs.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.234 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.*

### 11.234.1 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 119 of file default\_cfe\_sb\_extern\_typedefs.h.

### 11.234.2 Field Documentation

**11.234.2.1 Priority** `uint8 CFE_SB_Qos_t::Priority`

Specify high(1) or low(0) message priority for off-board routing, currently unused.

Definition at line 121 of file default\_cfe\_sb\_extern\_typedefs.h.

**11.234.2.2 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 122 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

## 11.235 CFE\_SB\_ResetCountersCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

#### 11.235.1 Detailed Description

Definition at line 50 of file default\_cfe\_sb\_msgstruct.h.

#### 11.235.2 Field Documentation

##### 11.235.2.1 CommandHeader

`CFE_MSG_CommandHeader_t CFE_SB_ResetCountersCmd::CommandHeader`

Definition at line 52 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

## 11.236 CFE\_SB\_RouteCmd\_Payload Struct Reference

Enable/Disable Route Command Payload.

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- `CFE_SB_MsgId_t MsgId`  
*Message ID of route to be enabled or disabled* `CFE_SB_MsgId_t`.
- `CFE_SB_Pipeld_t Pipe`  
*Pipe ID of route to be enabled or disabled* `CFE_SB_Pipeld_t`.
- `uint8 Spare`  
*Spare byte to make command even number of bytes.*

#### 11.236.1 Detailed Description

Enable/Disable Route Command Payload.

This structure contains a definition used by two SB commands, 'Enable Route' `CFE_SB_ENABLE_ROUTE_CC` and 'Disable Route' `CFE_SB_DISABLE_ROUTE_CC`. A route is the destination pipe for a particular message and is therefore defined as a MsgId and Pipeld combination.

Definition at line 53 of file default\_cfe\_sb\_msgdefs.h.

## 11.236.2 Field Documentation

**11.236.2.1 MsgId** [CFE\\_SB\\_MsgId\\_t](#) CFE\_SB\_RouteCmd\_Payload::MsgId  
Message ID of route to be enabled or disabled [CFE\\_SB\\_MsgId\\_t](#).  
Definition at line 55 of file default\_cfe\_sb\_msgdefs.h.

**11.236.2.2 Pipe** [CFE\\_SB\\_PipeId\\_t](#) CFE\_SB\_RouteCmd\_Payload::Pipe  
Pipe ID of route to be enabled or disabled [CFE\\_SB\\_PipeId\\_t](#).  
Definition at line 56 of file default\_cfe\_sb\_msgdefs.h.

**11.236.2.3 Spare** [uint8](#) CFE\_SB\_RouteCmd\_Payload::Spare  
Spare byte to make command even number of bytes.  
Definition at line 57 of file default\_cfe\_sb\_msgdefs.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.237 CFE\_SB\_RoutingFileEntry Struct Reference

SB Routing File Entry.

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- [CFE\\_SB\\_MsgId\\_t](#) MsgId  
*Message Id portion of the route.*
- [CFE\\_SB\\_PipeId\\_t](#) PipeId  
*Pipe Id portion of the route.*
- [uint8](#) State  
*Route Enabled or Disabled.*
- [uint16](#) MsgCnt  
*Number of msgs with this MsgId sent to this PipeId.*
- char [AppName](#) [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]  
*Pipe Depth Statistics.*
- char [PipeName](#) [[CFE\\_MISSION\\_MAX\\_API\\_LEN](#)]  
*Pipe Depth Statistics.*

### 11.237.1 Detailed Description

SB Routing File Entry.

Structure of one element of the routing information in response to [CFE\\_SB\\_WRITE\\_ROUTING\\_INFO\\_CC](#)  
Definition at line 211 of file default\_cfe\_sb\_msgdefs.h.

## 11.237.2 Field Documentation

**11.237.2.1 AppName** `char CFE_SB_RoutingFileEntry::AppName[CFE_MISSION_MAX_API_LEN]`  
Pipe Depth Statistics.  
Definition at line 217 of file default\_cfe\_sb\_msgdefs.h.

**11.237.2.2 MsgCnt** `uint16 CFE_SB_RoutingFileEntry::MsgCnt`  
Number of msgs with this MsgId sent to this PipeId.  
Definition at line 216 of file default\_cfe\_sb\_msgdefs.h.

**11.237.2.3 MsgId** `CFE_SB_MsgId_t CFE_SB_RoutingFileEntry::MsgId`  
Message Id portion of the route.  
Definition at line 213 of file default\_cfe\_sb\_msgdefs.h.

**11.237.2.4 PipeId** `CFE_SB_PipeId_t CFE_SB_RoutingFileEntry::PipeId`  
Pipe Id portion of the route.  
Definition at line 214 of file default\_cfe\_sb\_msgdefs.h.

**11.237.2.5 PipeName** `char CFE_SB_RoutingFileEntry::PipeName[CFE_MISSION_MAX_API_LEN]`  
Pipe Depth Statistics.  
Definition at line 218 of file default\_cfe\_sb\_msgdefs.h.

**11.237.2.6 State** `uint8 CFE_SB_RoutingFileEntry::State`  
Route Enabled or Disabled.  
Definition at line 215 of file default\_cfe\_sb\_msgdefs.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.238 CFE\_SB\_SendHkCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

#### 11.238.1 Detailed Description

Definition at line 75 of file default\_cfe\_sb\_msgstruct.h.

#### 11.238.2 Field Documentation

**11.238.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_SB_SendHkCmd::CommandHeader`  
Definition at line 77 of file default\_cfe\_sb\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.239 CFE\_SB\_SendPrevSubsCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)

#### 11.239.1 Detailed Description

Definition at line 70 of file default\_cfe\_sb\_msgstruct.h.

#### 11.239.2 Field Documentation

##### 11.239.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t CFE\\_SB\\_SendPrevSubsCmd::CommandHeader](#)

Definition at line 72 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](#)

## 11.240 CFE\_SB\_SendSbStatsCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)

#### 11.240.1 Detailed Description

Definition at line 65 of file default\_cfe\_sb\_msgstruct.h.

#### 11.240.2 Field Documentation

##### 11.240.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t CFE\\_SB\\_SendSbStatsCmd::CommandHeader](#)

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](#)

## 11.241 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.*

### 11.241.1 Detailed Description

Definition at line 132 of file default\_cfe\_sb\_msgstruct.h.

### 11.241.2 Field Documentation

**11.241.2.1 Payload** `CFE_SB_SingleSubscriptionTlm_Payload_t` `CFE_SB_SingleSubscriptionTlm::Payload`  
Telemetry payload.

Definition at line 135 of file default\_cfe\_sb\_msgstruct.h.

**11.241.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_SB_SingleSubscriptionTlm::TelemetryHeader`  
Telemetry header.

Definition at line 134 of file default\_cfe\_sb\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h

## 11.242 CFE\_SB\_SingleSubscriptionTlm\_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- `uint8 SubType`  
*Subscription or Unsubscription.*
- `CFE_SB_MsgId_t MsgId`  
*MsgId subscribed or unsubscribe to.*
- `CFE_SB_Qos_t Qos`  
*Quality of Service, used only for interprocessor communication.*
- `CFE_SB_PipeId_t Pipe`  
*Destination pipe id to send above msg id*

### 11.242.1 Detailed Description

**Name** SB Subscription Report Packet

This structure defines the pkt sent by SB when a subscription or a request to unsubscribe is received while subscription reporting is enabled. By default subscription reporting is disabled. This feature is intended to be used primarily by Software Bus Networking Application (SBN)

See also

`CFE_SB_ENABLE_SUB_REPORTING_CC`, `CFE_SB_DISABLE_SUB_REPORTING_CC`

Definition at line 242 of file default\_cfe\_sb\_msgdefs.h.

### 11.242.2 Field Documentation

**11.242.2.1 MsgId** `CFE_SB_MsgId_t` `CFE_SB_SingleSubscriptionTlm_Payload::MsgId`  
MsgId subscribed or unsubscribe to.  
Definition at line 245 of file default\_cfe\_sb\_msgdefs.h.

**11.242.2.2 Pipe** `CFE_SB_PipeId_t` `CFE_SB_SingleSubscriptionTlm_Payload::Pipe`  
Destination pipe id to send above msg id

Definition at line 247 of file default\_cfe\_sb\_msgdefs.h.

**11.242.2.3 Qos** `CFE_SB_Qos_t` `CFE_SB_SingleSubscriptionTlm_Payload::Qos`  
Quality of Service, used only for interprocessor communication.  
Definition at line 246 of file default\_cfe\_sb\_msgdefs.h.

**11.242.2.4 SubType** `uint8` `CFE_SB_SingleSubscriptionTlm_Payload::SubType`  
Subscription or Unsubscription.

Definition at line 244 of file default\_cfe\_sb\_msgdefs.h.

The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default\\_cfe\\_sb\\_msgdefs.h](#)

## 11.243 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.*

### 11.243.1 Detailed Description

Definition at line 126 of file default\_cfe\_sb\_msgstruct.h.

### 11.243.2 Field Documentation

**11.243.2.1 Payload** `CFE_SB_StatsTlm_Payload_t` `CFE_SB_StatsTlm::Payload`  
Telemetry payload.  
Definition at line 129 of file default\_cfe\_sb\_msgstruct.h.

**11.243.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_SB_StatsTlm::TelemetryHeader`  
Telemetry header.  
Definition at line 128 of file default\_cfe\_sb\_msgstruct.h.  
The documentation for this struct was generated from the following file:

- [cfe/modules/sb/config/default\\_cfe\\_sb\\_msgstruct.h](#)

## 11.244 CFE\_SB\_StatsTim\_Payload Struct Reference

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- `uint32 MsgIdsInUse`  
*Current number of MsgIds with a destination.*
- `uint32 PeakMsgIdsInUse`  
*Peak number of MsgIds with a destination.*
- `uint32 MaxMsgIdsAllowed`  
*cFE Cfg Param `CFE_PLATFORM_SB_MAX_MSG_IDS`*
- `uint32 PipesInUse`  
*Number of pipes currently in use.*
- `uint32 PeakPipesInUse`  
*Peak number of pipes since last reboot.*
- `uint32 MaxPipesAllowed`  
*cFE Cfg Param `CFE_PLATFORM_SB_MAX_PIPES`*
- `uint32 MemInUse`  
*Memory bytes currently in use for SB msg transfers.*
- `uint32 PeakMemInUse`  
*Peak memory bytes in use for SB msg transfers.*
- `uint32 MaxMemAllowed`  
*cFE Cfg Param `CFE_PLATFORM_SB_BUF_MEMORY_BYTES`*
- `uint32 SubscriptionsInUse`  
*Number of current subscriptions.*
- `uint32 PeakSubscriptionsInUse`  
*Peak number of subscriptions.*
- `uint32 MaxSubscriptionsAllowed`  
*product of `CFE_PLATFORM_SB_MAX_MSG_IDS` and `CFE_PLATFORM_SB_MAX_DEST_PER_PKT`*
- `uint32 SBBuffersInUse`  
*Number of SB message buffers currently in use.*
- `uint32 PeakSBBuffersInUse`  
*Max number of SB message buffers in use.*
- `uint32 MaxPipeDepthAllowed`  
*Maximum allowed pipe depth.*
- `CFE_SB_PipeDepthStats_t PipeDepthStats [CFE_MISSION_SB_MAX_PIPES]`  
*Pipe Depth Statistics `CFE_SB_PipeDepthStats_t`.*

### 11.244.1 Detailed Description

**Name** SB Statistics Telemetry Packet

SB Statistics packet sent in response to `CFE_SB_SEND_SB_STATS_CC`  
Definition at line 163 of file `default_cfe_sb_msgdefs.h`.

### 11.244.2 Field Documentation

**11.244.2.1 MaxMemAllowed** `uint32` `CFE_SB_StatsTlm_Payload::MaxMemAllowed`  
cFE Cfg Param [CFE\\_PLATFORM\\_SB\\_BUFS\\_MEMORY\\_BYTES](#)

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMMBMALW`

Definition at line 183 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.2 MaxMsgIdsAllowed** `uint32` `CFE_SB_StatsTlm_Payload::MaxMsgIdsAllowed`  
cFE Cfg Param [CFE\\_PLATFORM\\_SB\\_MAX\\_MSG\\_IDS](#)

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMMMIDALW`

Definition at line 169 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.3 MaxPipeDepthAllowed** `uint32` `CFE_SB_StatsTlm_Payload::MaxPipeDepthAllowed`  
Maximum allowed pipe depth.

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMMPDALW`

Definition at line 199 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.4 MaxPipesAllowed** `uint32` `CFE_SB_StatsTlm_Payload::MaxPipesAllowed`  
cFE Cfg Param [CFE\\_PLATFORM\\_SB\\_MAX\\_PIPES](#)

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMMPALW`

Definition at line 176 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.5 MaxSubscriptionsAllowed** `uint32` `CFE_SB_StatsTlm_Payload::MaxSubscriptionsAllowed`  
product of [CFE\\_PLATFORM\\_SB\\_MAX\\_MSG\\_IDS](#) and [CFE\\_PLATFORM\\_SB\\_MAX\\_DEST\\_PER\\_PKT](#)

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMMSALW`

Definition at line 190 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.6 MemInUse** `uint32` `CFE_SB_StatsTlm_Payload::MemInUse`  
Memory bytes currently in use for SB msg transfers.

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMBMIU`

Definition at line 179 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.7 MsgIdsInUse** `uint32` `CFE_SB_StatsTlm_Payload::MsgIdsInUse`  
Current number of MsgIds with a destination.

**Telemetry Mnemonic(s)** `$sc_$cpu_SB_Stat.SB_SMMIDIU`

Definition at line 165 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.8 PeakMemInUse** `uint32 CFE_SB_StatsTlm_Payload::PeakMemInUse`  
Peak memory bytes in use for SB msg transfers.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPBMIU

Definition at line 181 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.9 PeakMsgIdsInUse** `uint32 CFE_SB_StatsTlm_Payload::PeakMsgIdsInUse`  
Peak number of MsgIds with a destination.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPMIDIU

Definition at line 167 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.10 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 174 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.11 PeakSBBuffersInUse** `uint32 CFE_SB_StatsTlm_Payload::PeakSBBuffersInUse`  
Max number of SB message buffers in use.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPSBBIU

Definition at line 196 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.12 PeakSubscriptionsInUse** `uint32 CFE_SB_StatsTlm_Payload::PeakSubscriptionsInUse`  
Peak number of subscriptions.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPSIU

Definition at line 188 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.13 PipeDepthStats** `CFE_SB_PipeDepthStats_t CFE_SB_StatsTlm_Payload::PipeDepthStats[CFE_MISSION_SB_MAX_PIPES]`  
Pipe Depth Statistics `CFE_SB_PipeDepthStats_t`.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPDS[CFE\_PLATFORM\_SB\_MAX\_PIPES]

Definition at line 202 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.14 PipesInUse** `uint32 CFE_SB_StatsTlm_Payload::PipesInUse`  
Number of pipes currently in use.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMPIU

Definition at line 172 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.15 SBBuffersInUse** `uint32` `CFE_SB_StatsTlm_Payload::SBBuffersInUse`  
Number of SB message buffers currently in use.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMSBBIU

Definition at line 194 of file default\_cfe\_sb\_msgdefs.h.

**11.244.2.16 SubscriptionsInUse** `uint32` `CFE_SB_StatsTlm_Payload::SubscriptionsInUse`  
Number of current subscriptions.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_SB\_Stat.SB\_SMSIU

Definition at line 186 of file default\_cfe\_sb\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.245 CFE\_SB\_SubEntries Struct Reference

SB Previous Subscriptions Entry.

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- **CFE\_SB\_MsgId\_t MsgId**  
*MsgId portion of the subscription.*
- **CFE\_SB\_Qos\_t Qos**  
*Qos portion of the subscription.*
- **CFE\_SB\_PipeId\_t Pipe**  
*PipeId portion of the subscription.*

### 11.245.1 Detailed Description

SB Previous Subscriptions Entry.

This structure defines an entry used in the `CFE_SB_PrevSubsPkt_t` Intended to be used primarily by Software Bus Networking Application (SBN)

Used in structure definition `CFE_SB_AllSubscriptionsTlm_t`

Definition at line 258 of file default\_cfe\_sb\_msgdefs.h.

### 11.245.2 Field Documentation

#### 11.245.2.1 MsgId `CFE_SB_MsgId_t` `CFE_SB_SubEntries::MsgId`

MsgId portion of the subscription.

Definition at line 260 of file default\_cfe\_sb\_msgdefs.h.

#### 11.245.2.2 Pipe `CFE_SB_PipeId_t` `CFE_SB_SubEntries::Pipe`

PipeId portion of the subscription.

Definition at line 262 of file default\_cfe\_sb\_msgdefs.h.

**11.245.2.3 Qos** [CFE\\_SB\\_Qos\\_t](#) CFE\_SB\_SubEntries::Qos

Qos portion of the subscription.

Definition at line 261 of file default\_cfe\_sb\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.246 CFE\_SB\_WriteFileInfoCmd\_Payload Struct Reference

Write File Info Command Payload.

```
#include <default_cfe_sb_msgdefs.h>
```

### Data Fields

- char [Filename \[CFE\\_MISSION\\_MAX\\_PATH\\_LEN\]](#)

*Path and Filename of data to be loaded.*

### 11.246.1 Detailed Description

Write File Info Command Payload.

This structure contains a generic definition used by SB commands that write to a file

Definition at line 40 of file default\_cfe\_sb\_msgdefs.h.

### 11.246.2 Field Documentation

#### 11.246.2.1 Filename

[char CFE\\_SB\\_WriteFileInfoCmd\\_Payload::Filename \[CFE\\_MISSION\\_MAX\\_PATH\\_LEN\]](#)

Path and Filename of data to be loaded.

Definition at line 42 of file default\_cfe\_sb\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h

## 11.247 CFE\_SB\_WriteMapInfoCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t CommandHeader](#)

*Command header.*

- [CFE\\_SB\\_WriteFileInfoCmd\\_Payload\\_t Payload](#)

*Command payload.*

### 11.247.1 Detailed Description

Definition at line 95 of file default\_cfe\_sb\_msgstruct.h.

### 11.247.2 Field Documentation

**11.247.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_WriteMapInfoCmd::CommandHeader  
Command header.

Definition at line 97 of file default\_cfe\_sb\_msgstruct.h.

**11.247.2.2 Payload** [CFE\\_SB\\_WriteFileInfoCmd\\_Payload\\_t](#) CFE\_SB\_WriteMapInfoCmd::Payload

Command payload.

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

## 11.248 CFE\_SB\_WritePipeInfoCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_SB\\_WriteFileInfoCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.248.1 Detailed Description

Definition at line 89 of file default\_cfe\_sb\_msgstruct.h.

### 11.248.2 Field Documentation

**11.248.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_SB\_WritePipeInfoCmd::CommandHeader  
Command header.

Definition at line 91 of file default\_cfe\_sb\_msgstruct.h.

**11.248.2.2 Payload** [CFE\\_SB\\_WriteFileInfoCmd\\_Payload\\_t](#) CFE\_SB\_WritePipeInfoCmd::Payload  
Command payload.

Definition at line 92 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

## 11.249 CFE\_SB\_WriteRoutingInfoCmd Struct Reference

```
#include <default_cfe_sb_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_SB\\_WriteFileInfoCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.249.1 Detailed Description

Definition at line 83 of file default\_cfe\_sb\_msgstruct.h.

### 11.249.2 Field Documentation

#### 11.249.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_SB_WriteRoutingInfoCmd::CommandHeader`

Command header.

Definition at line 85 of file default\_cfe\_sb\_msgstruct.h.

#### 11.249.2.2 Payload `CFE_SB_WriteFileInfoCmd_Payload_t` `CFE_SB_WriteRoutingInfoCmd::Payload`

Command payload.

Definition at line 86 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](#)

## 11.250 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.*

### 11.250.1 Detailed Description

Abort Load Command.

Definition at line 124 of file default\_cfe\_tbl\_msgstruct.h.

### 11.250.2 Field Documentation

#### 11.250.2.1 CommandHeader `CFE_MSG_CommandHeader_t` `CFE_TBL_AbortLoadCmd::CommandHeader`

Command header.

Definition at line 126 of file default\_cfe\_tbl\_msgstruct.h.

#### 11.250.2.2 Payload `CFE_TBL_AbortLoadCmd_Payload_t` `CFE_TBL_AbortLoadCmd::Payload`

Command payload.

Definition at line 127 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](#)

## 11.251 CFE\_TBL\_AbortLoadCmd\_Payload Struct Reference

Abort Load Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

### Data Fields

- char **TableName** [CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN]

*Full Name of Table whose load is to be aborted.*

### 11.251.1 Detailed Description

Abort Load Command Payload.

For command details, see [CFE\\_TBL\\_ABORT\\_LOAD\\_CC](#)

Definition at line 146 of file default\_cfe\_tbl\_msgdefs.h.

### 11.251.2 Field Documentation

#### 11.251.2.1 TableName char CFE\_TBL\_AbortLoadCmd\_Payload::TableName [CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN]

Full Name of Table whose load is to be aborted.

ASCII string containing full table name identifier of a table whose load is to be aborted

Definition at line 148 of file default\_cfe\_tbl\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h

## 11.252 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.*

### 11.252.1 Detailed Description

Activate Table Command.

Definition at line 88 of file default\_cfe\_tbl\_msgstruct.h.

### 11.252.2 Field Documentation

#### 11.252.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_ActivateCmd::CommandHeader

Command header.

Definition at line 90 of file default\_cfe\_tbl\_msgstruct.h.

**11.252.2.2 Payload** `CFE_TBL_ActivateCmd_Payload_t` `CFE_TBL_ActivateCmd::Payload`  
Command payload.

Definition at line 91 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`

## 11.253 CFE\_TBL\_ActivateCmd\_Payload Struct Reference

Activate Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

### Data Fields

- `char TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`  
*Full Name of Table to be activated.*

#### 11.253.1 Detailed Description

Activate Table Command Payload.

For command details, see [CFE\\_TBL\\_ACTIVATE\\_CC](#)

Definition at line 93 of file `default_cfe_tbl_msgdefs.h`.

#### 11.253.2 Field Documentation

**11.253.2.1 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 95 of file `default_cfe_tbl_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h`

## 11.254 CFE\_TBL\_CombinedFileHdr Struct Reference

Complete header for CFE table files.

```
#include <default_cfe_tbl_extern_typedefs.h>
```

### Data Fields

- [CFE\\_FS\\_Header\\_t Std](#)
- [CFE\\_TBL\\_File\\_Hdr\\_t Tbl](#)

#### 11.254.1 Detailed Description

Complete header for CFE table files.

Table files always have a combination of the standard file header and the table-specific file header. This struct just combines the two and makes for an easier item to pass around, simplifying APIs

Definition at line 80 of file `default_cfe_tbl_extern_typedefs.h`.

#### 11.254.2 Field Documentation

**11.254.2.1 Std** [CFE\\_FS\\_Header\\_t](#) CFE\_TBL\_CombinedFileHdr::Std  
Definition at line 82 of file default\_cfe\_tbl\_extern\_typedefs.h.

**11.254.2.2 Tbl** [CFE\\_TBL\\_File\\_Hdr\\_t](#) CFE\_TBL\_CombinedFileHdr::Tbl  
Definition at line 83 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

## 11.255 CFE\_TBL\_DelCDSCmd\_Payload Struct Reference

Delete Critical Table CDS Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

### Data Fields

- char **TableName** [[CFE\\_MISSION\\_TBL\\_MAX\\_FULL\\_NAME\\_LEN](#)]  
*Full Name of Table whose CDS is to be deleted.*

### 11.255.1 Detailed Description

Delete Critical Table CDS Command Payload.

For command details, see [CFE\\_TBL\\_DELETE\\_CDS\\_CC](#)

Definition at line 132 of file default\_cfe\_tbl\_msgdefs.h.

### 11.255.2 Field Documentation

**11.255.2.1 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 134 of file default\_cfe\_tbl\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h

## 11.256 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.*

### 11.256.1 Detailed Description

Delete Critical Table CDS Command.

Definition at line 115 of file default\_cfe\_tbl\_msgstruct.h.

## 11.256.2 Field Documentation

**11.256.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_DeleteCDSCmd::CommandHeader  
Command header.

Definition at line 117 of file default\_cfe\_tbl\_msgstruct.h.

**11.256.2.2 Payload** [CFE\\_TBL\\_DelCDSCmd\\_Payload\\_t](#) CFE\_TBL\_DeleteCDSCmd::Payload  
Command payload.

Definition at line 118 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

## 11.257 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.*

### 11.257.1 Detailed Description

/brief Dump Table Command

Definition at line 70 of file default\_cfe\_tbl\_msgstruct.h.

### 11.257.2 Field Documentation

**11.257.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_DumpCmd::CommandHeader  
Command header.

Definition at line 72 of file default\_cfe\_tbl\_msgstruct.h.

**11.257.2.2 Payload** [CFE\\_TBL\\_DumpCmd\\_Payload\\_t](#) CFE\_TBL\_DumpCmd::Payload  
Command payload.

Definition at line 73 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

## 11.258 CFE\_TBL\_DumpCmd\_Payload Struct Reference

Dump Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

## Data Fields

- **CFE\_TBL\_BufferSelect\_Enum\_t ActiveTableFlag**  
*CFE\_TBL\_BufferSelect\_INACTIVE=Inactive Table, CFE\_TBL\_BufferSelect\_ACTIVE=Active Table*
- char **TableName** [CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN]  
*Full name of table to be dumped.*
- char **DumpFilename** [CFE\_MISSION\_MAX\_PATH\_LEN]  
*Full Filename where data is to be written.*

### 11.258.1 Detailed Description

Dump Table Command Payload.

For command details, see [CFE\\_TBL\\_DUMP\\_CC](#)

Definition at line 54 of file default\_cfe\_tbl\_msgdefs.h.

### 11.258.2 Field Documentation

#### 11.258.2.1 ActiveTableFlag [CFE\\_TBL\\_BufferSelect\\_Enum\\_t](#) CFE\_TBL\_DumpCmd\_Payload::ActiveTableFlag

**CFE\_TBL\_BufferSelect\_INACTIVE**=Inactive Table, **CFE\_TBL\_BufferSelect\_ACTIVE**=Active Table

Selects either the "Inactive" ([CFE\\_TBL\\_BufferSelect\\_INACTIVE](#)) buffer or the "Active" ([CFE\\_TBL\\_BufferSelect\\_ACTIVE](#)) buffer to be dumped

Definition at line 56 of file default\_cfe\_tbl\_msgdefs.h.

#### 11.258.2.2 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 65 of file default\_cfe\_tbl\_msgdefs.h.

#### 11.258.2.3 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 62 of file default\_cfe\_tbl\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h

## 11.259 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.*

### 11.259.1 Detailed Description

Dump Registry Command.

Definition at line 97 of file default\_cfe\_tbl\_msgstruct.h.

### 11.259.2 Field Documentation

#### 11.259.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_DumpRegistryCmd::CommandHeader

Command header.

Definition at line 99 of file default\_cfe\_tbl\_msgstruct.h.

#### 11.259.2.2 Payload [CFE\\_TBL\\_DumpRegistryCmd\\_Payload\\_t](#) CFE\_TBL\_DumpRegistryCmd::Payload

Command payload.

Definition at line 100 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.260 CFE\_TBL\_DumpRegistryCmd\_Payload Struct Reference

Dump Registry Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

### Data Fields

- char [DumpFilename \[CFE\\_MISSION\\_MAX\\_PATH\\_LEN\]](#)

*Full Filename where dumped data is to be written.*

### 11.260.1 Detailed Description

Dump Registry Command Payload.

For command details, see [CFE\\_TBL\\_DUMP\\_REGISTRY\\_CC](#)

Definition at line 105 of file default\_cfe\_tbl\_msgdefs.h.

### 11.260.2 Field Documentation

#### 11.260.2.1 DumpFilename char CFE\_TBL\_DumpRegistryCmd\_Payload::DumpFilename [[CFE\\_MISSION\\_MAX\\_PATH\\_LEN](#)]

Full Filenname where dumped data is to be written.

ASCII string containing full path of filename where registry is to be dumped

Definition at line 107 of file default\_cfe\_tbl\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h

## 11.261 CFE\_TBL\_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]`

### 11.261.1 Detailed Description

The definition of the header fields that are included in CFE Table Data files. This header follows the CFE\_FS header and precedes the actual table data.

#### Note

The Offset and NumBytes fields in the table header are 32 bits for backward compatibility with existing CFE versions. This means that even on 64-bit CPUs, individual table files will be limited to 4GiB in size.

Definition at line 65 of file default\_cfe\_tbl\_extern\_typedefs.h.

### 11.261.2 Field Documentation

#### 11.261.2.1 NumBytes `uint32 CFE_TBL_File_Hdr::NumBytes`

Number of bytes to load into table

Definition at line 69 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 11.261.2.2 Offset `uint32 CFE_TBL_File_Hdr::Offset`

Byte Offset at which load should commence

Definition at line 68 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 11.261.2.3 Reserved `uint32 CFE_TBL_File_Hdr::Reserved`

Future Use: NumTblSegments in File?

Definition at line 67 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 11.261.2.4 TableName `char CFE_TBL_File_Hdr::TableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`

Fully qualified name of table to load

Definition at line 70 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

## 11.262 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.*

### 11.262.1 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`.

### 11.262.2 Field Documentation

#### 11.262.2.1 **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`.

#### 11.262.2.2 **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`.

#### 11.262.2.3 **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`.

**11.262.2.4 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`.

**11.262.2.5 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](#)

**11.263 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.*

**11.263.1 Detailed Description**

Definition at line 147 of file `default_cfe_tbl_msgstruct.h`.

**11.263.2 Field Documentation****11.263.2.1 Payload** `CFE_TBL_HousekeepingTlm_Payload_t CFE_TBL_HousekeepingTlm::Payload`

Telemetry payload.

Definition at line 150 of file `default_cfe_tbl_msgstruct.h`.

**11.263.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t CFE_TBL_HousekeepingTlm::TelemetryHeader`

Telemetry header.

Definition at line 149 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](#)

**11.264 CFE\_TBL\_HousekeepingTlm\_Payload Struct Reference**

```
#include <default_cfe_tbl_msgdefs.h>
```

## Data Fields

- `uint8 CommandCounter`  
*Count of valid commands received.*
- `uint8 CommandErrorCounter`  
*Count of invalid commands received.*
- `uint16 NumTables`  
*Number of Tables Registered.*
- `uint16 NumLoadPending`  
*Number of Tables pending on Applications for their update.*
- `uint16 ValidationCounter`  
*Number of completed table validations.*
- `uint32 LastValCrc`  
*Data Integrity Value computed for last table validated.*
- `int32 LastValStatus`  
*Returned status from validation function for last table validated.*
- `bool ActiveBuffer`  
*Indicator of whether table buffer validated was 0=Inactive, 1=Active.*
- `char LastValTableName [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`  
*Name of last table validated.*
- `uint8 SuccessValCounter`  
*Total number of successful table validations.*
- `uint8 FailedValCounter`  
*Total number of unsuccessful table validations.*
- `uint8 NumValRequests`  
*Number of times Table Services has requested validations from Apps.*
- `uint8 NumFreeSharedBufs`  
*Number of free Shared Working Buffers.*
- `uint8 ByteAlignPad1`  
*Spare byte to ensure longword alignment.*
- `CFE_ES_MemHandle_t MemPoolHandle`  
*Handle to TBL's memory pool.*
- `CFE_TIME_SysTime_t LastUpdateTime`  
*Time of last table update.*
- `char LastUpdatedTable [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`  
*Name of the last table updated.*
- `char LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]`  
*Path and Name of last table image file loaded.*
- `char LastFileDumped [CFE_MISSION_MAX_PATH_LEN]`  
*Path and Name of last file dumped to.*
- `char LastTableLoaded [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`  
*Name of the last table loaded.*

### 11.264.1 Detailed Description

**Name** Table Services Housekeeping Packet

Definition at line 177 of file default\_cfe\_tbl\_msgdefs.h.

## 11.264.2 Field Documentation

**11.264.2.1 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 204 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.2 ByteAlignPad1** `uint8 CFE_TBL_HousekeepingTlm_Payload::ByteAlignPad1`  
Spare byte to ensure longword alignment.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_ByteAlignPad1

Definition at line 220 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.3 CommandCounter** `uint8 CFE_TBL_HousekeepingTlm_Payload::CommandCounter`  
Count of valid commands received.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_CMDPC

Definition at line 182 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.4 CommandErrorCounter** `uint8 CFE_TBL_HousekeepingTlm_Payload::CommandErrorCounter`  
Count of invalid commands received.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_CMDEC

Definition at line 184 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.5 FailedValCounter** `uint8 CFE_TBL_HousekeepingTlm_Payload::FailedValCounter`  
Total number of unsuccessful table validations.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_ValFailedCtr

Definition at line 210 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.6 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 230 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.7 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 228 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.8 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 232 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.9 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 226 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.10 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 224 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.11 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 200 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.12 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 202 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.13 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 206 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.14 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 222 of file default\_cfe\_tbl\_msgdefs.h.

**11.264.2.15 NumFreeSharedBufs** `uint8` `CFE_TBL_HousekeepingTlm_Payload::NumFreeSharedBufs`  
Number of free Shared Working Buffers.

**Telemetry Mnemonic(s)** `$sc_$cpu_TBL_NumFreeShrBuf`

Definition at line 218 of file `default_cfe_tbl_msgdefs.h`.

**11.264.2.16 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 192 of file `default_cfe_tbl_msgdefs.h`.

**11.264.2.17 NumTables** `uint16` `CFE_TBL_HousekeepingTlm_Payload::NumTables`  
Number of Tables Registered.

**Telemetry Mnemonic(s)** `$sc_$cpu_TBL_NumTables`

Definition at line 190 of file `default_cfe_tbl_msgdefs.h`.

**11.264.2.18 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 212 of file `default_cfe_tbl_msgdefs.h`.

**11.264.2.19 SuccessValCounter** `uint8` `CFE_TBL_HousekeepingTlm_Payload::SuccessValCounter`  
Total number of successful table validations.

**Telemetry Mnemonic(s)** `$sc_$cpu_TBL_ValSuccessCtr`

Definition at line 208 of file `default_cfe_tbl_msgdefs.h`.

**11.264.2.20 ValidationCounter** `uint16` `CFE_TBL_HousekeepingTlm_Payload::ValidationCounter`  
Number of completed table validations.

**Telemetry Mnemonic(s)** `$sc_$cpu_TBL_ValCompltdCtr`

Definition at line 198 of file `default_cfe_tbl_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h`

## 11.265 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.*
- `CFE_TIME_SysTime_t FileTime`  
*File creation time from last file loaded into table.*
- `uint32 Crc`  
*Most recently calculated CRC by TBL services on table contents.*
- `CFE_TIME_SysTime_t TimeOfLastUpdate`  
*Time when Table was last updated.*
- `bool TableLoadedOnce`  
*Flag indicating whether table has been loaded once or not.*
- `bool DumpOnly`  
*Flag indicating Table is NOT to be loaded.*
- `bool DoubleBuffered`  
*Flag indicating Table has a dedicated inactive buffer.*
- `bool UserDefAddr`  
*Flag indicating Table address was defined by Owner Application.*
- `bool Critical`  
*Flag indicating Table contents are maintained in a CDS.*
- `char LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]`  
*Filename of last file loaded into table.*

### 11.265.1 Detailed Description

Table Info.

Definition at line 104 of file cfe\_tbl\_api\_typedefs.h.

### 11.265.2 Field Documentation

#### 11.265.2.1 `Crc uint32 CFE_TBL_Info::Crc`

Most recently calculated CRC by TBL services on table contents.

Definition at line 109 of file cfe\_tbl\_api\_typedefs.h.

#### 11.265.2.2 `Critical bool CFE_TBL_Info::Critical`

Flag indicating Table contents are maintained in a CDS.

Definition at line 115 of file cfe\_tbl\_api\_typedefs.h.

#### 11.265.2.3 `DoubleBuffered bool CFE_TBL_Info::DoubleBuffered`

Flag indicating Table has a dedicated inactive buffer.

Definition at line 113 of file cfe\_tbl\_api\_typedefs.h.

**11.265.2.4 DumpOnly** `bool CFE_TBL_Info::DumpOnly`  
Flag indicating Table is NOT to be loaded.  
Definition at line 112 of file `cfe_tbl_api_typedefs.h`.

**11.265.2.5 FileTime** `CFE_TIME_SysTime_t CFE_TBL_Info::FileTime`  
File creation time from last file loaded into table.  
Definition at line 108 of file `cfe_tbl_api_typedefs.h`.

**11.265.2.6 LastFileLoaded** `char CFE_TBL_Info::LastFileLoaded[CFE_MISSION_MAX_PATH_LEN]`  
Filename of last file loaded into table.  
Definition at line 116 of file `cfe_tbl_api_typedefs.h`.

**11.265.2.7 NumUsers** `uint32 CFE_TBL_Info::NumUsers`  
Number of Apps with access to the table.  
Definition at line 107 of file `cfe_tbl_api_typedefs.h`.

**11.265.2.8 Size** `size_t CFE_TBL_Info::Size`  
Size, in bytes, of Table.  
Definition at line 106 of file `cfe_tbl_api_typedefs.h`.

**11.265.2.9 TableLoadedOnce** `bool CFE_TBL_Info::TableLoadedOnce`  
Flag indicating whether table has been loaded once or not.  
Definition at line 111 of file `cfe_tbl_api_typedefs.h`.

**11.265.2.10 TimeOfLastUpdate** `CFE_TIME_SysTime_t CFE_TBL_Info::TimeOfLastUpdate`  
Time when Table was last updated.  
Definition at line 110 of file `cfe_tbl_api_typedefs.h`.

**11.265.2.11 UserDefAddr** `bool CFE_TBL_Info::UserDefAddr`  
Flag indicating Table address was defined by Owner Application.  
Definition at line 114 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`

## 11.266 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.*

### 11.266.1 Detailed Description

Load Table Command.

Definition at line 61 of file default\_cfe\_tbl\_msgstruct.h.

### 11.266.2 Field Documentation

#### 11.266.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_LoadCmd::CommandHeader

Command header.

Definition at line 63 of file default\_cfe\_tbl\_msgstruct.h.

#### 11.266.2.2 Payload [CFE\\_TBL\\_LoadCmd\\_Payload\\_t](#) CFE\_TBL\_LoadCmd::Payload

Command payload.

Definition at line 64 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

## 11.267 CFE\_TBL\_LoadCmd\_Payload Struct Reference

Load Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

### Data Fields

- char [LoadFilename](#) [[CFE\\_MISSION\\_MAX\\_PATH\\_LEN](#)]

*Filename (and path) of data to be loaded.*

### 11.267.1 Detailed Description

Load Table Command Payload.

For command details, see [CFE\\_TBL\\_LOAD\\_CC](#)

Definition at line 44 of file default\_cfe\_tbl\_msgdefs.h.

### 11.267.2 Field Documentation

#### 11.267.2.1 LoadFilename char CFE\_TBL\_LoadCmd\_Payload::LoadFilename[[CFE\\_MISSION\\_MAX\\_PATH\\_LEN](#)]

Filename (and path) of data to be loaded.

Definition at line 46 of file default\_cfe\_tbl\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h

## 11.268 CFE\_TBL\_NoopCmd Struct Reference

```
#include <default_cfe_tbl_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader

*Command header.*

### 11.268.1 Detailed Description

Definition at line 43 of file default\_cfe\_tbl\_msgstruct.h.

### 11.268.2 Field Documentation

**11.268.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_NoopCmd::CommandHeader  
Command header.

Definition at line 45 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h

## 11.269 CFE\_TBL\_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.*

### 11.269.1 Detailed Description

/brief Table Management Notification Command

Definition at line 135 of file default\_cfe\_tbl\_msgstruct.h.

### 11.269.2 Field Documentation

**11.269.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_NotifyCmd::CommandHeader  
Command header.

Definition at line 137 of file default\_cfe\_tbl\_msgstruct.h.

**11.269.2.2 Payload** [CFE\\_TBL\\_NotifyCmd\\_Payload\\_t](#) CFE\_TBL\_NotifyCmd::Payload  
Command payload.

Definition at line 138 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

## 11.270 CFE\_TBL\_NotifyCmd\_Payload Struct Reference

Table Management Notification Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

**Data Fields**

- `uint32 Parameter`

*Application specified command parameter.*

**11.270.1 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 164 of file `default_cfe_tbl_msgdefs.h`.

**11.270.2 Field Documentation****11.270.2.1 Parameter `uint32 CFE_TBL_NotifyCmd_Payload::Parameter`**

Application specified command parameter.

Definition at line 166 of file `default_cfe_tbl_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h`

**11.271 CFE\_TBL\_ResetCountersCmd Struct Reference**

```
#include <default_cfe_tbl_msgstruct.h>
```

**Data Fields**

- `CFE_MSG_CommandHeader_t CommandHeader`

*Command header.*

**11.271.1 Detailed Description**

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

**11.271.2 Field Documentation****11.271.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_TBL_ResetCountersCmd::CommandHeader`**

Command header.

Definition at line 50 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

**11.272 CFE\_TBL\_SendHkCmd Struct Reference**

```
#include <default_cfe_tbl_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*

**11.272.1 Detailed Description**

Definition at line 53 of file default\_cfe\_tbl\_msgstruct.h.

**11.272.2 Field Documentation****11.272.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_SendHkCmd::CommandHeader  
Command header.

Definition at line 55 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](#)

**11.273 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.*

**11.273.1 Detailed Description**

Send Table Registry Command.

Definition at line 106 of file default\_cfe\_tbl\_msgstruct.h.

**11.273.2 Field Documentation****11.273.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TBL\_SendRegistryCmd::CommandHeader  
Command header.

Definition at line 108 of file default\_cfe\_tbl\_msgstruct.h.

**11.273.2.2 Payload** [CFE\\_TBL\\_SendRegistryCmd\\_Payload\\_t](#) CFE\_TBL\_SendRegistryCmd::Payload  
Command payload.

Definition at line 109 of file default\_cfe\_tbl\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/[default\\_cfe\\_tbl\\_msgstruct.h](#)

## 11.274 CFE\_TBL\_SendRegistryCmd\_Payload Struct Reference

Send Table Registry Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

### Data Fields

- char **TableName** [CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN]

*Full Name of Table whose registry entry is to be telemetered.*

### 11.274.1 Detailed Description

Send Table Registry Command Payload.

For command details, see [CFE\\_TBL\\_SEND\\_REGISTRY\\_CC](#)

Definition at line 118 of file default\_cfe\_tbl\_msgdefs.h.

### 11.274.2 Field Documentation

#### 11.274.2.1 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 120 of file default\_cfe\_tbl\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h

## 11.275 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.*

### 11.275.1 Detailed Description

Definition at line 153 of file default\_cfe\_tbl\_msgstruct.h.

### 11.275.2 Field Documentation

#### 11.275.2.1 Payload [CFE\\_TBL\\_TblRegPacket\\_Payload\\_t](#) CFE\_TBL\_TableRegistryTlm::Payload

Telemetry payload.

Definition at line 156 of file default\_cfe\_tbl\_msgstruct.h.

**11.275.2.2 TelemetryHeader** `CFE_MSG_TelemetryHeader_t` `CFE_TBL_TableRegistryTlm::TelemetryHeader`  
Telemetry header.

Definition at line 155 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

## 11.276 CFE\_TBL\_TblRegPacket\_Payload Struct Reference

```
#include <default_cfe_tbl_msgdefs.h>
```

### Data Fields

- `CFE_ES_MemOffset_t Size`  
*Size, in bytes, of Table.*
- `uint32 Crc`  
*Most recently calculated CRC of Table.*
- `CFE_ES_MemAddress_t ActiveBufferAddr`  
*Address of Active Buffer.*
- `CFE_ES_MemAddress_t InactiveBufferAddr`  
*Address of Inactive Buffer.*
- `CFE_ES_MemAddress_t ValidationFuncPtr`  
*Ptr to Owner App's function that validates tbl contents.*
- `CFE_TIME_SysTime_t TimeOfLastUpdate`  
*Time when Table was last updated.*
- `CFE_TIME_SysTime_t FileTime`  
*File creation time from last file loaded into table.*
- `bool TableLoadedOnce`  
*Flag indicating whether table has been loaded once or not.*
- `bool LoadPending`  
*Flag indicating an inactive buffer is ready to be copied.*
- `bool DumpOnly`  
*Flag indicating Table is NOT to be loaded.*
- `bool DoubleBuffered`  
*Flag indicating Table has a dedicated inactive buffer.*
- `char Name [CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`  
*Processor specific table name.*
- `char LastFileLoaded [CFE_MISSION_MAX_PATH_LEN]`  
*Filename of last file loaded into table.*
- `char OwnerAppName [CFE_MISSION_MAX_API_LEN]`  
*Name of owning application.*
- `bool Critical`  
*Indicates whether table is Critical or not.*
- `uint8 ByteAlign4`  
*Spare byte to maintain byte alignment.*

### 11.276.1 Detailed Description

**Name** Table Registry Info Packet

Definition at line 239 of file `default_cfe_tbl_msgdefs.h`.

## 11.276.2 Field Documentation

**11.276.2.1 ActiveBufferAddr** `CFE_ES_MemAddress_t CFE_TBL_TblRegPacket_Payload::ActiveBufferAddr`  
Address of Active Buffer.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_ActBufAdd

Definition at line 245 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.2 ByteAlign4** `uint8 CFE_TBL_TblRegPacket_Payload::ByteAlign4`  
Spare byte to maintain byte alignment.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_Spare4

Definition at line 271 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.3 Crc** `uint32 CFE_TBL_TblRegPacket_Payload::Crc`  
Most recently calculated CRC of Table.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_CRC

Definition at line 243 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.4 Critical** `bool CFE_TBL_TblRegPacket_Payload::Critical`  
Indicates whether table is Critical or not.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_Spare3

Definition at line 269 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.5 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 261 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.6 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 259 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.7 FileTime** `CFE_TIME_SysTime_t CFE_TBL_TblRegPacket_Payload::FileTime`  
File creation time from last file loaded into table.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_FILECTIME

Definition at line 253 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.8 InactiveBufferAddr** `CFE_ES_MemAddress_t` `CFE_TBL_TblRegPacket_Payload::InactiveBufferAddr`  
Address of Inactive Buffer.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_IActBufAdd

Definition at line 247 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.9 LastFileLoaded** `char` `CFE_TBL_TblRegPacket_Payload::LastFileLoaded[CFE_MISSION_MAX_PATH_LEN]`  
Filename of last file loaded into table.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_LastFileUpd[OS\_MAX\_PATH\_LEN]

Definition at line 265 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.10 LoadPending** `bool` `CFE_TBL_TblRegPacket_Payload::LoadPending`  
Flag indicating an inactive buffer is ready to be copied.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_UpdatePndng

Definition at line 257 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.11 Name** `char` `CFE_TBL_TblRegPacket_Payload::Name[CFE_MISSION_TBL_MAX_FULL_NAME_LEN]`  
Processor specific table name.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_Name[CFE\_TB\_MAX\_FULL\_NAME\_LEN]

Definition at line 263 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.12 OwnerAppName** `char` `CFE_TBL_TblRegPacket_Payload::OwnerAppName[CFE_MISSION_MAX_API_LEN]`  
Name of owning application.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_OwnerApp[OS\_MAX\_API\_NAME]

Definition at line 267 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.13 Size** `CFE_ES_MemOffset_t` `CFE_TBL_TblRegPacket_Payload::Size`  
Size, in bytes, of Table.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_SIZE

Definition at line 241 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.14 TableLoadedOnce** `bool` `CFE_TBL_TblRegPacket_Payload::TableLoadedOnce`  
Flag indicating whether table has been loaded once or not.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TBL\_LoadedOnce

Definition at line 255 of file default\_cfe\_tbl\_msgdefs.h.

**11.276.2.15 TimeOfLastUpdate** `CFE_TIME_SysTime_t` `CFE_TBL_TblRegPacket_Payload::TimeOfLastUpdate`  
Time when Table was last updated.

**Telemetry Mnemonic(s)** `$sc_$cpu_TBL_TimeLastUpd`, `$sc_$cpu_TBL_TLUSECONDS`, `$sc_$cpu_TBL_TLUSUBSECONDS`

Definition at line 251 of file `default_cfe_tbl_msgdefs.h`.

**11.276.2.16 ValidationFuncPtr** `CFE_ES_MemAddress_t` `CFE_TBL_TblRegPacket_Payload::ValidationFuncPtr`  
Ptr to Owner App's function that validates tbl contents.

**Telemetry Mnemonic(s)** `$sc_$cpu_TBL_ValFuncPtr`

Definition at line 249 of file `default_cfe_tbl_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgdefs.h`

## 11.277 CFE\_TBL\_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.*

### 11.277.1 Detailed Description

Validate Table Command.

Definition at line 79 of file `default_cfe_tbl_msgstruct.h`.

### 11.277.2 Field Documentation

**11.277.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_TBL_ValidateCmd::CommandHeader`  
Command header.

Definition at line 81 of file `default_cfe_tbl_msgstruct.h`.

**11.277.2.2 Payload** `CFE_TBL_ValidateCmd_Payload_t` `CFE_TBL_ValidateCmd::Payload`  
Command payload.

Definition at line 82 of file `default_cfe_tbl_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/tbl/config/default_cfe_tbl_msgstruct.h`

## 11.278 CFE\_TBL\_ValidateCmd\_Payload Struct Reference

Validate Table Command Payload.

```
#include <default_cfe_tbl_msgdefs.h>
```

**Data Fields**

- [CFE\\_TBL\\_BufferSelect\\_Enum\\_t](#) ActiveTableFlag
  - CFE\_TBL\_BufferSelect\_INACTIVE=Inactive Table, CFE\_TBL\_BufferSelect\_ACTIVE=Active Table*
- char [TableName \[CFE\\_MISSION\\_TBL\\_MAX\\_FULL\\_NAME\\_LEN\]](#)  
*Full Name of Table to be validated.*

**11.278.1 Detailed Description**

Validate Table Command Payload.

For command details, see [CFE\\_TBL\\_VALIDATE\\_CC](#)

Definition at line 75 of file default\_cfe\_tbl\_msgdefs.h.

**11.278.2 Field Documentation**

**11.278.2.1 ActiveTableFlag** [CFE\\_TBL\\_BufferSelect\\_Enum\\_t](#) CFE\_TBL\_ValidateCmd\_Payload::ActiveTable↔Flag  
[CFE\\_TBL\\_BufferSelect\\_INACTIVE](#)=Inactive Table, [CFE\\_TBL\\_BufferSelect\\_ACTIVE](#)=Active Table  
Selects either the "Inactive" ([CFE\\_TBL\\_BufferSelect\\_INACTIVE](#)) buffer or the "Active" ([CFE\\_TBL\\_BufferSelect\\_ACTIVE](#)) buffer to be validated  
Definition at line 77 of file default\_cfe\_tbl\_msgdefs.h.

**11.278.2.2 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 83 of file default\_cfe\_tbl\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h

**11.279 CFE\_TIME\_AddAdjustCmd Struct Reference**

```
#include <default_cfe_time_msgstruct.h>
```

**Data Fields**

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader
  - Command header.*
- [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) Payload
  - Command payload.*

**11.279.1 Detailed Description**

Definition at line 137 of file default\_cfe\_time\_msgstruct.h.

**11.279.2 Field Documentation**

**11.279.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_AddAdjustCmd::CommandHeader  
Command header.  
Definition at line 139 of file default\_cfe\_time\_msgstruct.h.

**11.279.2.2 Payload** [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) CFE\_TIME\_AddAdjustCmd::Payload  
Command payload.  
Definition at line 140 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

## 11.280 CFE\_TIME\_AddDelayCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.280.1 Detailed Description

Definition at line 113 of file default\_cfe\_time\_msgstruct.h.

### 11.280.2 Field Documentation

**11.280.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_AddDelayCmd::CommandHeader  
Command header.  
Definition at line 115 of file default\_cfe\_time\_msgstruct.h.

**11.280.2.2 Payload** [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) CFE\_TIME\_AddDelayCmd::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

## 11.281 CFE\_TIME\_AddOneHzAdjustmentCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_OneHzAdjustmentCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.281.1 Detailed Description

Definition at line 160 of file default\_cfe\_time\_msgstruct.h.

### 11.281.2 Field Documentation

#### 11.281.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_AddOneHzAdjustmentCmd::CommandHeader

Command header.

Definition at line 162 of file default\_cfe\_time\_msgstruct.h.

#### 11.281.2.2 Payload [CFE\\_TIME\\_OneHzAdjustmentCmd\\_Payload\\_t](#) CFE\_TIME\_AddOneHzAdjustmentCmd::Payload

Command payload.

Definition at line 163 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/[default\\_cfe\\_time\\_msgstruct.h](#)

## 11.282 CFE\_TIME\_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.*

### 11.282.1 Detailed Description

Definition at line 189 of file default\_cfe\_time\_msgstruct.h.

### 11.282.2 Field Documentation

#### 11.282.2.1 Payload [CFE\\_TIME\\_DiagnosticTlm\\_Payload\\_t](#) CFE\_TIME\_DiagnosticTlm::Payload

Telemetry payload.

Definition at line 192 of file default\_cfe\_time\_msgstruct.h.

#### 11.282.2.2 TelemetryHeader [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_TIME\_DiagnosticTlm::TelemetryHeader

Telemetry header.

Definition at line 191 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/[default\\_cfe\\_time\\_msgstruct.h](#)

## 11.283 CFE\_TIME\_DiagnosticTlm\_Payload Struct Reference

```
#include <default_cfe_time_msgdefs.h>
```

## Data Fields

- [CFE\\_TIME\\_SysTime\\_t AtToneMET](#)  
*MET at time of tone.*
- [CFE\\_TIME\\_SysTime\\_t AtToneSTCF](#)  
*STCF at time of tone.*
- [CFE\\_TIME\\_SysTime\\_t AtToneDelay](#)  
*Adjustment for slow tone detection.*
- [CFE\\_TIME\\_SysTime\\_t AtToneLatch](#)  
*Local clock latched at time of tone.*
- [int16 AtToneLeapSeconds](#)  
*Leap Seconds at time of tone.*
- [CFE\\_TIME\\_ClockState\\_Enum\\_t ClockStateAPI](#)  
*Clock state as per API.*
- [CFE\\_TIME\\_SysTime\\_t TimeSinceTone](#)  
*Time elapsed since the tone.*
- [CFE\\_TIME\\_SysTime\\_t CurrentLatch](#)  
*Local clock latched just "now".*
- [CFE\\_TIME\\_SysTime\\_t CurrentMET](#)  
*MET at this instant.*
- [CFE\\_TIME\\_SysTime\\_t CurrentTAI](#)  
*TAI at this instant.*
- [CFE\\_TIME\\_SysTime\\_t CurrentUTC](#)  
*UTC at this instant.*
- [int16 ClockSetState](#)  
*Time has been "set".*
- [int16 ClockFlyState](#)  
*Current fly-wheel state.*
- [int16 ClockSource](#)  
*Internal vs external, etc.*
- [int16 ClockSignal](#)  
*Primary vs redundant, etc.*
- [int16 ServerFlyState](#)  
*Used by clients only.*
- [int16 Forced2Fly](#)  
*Commanded into fly-wheel.*
- [uint16 ClockStateFlags](#)  
*Clock State Flags.*
- [int16 OneTimeDirection](#)  
*One time STCF adjustment direction (Add = 1, Sub = 2)*
- [int16 OneHzDirection](#)  
*1Hz STCF adjustment direction*
- [int16 DelayDirection](#)  
*Client latency adjustment direction.*
- [CFE\\_TIME\\_SysTime\\_t OneTimeAdjust](#)  
*Previous one-time STCF adjustment.*
- [CFE\\_TIME\\_SysTime\\_t OneHzAdjust](#)  
*Current 1Hz STCF adjustment.*

- `CFE_TIME_SysTime_t ToneSignalLatch`  
*Local Clock latched at most recent tone signal.*
- `CFE_TIME_SysTime_t ToneDataLatch`  
*Local Clock latched at arrival of tone data.*
- `uint32 ToneMatchCounter`  
*Tone signal / data verification count.*
- `uint32 ToneMatchErrorCounter`  
*Tone signal / data verification error count.*
- `uint32 ToneSignalCounter`  
*Tone signal detected SB message count.*
- `uint32 ToneDataCounter`  
*Time at the tone data SB message count.*
- `uint32 ToneIntCounter`  
*Tone signal ISR execution count.*
- `uint32 ToneIntErrorCounter`  
*Tone signal ISR error count.*
- `uint32 ToneTaskCounter`  
*Tone task execution count.*
- `uint32 VersionCounter`  
*Count of mods to time at tone reference data (version)*
- `uint32 LocalIntCounter`  
*Local 1Hz ISR execution count.*
- `uint32 LocalTaskCounter`  
*Local 1Hz task execution count.*
- `uint32 VirtualMET`  
*Software MET.*
- `uint32 MinElapsed`  
*Min tone signal / data pkt arrival window (Sub-seconds)*
- `uint32 MaxElapsed`  
*Max tone signal / data pkt arrival window (Sub-seconds)*
- `CFE_TIME_SysTime_t MaxLocalClock`  
*Max local clock value before rollover.*
- `uint32 ToneOverLimit`  
*Max between tone signal interrupts.*
- `uint32 ToneUnderLimit`  
*Min between tone signal interrupts.*
- `uint32 DataStoreStatus`  
*Data Store status (preserved across processor reset)*

### 11.283.1 Detailed Description

**Name** Time Services Diagnostics Packet

Definition at line 175 of file default\_cfe\_time\_msgdefs.h.

### 11.283.2 Field Documentation

**11.283.2.1 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 184 of file `default_cfe_time_msgdefs.h`.

**11.283.2.2 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 186 of file `default_cfe_time_msgdefs.h`.

**11.283.2.3 AtToneLeapSeconds** `int16` `CFE_TIME_DiagnosticTlm_Payload::AtToneLeapSeconds`  
Leap Seconds at time of tone.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DLepS`

Definition at line 189 of file `default_cfe_time_msgdefs.h`.

**11.283.2.4 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 180 of file `default_cfe_time_msgdefs.h`.

**11.283.2.5 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 182 of file `default_cfe_time_msgdefs.h`.

**11.283.2.6 ClockFlyState** `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockFlyState`  
Current fly-wheel state.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DFlywheel`

Definition at line 213 of file `default_cfe_time_msgdefs.h`.

**11.283.2.7 ClockSetState** `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSetState`  
Time has been "set".

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DValid`

Definition at line 211 of file `default_cfe_time_msgdefs.h`.

**11.283.2.8 ClockSignal** `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSignal`  
Primary vs redundant, etc.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DSIGNAL`

Definition at line 217 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.9 ClockSource** `int16` `CFE_TIME_DiagnosticTlm_Payload::ClockSource`  
Internal vs external, etc.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DSOURCE`

Definition at line 215 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.10 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 191 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.11 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 227 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.12 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_DLCLSS`

Definition at line 199 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.13 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 201 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.14 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 203 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.15 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 205 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.16 DataStoreStatus** `uint32` `CFE_TIME_DiagnosticTlm_Payload::DataStoreStatus`  
Data Store status (preserved across processor reset)

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_DataStStat

Definition at line 317 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.17 DelayDirection** `int16` `CFE_TIME_DiagnosticTlm_Payload::DelayDirection`  
Client latency adjustment direction.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_DLlatentDir

Definition at line 237 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.18 Forced2Fly** `int16` `CFE_TIME_DiagnosticTlm_Payload::Forced2Fly`  
Commanded into fly-wheel.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_DCMD2Fly

Definition at line 221 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.19 LocalIntCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::LocalIntCounter`  
Local 1Hz ISR execution count.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_D1HzISRCNT

Definition at line 275 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.20 LocalTaskCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::LocalTaskCounter`  
Local 1Hz task execution count.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_D1HzTaskCNT

Definition at line 277 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.21 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 297 of file default\_cfe\_time\_msgdefs.h.

**11.283.2.22 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 303 of file `default_cfe_time_msgdefs.h`.

**11.283.2.23 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 295 of file `default_cfe_time_msgdefs.h`.

**11.283.2.24 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 245 of file `default_cfe_time_msgdefs.h`.

**11.283.2.25 OneHzDirection** `int16` `CFE_TIME_DiagnosticTlm_Payload::OneHzDirection`  
1Hz STCF adjustment direction

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_D1HzAdjDir`

Definition at line 235 of file `default_cfe_time_msgdefs.h`.

**11.283.2.26 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 243 of file `default_cfe_time_msgdefs.h`.

**11.283.2.27 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 233 of file `default_cfe_time_msgdefs.h`.

**11.283.2.28 ServerFlyState** `int16` `CFE_TIME_DiagnosticTlm_Payload::ServerFlyState`  
Used by clients only.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DSrvFly`

Definition at line 219 of file `default_cfe_time_msgdefs.h`.

**11.283.2.29 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 197 of file `default_cfe_time_msgdefs.h`.

**11.283.2.30 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 265 of file `default_cfe_time_msgdefs.h`.

**11.283.2.31 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 253 of file `default_cfe_time_msgdefs.h`.

**11.283.2.32 ToneIntCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneIntCounter`  
Tone signal ISR execution count.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DTsISRCNT`

Definition at line 267 of file `default_cfe_time_msgdefs.h`.

**11.283.2.33 ToneIntErrorCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneIntErrorCounter`  
Tone signal ISR error count.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DTsISRERR`

Definition at line 269 of file `default_cfe_time_msgdefs.h`.

**11.283.2.34 ToneMatchCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneMatchCounter`  
Tone signal / data verification count.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DVerifyCNT`

Definition at line 259 of file `default_cfe_time_msgdefs.h`.

**11.283.2.35 ToneMatchErrorCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneMatchErrorCounter`  
Tone signal / data verification error count.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DVerifyER`

Definition at line 261 of file `default_cfe_time_msgdefs.h`.

**11.283.2.36 ToneOverLimit** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneOverLimit`  
Max between tone signal interrupts.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DMaxSs`

Definition at line 309 of file `default_cfe_time_msgdefs.h`.

**11.283.2.37 ToneSignalCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneSignalCounter`  
Tone signal detected SB message count.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DTSDetCNT`

Definition at line 263 of file `default_cfe_time_msgdefs.h`.

**11.283.2.38 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 251 of file `default_cfe_time_msgdefs.h`.

**11.283.2.39 ToneTaskCounter** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneTaskCounter`  
Tone task execution count.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DTsTaskCNT`

Definition at line 271 of file `default_cfe_time_msgdefs.h`.

**11.283.2.40 ToneUnderLimit** `uint32` `CFE_TIME_DiagnosticTlm_Payload::ToneUnderLimit`  
Min between tone signal interrupts.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DMinSs`

Definition at line 311 of file `default_cfe_time_msgdefs.h`.

**11.283.2.41 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 273 of file `default_cfe_time_msgdefs.h`.

**11.283.2.42 VirtualMET** `uint32` `CFE_TIME_DiagnosticTlm_Payload::VirtualMET`  
Software MET.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_DLLogicalMET`

Definition at line 283 of file `default_cfe_time_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgdefs.h`

## 11.284 CFE\_TIME\_FakeToneCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader

*Command header.*

#### 11.284.1 Detailed Description

Definition at line 62 of file default\_cfe\_time\_msgstruct.h.

#### 11.284.2 Field Documentation

##### 11.284.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_FakeToneCmd::CommandHeader

Command header.

Definition at line 64 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.285 CFE\_TIME\_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.*

#### 11.285.1 Detailed Description

Definition at line 183 of file default\_cfe\_time\_msgstruct.h.

#### 11.285.2 Field Documentation

##### 11.285.2.1 Payload [CFE\\_TIME\\_HousekeepingTlm\\_Payload\\_t](#) CFE\_TIME\_HousekeepingTlm::Payload

Telemetry payload.

Definition at line 186 of file default\_cfe\_time\_msgstruct.h.

##### 11.285.2.2 TelemetryHeader [CFE\\_MSG\\_TelemetryHeader\\_t](#) CFE\_TIME\_HousekeepingTlm::TelemetryHeader

Telemetry header.

Definition at line 185 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

## 11.286 CFE\_TIME\_HousekeepingTlm\_Payload Struct Reference

```
#include <default_cfe_time_msgdefs.h>
```

### Data Fields

- `uint8 CommandCounter`  
*Time Command Execution Counter.*
- `uint8 CommandErrorCounter`  
*Time Command Error Counter.*
- `uint16 ClockStateFlags`  
*State Flags.*
- `CFE_TIME_ClockState_Enum_t ClockStateAPI`  
*API State.*
- `int16 LeapSeconds`  
*Current Leaps Seconds.*
- `CFE_TIME_SysTime_t MET`  
*Current MET (seconds+subseconds)*
- `CFE_TIME_SysTime_t STCF`  
*Current STCF (seconds+subseconds)*
- `CFE_TIME_SysTime_t AdjustmentFactor`  
*Current Adjustment Factor (seconds+subseconds)*

### 11.286.1 Detailed Description

**Name** Time Services Housekeeping Packet

Definition at line 125 of file default\_cfe\_time\_msgdefs.h.

### 11.286.2 Field Documentation

**11.286.2.1 AdjustmentFactor** `CFE_TIME_SysTime_t CFE_TIME_HousekeepingTlm_Payload::AdjustmentFactor`  
Current Adjustment Factor (seconds+subseconds)

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_ADJ

- In time server mode, the adjustment factor is to compensate for drift over time This is known as the "1Hz STCF adjustment value" and is applied continuously during server operation
- In time client mode, the adjustment factor is a compensation for transit delay from the server This is known as the "Time at tone delay" and is applied to every time at tone update

Definition at line 166 of file default\_cfe\_time\_msgdefs.h.

**11.286.2.2 ClockStateAPI** `CFE_TIME_ClockState_Enum_t CFE_TIME_HousekeepingTlm_Payload::ClockStateAPI`  
API State.

**Telemetry Mnemonic(s)** \$sc\_\$cpu\_TIME\_DAPIStrate

Definition at line 140 of file default\_cfe\_time\_msgdefs.h.

**11.286.2.3 ClockStateFlags** `uint16` `CFE_TIME_HousekeepingTlm_Payload::ClockStateFlags`  
State Flags.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_StateFlg`, `$sc_$cpu_TIME_FlagSet`, `$sc_$cpu_TIME_FlagFly`, `$sc_$cpu_TIME_FlagSrc`,  
`$sc_$cpu_TIME_FlagPri`, `$sc_$cpu_TIME_FlagSfly`, `$sc_$cpu_TIME_FlagCfly`, `$sc_$cpu_TIME_FlagAdjd`,  
`$sc_$cpu_TIME_Flag1Hzd`, `$sc_$cpu_TIME_FlagClat`, `$sc_$cpu_TIME_FlagSorC`, `$sc_$cpu_TIME_FlagNIU`

Definition at line 138 of file `default_cfe_time_msgdefs.h`.

**11.286.2.4 CommandCounter** `uint8` `CFE_TIME_HousekeepingTlm_Payload::CommandCounter`  
Time Command Execution Counter.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_CMDPC`

Definition at line 130 of file `default_cfe_time_msgdefs.h`.

**11.286.2.5 CommandErrorCounter** `uint8` `CFE_TIME_HousekeepingTlm_Payload::CommandErrorCounter`  
Time Command Error Counter.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_CMDEC`

Definition at line 132 of file `default_cfe_time_msgdefs.h`.

**11.286.2.6 LeapSeconds** `int16` `CFE_TIME_HousekeepingTlm_Payload::LeapSeconds`  
Current Leaps Seconds.

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_LeapSecs`

Definition at line 146 of file `default_cfe_time_msgdefs.h`.

**11.286.2.7 MET** `CFE_TIME_SysTime_t` `CFE_TIME_HousekeepingTlm_Payload::MET`  
Current MET (seconds+subseconds)

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_MET`

Definition at line 152 of file `default_cfe_time_msgdefs.h`.

**11.286.2.8 STCF** `CFE_TIME_SysTime_t` `CFE_TIME_HousekeepingTlm_Payload::STCF`  
Current STCF (seconds+subseconds)

**Telemetry Mnemonic(s)** `$sc_$cpu_TIME_STCF`

Definition at line 153 of file `default_cfe_time_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgdefs.h`

## 11.287 CFE\_TIME\_LeapsCmd\_Payload Struct Reference

Set leap seconds command payload.

```
#include <default_cfe_time_msgdefs.h>
```

**Data Fields**

- int16 LeapSeconds

**11.287.1 Detailed Description**

Set leap seconds command payload.

Definition at line 54 of file default\_cfe\_time\_msgdefs.h.

**11.287.2 Field Documentation****11.287.2.1 LeapSeconds int16 CFE\_TIME\_LeapsCmd\_Payload::LeapSeconds**

Definition at line 56 of file default\_cfe\_time\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgdefs.h

**11.288 CFE\_TIME\_NoopCmd Struct Reference**

```
#include <default_cfe_time_msgstruct.h>
```

**Data Fields**

- CFE\_MSG\_CommandHeader\_t CommandHeader  
*Command header.*

**11.288.1 Detailed Description**

Definition at line 37 of file default\_cfe\_time\_msgstruct.h.

**11.288.2 Field Documentation****11.288.2.1 CommandHeader CFE\_MSG\_CommandHeader\_t CFE\_TIME\_NoopCmd::CommandHeader**  
Command header.

Definition at line 39 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

**11.289 CFE\_TIME\_OneHzAdjustmentCmd\_Payload Struct Reference**

Generic seconds, subseconds command payload.

```
#include <default_cfe_time_msgdefs.h>
```

**Data Fields**

- uint32 Seconds
- uint32 Subseconds

### 11.289.1 Detailed Description

Generic seconds, subseconds command payload.

Definition at line 103 of file default\_cfe\_time\_msgdefs.h.

### 11.289.2 Field Documentation

#### 11.289.2.1 Seconds `uint32 CFE_TIME_OneHzAdjustmentCmd_Payload::Seconds`

Definition at line 105 of file default\_cfe\_time\_msgdefs.h.

#### 11.289.2.2 Subseconds `uint32 CFE_TIME_OneHzAdjustmentCmd_Payload::Subseconds`

Definition at line 106 of file default\_cfe\_time\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgdefs.h

## 11.290 CFE\_TIME\_OneHzCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

*Command header.*

### 11.290.1 Detailed Description

Definition at line 52 of file default\_cfe\_time\_msgstruct.h.

### 11.290.2 Field Documentation

#### 11.290.2.1 CommandHeader `CFE_MSG_CommandHeader_t CFE_TIME_OneHzCmd::CommandHeader`

Command header.

Definition at line 54 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.291 CFE\_TIME\_ResetCountersCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`

*Command header.*

### 11.291.1 Detailed Description

Definition at line 42 of file default\_cfe\_time\_msgstruct.h.

## 11.291.2 Field Documentation

**11.291.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_ResetCountersCmd::CommandHeader  
Command header.

Definition at line 44 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

## 11.292 CFE\_TIME\_SendDiagnosticCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*

### 11.292.1 Detailed Description

Definition at line 47 of file default\_cfe\_time\_msgstruct.h.

## 11.292.2 Field Documentation

**11.292.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SendDiagnosticCmd::CommandHeader  
Command header.

Definition at line 49 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.293 CFE\_TIME\_SendHkCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*

### 11.293.1 Detailed Description

Definition at line 67 of file default\_cfe\_time\_msgstruct.h.

## 11.293.2 Field Documentation

**11.293.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SendHkCmd::CommandHeader  
Command header.

Definition at line 69 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.294 CFE\_TIME\_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.*

### 11.294.1 Detailed Description

Set leap seconds command.

Definition at line 75 of file default\_cfe\_time\_msgstruct.h.

### 11.294.2 Field Documentation

**11.294.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetLeapSecondsCmd::CommandHeader  
Command header.

Definition at line 77 of file default\_cfe\_time\_msgstruct.h.

**11.294.2.2 Payload** [CFE\\_TIME\\_LeapsCmd\\_Payload\\_t](#) CFE\_TIME\_SetLeapSecondsCmd::Payload  
Command payload.

Definition at line 78 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.295 CFE\_TIME\_SetMETCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.295.1 Detailed Description

Definition at line 125 of file default\_cfe\_time\_msgstruct.h.

## 11.295.2 Field Documentation

**11.295.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetMETCmd::CommandHeader  
Command header.

Definition at line 127 of file default\_cfe\_time\_msgstruct.h.

**11.295.2.2 Payload** [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) CFE\_TIME\_SetMETCmd::Payload  
Command payload.

Definition at line 128 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.296 CFE\_TIME\_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.*

### 11.296.1 Detailed Description

Set tone signal source command.

Definition at line 102 of file default\_cfe\_time\_msgstruct.h.

## 11.296.2 Field Documentation

**11.296.2.1 CommandHeader** [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetSignalCmd::CommandHeader  
Command header.

Definition at line 104 of file default\_cfe\_time\_msgstruct.h.

**11.296.2.2 Payload** [CFE\\_TIME\\_SignalCmd\\_Payload\\_t](#) CFE\_TIME\_SetSignalCmd::Payload  
Command payload.

Definition at line 105 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.297 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.*

**11.297.1 Detailed Description**

Set time data source command.

Definition at line 93 of file default\_cfe\_time\_msgstruct.h.

**11.297.2 Field Documentation****11.297.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetSourceCmd::CommandHeader**  
Command header.

Definition at line 95 of file default\_cfe\_time\_msgstruct.h.

**11.297.2.2 Payload [CFE\\_TIME\\_SourceCmd\\_Payload\\_t](#) CFE\_TIME\_SetSourceCmd::Payload**  
Command payload.

Definition at line 96 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

**11.298 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.*

**11.298.1 Detailed Description**

Set clock state command.

Definition at line 84 of file default\_cfe\_time\_msgstruct.h.

**11.298.2 Field Documentation****11.298.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SetStateCmd::CommandHeader**  
Command header.

Definition at line 86 of file default\_cfe\_time\_msgstruct.h.

**11.298.2.2 Payload** `CFE_TIME_StateCmd_Payload_t` `CFE_TIME_SetStateCmd::Payload`  
Command payload.

Definition at line 87 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

## 11.299 CFE\_TIME\_SetSTCFCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_TIME_TimeCmd_Payload_t Payload`  
*Command payload.*

### 11.299.1 Detailed Description

Definition at line 131 of file default\_cfe\_time\_msgstruct.h.

### 11.299.2 Field Documentation

**11.299.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_TIME_SetSTCFCmd::CommandHeader`  
Command header.

Definition at line 133 of file default\_cfe\_time\_msgstruct.h.

**11.299.2.2 Payload** `CFE_TIME_TimeCmd_Payload_t` `CFE_TIME_SetSTCFCmd::Payload`  
Command payload.

Definition at line 134 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

## 11.300 CFE\_TIME\_SetTimeCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_TIME_TimeCmd_Payload_t Payload`  
*Command payload.*

### 11.300.1 Detailed Description

Definition at line 149 of file default\_cfe\_time\_msgstruct.h.

## 11.300.2 Field Documentation

**11.300.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_TIME_SetTimeCmd::CommandHeader`  
Command header.

Definition at line 151 of file default\_cfe\_time\_msgstruct.h.

**11.300.2.2 Payload** `CFE_TIME_TimeCmd_Payload_t` `CFE_TIME_SetTimeCmd::Payload`  
Command payload.

Definition at line 152 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.301 CFE\_TIME\_SignalCmd\_Payload Struct Reference

Set tone signal source command payload.

```
#include <default_cfe_time_msgdefs.h>
```

### Data Fields

- `int16 ToneSource`  
`CFE_TIME_ToneSignalSelect_PRIMARY=Primary Source, CFE_TIME_ToneSignalSelect_REDUNDANT=Redundant Source`

### 11.301.1 Detailed Description

Set tone signal source command payload.

Definition at line 84 of file default\_cfe\_time\_msgdefs.h.

## 11.301.2 Field Documentation

**11.301.2.1 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 86 of file default\_cfe\_time\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgdefs.h

## 11.302 CFE\_TIME\_SourceCmd\_Payload Struct Reference

Set time data source command payload.

```
#include <default_cfe_time_msgdefs.h>
```

**Data Fields**

- int16 TimeSource

*CFE\_TIME\_SourceSelect\_INTERNAL=Internal Source, CFE\_TIME\_SourceSelect\_EXTERNAL=External Source*

**11.302.1 Detailed Description**

Set time data source command payload.

Definition at line 74 of file default\_cfe\_time\_msgdefs.h.

**11.302.2 Field Documentation****11.302.2.1 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 76 of file default\_cfe\_time\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgdefs.h

**11.303 CFE\_TIME\_StateCmd\_Payload Struct Reference**

Set clock state command payload.

```
#include <default_cfe_time_msgdefs.h>
```

**Data Fields**

- CFE\_TIME\_ClockState\_Enum\_t ClockState

*CFE\_TIME\_ClockState\_INVALID=Spacecraft time has not been accurately set, CFE\_TIME\_ClockState\_VALID=Spacecraft clock has been accurately set, CFE\_TIME\_ClockState\_FLYWHEEL=Force into FLYWHEEL mode*

**11.303.1 Detailed Description**

Set clock state command payload.

Definition at line 62 of file default\_cfe\_time\_msgdefs.h.

**11.303.2 Field Documentation****11.303.2.1 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 64 of file default\_cfe\_time\_msgdefs.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgdefs.h

## 11.304 CFE\_TIME\_SubAdjustCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.304.1 Detailed Description

Definition at line 143 of file default\_cfe\_time\_msgstruct.h.

### 11.304.2 Field Documentation

#### 11.304.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SubAdjustCmd::CommandHeader

Command header.

Definition at line 145 of file default\_cfe\_time\_msgstruct.h.

#### 11.304.2.2 Payload [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) CFE\_TIME\_SubAdjustCmd::Payload

Command payload.

Definition at line 146 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

## 11.305 CFE\_TIME\_SubDelayCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- [CFE\\_MSG\\_CommandHeader\\_t](#) CommandHeader  
*Command header.*
- [CFE\\_TIME\\_TimeCmd\\_Payload\\_t](#) Payload  
*Command payload.*

### 11.305.1 Detailed Description

Definition at line 119 of file default\_cfe\_time\_msgstruct.h.

### 11.305.2 Field Documentation

#### 11.305.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_SubDelayCmd::CommandHeader

Command header.

Definition at line 121 of file default\_cfe\_time\_msgstruct.h.

**11.305.2.2 Payload** `CFE_TIME_TimeCmd_Payload_t` `CFE_TIME_SubDelayCmd::Payload`  
Command payload.

Definition at line 122 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`

## 11.306 CFE\_TIME\_SubOneHzAdjustmentCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*
- `CFE_TIME_OneHzAdjustmentCmd_Payload_t Payload`  
*Command payload.*

### 11.306.1 Detailed Description

Definition at line 166 of file default\_cfe\_time\_msgstruct.h.

### 11.306.2 Field Documentation

**11.306.2.1 CommandHeader** `CFE_MSG_CommandHeader_t` `CFE_TIME_SubOneHzAdjustmentCmd::CommandHeader`  
Command header.

Definition at line 168 of file default\_cfe\_time\_msgstruct.h.

**11.306.2.2 Payload** `CFE_TIME_OneHzAdjustmentCmd_Payload_t` `CFE_TIME_SubOneHzAdjustmentCmd::Payload`  
Command payload.

Definition at line 169 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

## 11.307 CFE\_TIME\_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)*

### 11.307.1 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`.

### 11.307.2 Field Documentation

#### 11.307.2.1 Seconds `uint32 CFE_TIME_SysTime::Seconds`

Number of seconds since epoch.

Definition at line 43 of file `default_cfe_time_extern_typedefs.h`.

#### 11.307.2.2 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`

## 11.308 CFE\_TIME\_TimeCmd\_Payload Struct Reference

Generic seconds, microseconds command payload.

```
#include <default_cfe_time_msgdefs.h>
```

#### Data Fields

- `uint32 Seconds`
- `uint32 MicroSeconds`

### 11.308.1 Detailed Description

Generic seconds, microseconds command payload.

Definition at line 94 of file `default_cfe_time_msgdefs.h`.

### 11.308.2 Field Documentation

#### 11.308.2.1 MicroSeconds `uint32 CFE_TIME_TimeCmd_Payload::MicroSeconds`

Definition at line 97 of file `default_cfe_time_msgdefs.h`.

#### 11.308.2.2 Seconds `uint32 CFE_TIME_TimeCmd_Payload::Seconds`

Definition at line 96 of file `default_cfe_time_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgdefs.h`

## 11.309 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.*

### 11.309.1 Detailed Description

Time at tone data command.

Definition at line 175 of file default\_cfe\_time\_msgstruct.h.

### 11.309.2 Field Documentation

#### 11.309.2.1 CommandHeader [CFE\\_MSG\\_CommandHeader\\_t](#) CFE\_TIME\_ToneDataCmd::CommandHeader

Command header.

Definition at line 177 of file default\_cfe\_time\_msgstruct.h.

#### 11.309.2.2 Payload [CFE\\_TIME\\_ToneDataCmd\\_Payload\\_t](#) CFE\_TIME\_ToneDataCmd::Payload

Command payload.

Definition at line 178 of file default\_cfe\_time\_msgstruct.h.

The documentation for this struct was generated from the following file:

- cfe/modules/time/config/default\_cfe\_time\_msgstruct.h

## 11.310 CFE\_TIME\_ToneDataCmd\_Payload Struct Reference

Time at tone data command payload.

```
#include <default_cfe_time_msgdefs.h>
```

### Data Fields

- [CFE\\_TIME\\_SysTime\\_t](#) AtToneMET  
*MET at time of tone.*
- [CFE\\_TIME\\_SysTime\\_t](#) AtToneSTCF  
*STCF at time of tone.*
- [int16](#) AtToneLeapSeconds  
*Leap Seconds at time of tone.*
- [CFE\\_TIME\\_ClockState\\_Enum\\_t](#) AtToneState  
*Clock state at time of tone.*

### 11.310.1 Detailed Description

Time at tone data command payload.

Definition at line 112 of file default\_cfe\_time\_msgdefs.h.

### 11.310.2 Field Documentation

**11.310.2.1 AtToneLeapSeconds** `int16 CFE_TIME_ToneDataCmd_Payload::AtToneLeapSeconds`  
Leap Seconds at time of tone.

Definition at line 116 of file `default_cfe_time_msgdefs.h`.

**11.310.2.2 AtToneMET** `CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneMET`  
MET at time of tone.

Definition at line 114 of file `default_cfe_time_msgdefs.h`.

**11.310.2.3 AtToneState** `CFE_TIME_ClockState_Enum_t CFE_TIME_ToneDataCmd_Payload::AtToneState`  
Clock state at time of tone.

Definition at line 117 of file `default_cfe_time_msgdefs.h`.

**11.310.2.4 AtToneSTCF** `CFE_TIME_SysTime_t CFE_TIME_ToneDataCmd_Payload::AtToneSTCF`  
STCF at time of tone.

Definition at line 115 of file `default_cfe_time_msgdefs.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgdefs.h`

## 11.311 CFE\_TIME\_ToneSignalCmd Struct Reference

```
#include <default_cfe_time_msgstruct.h>
```

### Data Fields

- `CFE_MSG_CommandHeader_t CommandHeader`  
*Command header.*

### 11.311.1 Detailed Description

Definition at line 57 of file `default_cfe_time_msgstruct.h`.

### 11.311.2 Field Documentation

**11.311.2.1 CommandHeader** `CFE_MSG_CommandHeader_t CFE_TIME_ToneSignalCmd::CommandHeader`  
Command header.

Definition at line 59 of file `default_cfe_time_msgstruct.h`.

The documentation for this struct was generated from the following file:

- `cfe/modules/time/config/default_cfe_time_msgstruct.h`

## 11.312 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`

**11.312.1 Detailed Description**

OSAL binary semaphore properties.

Definition at line 39 of file osapi-binsem.h.

**11.312.2 Field Documentation****11.312.2.1 creator `osal_id_t` `OS_bin_sem_prop_t::creator`**

Definition at line 42 of file osapi-binsem.h.

**11.312.2.2 name char `OS_bin_sem_prop_t::name[OS_MAX_API_NAME]`**

Definition at line 41 of file osapi-binsem.h.

**11.312.2.3 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`

**11.313 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`

**11.313.1 Detailed Description**

OSAL condition variable properties.

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

**11.313.2 Field Documentation****11.313.2.1 creator `osal_id_t` `OS_condvar_prop_t::creator`**

Definition at line 37 of file osapi-condvar.h.

**11.313.2.2 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](#)

**11.314 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](#)

**11.314.1 Detailed Description**

OSAL counting semaphore properties.

Definition at line 32 of file osapi-countsem.h.

**11.314.2 Field Documentation****11.314.2.1 creator** [osal\\_id\\_t](#) OS\_count\_sem\_prop\_t::creator

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

**11.314.2.2 name** char OS\_count\_sem\_prop\_t::name[OS\_MAX\_API\_NAME]

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

**11.314.2.3 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](#)

**11.315 os\_dirent\_t Struct Reference**

Directory entry.

```
#include <osapi-dir.h>
```

**Data Fields**

- char [FileName](#) [OS\_MAX\_FILE\_NAME]

**11.315.1 Detailed Description**

Directory entry.

Definition at line 32 of file osapi-dir.h.

### 11.315.2 Field Documentation

#### 11.315.2.1 FileName char os\_dirent\_t::FileName[OS\_MAX\_FILE\_NAME]

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

Referenced by CF\_CFDP\_ProcessPlaybackDirectory().

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-dir.h](#)

## 11.316 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\]](#)

#### 11.316.1 Detailed Description

An abstract structure capable of holding several OSAL IDs.

This is part of the select API and is manipulated using the related API calls. It should not be modified directly by applications.

Note: Math is to determine uint8 array size needed to represent single bit OS\_MAX\_NUM\_OPEN\_FILES objects, + 7 rounds up and 8 is the size of uint8.

### See also

[OS\\_SelectFdZero\(\)](#), [OS\\_SelectFdAdd\(\)](#), [OS\\_SelectFdClear\(\)](#), [OS\\_SelectFdIsSet\(\)](#)

Definition at line 44 of file osapi-select.h.

### 11.316.2 Field Documentation

#### 11.316.2.1 object\_ids uint8 OS\_FdSet::object\_ids[ (OS\_MAX\_NUM\_OPEN\_FILES+7) / 8 ]

Definition at line 46 of file osapi-select.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-select.h](#)

## 11.317 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](#)

### 11.317.1 Detailed Description

OSAL file properties.

Definition at line 49 of file osapi-file.h.

### 11.317.2 Field Documentation

#### 11.317.2.1 IsValid `uint8 OS_file_prop_t::IsValid`

Definition at line 53 of file osapi-file.h.

#### 11.317.2.2 Path `char OS_file_prop_t::Path[OS_MAX_PATH_LEN]`

Definition at line 51 of file osapi-file.h.

#### 11.317.2.3 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](#)

## 11.318 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.*

### 11.318.1 Detailed Description

OSAL file system info.

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

### 11.318.2 Field Documentation

#### 11.318.2.1 FreeFds `uint32 os_fsinfo_t::FreeFds`

Total number that are free.

Definition at line 38 of file osapi-filesystem.h.

**11.318.2.2 FreeVolumes** `uint32` `os_fsinfo_t::FreeVolumes`

Total number of volumes free.

Definition at line 40 of file osapi-filesystems.h.

**11.318.2.3 MaxFds** `uint32` `os_fsinfo_t::MaxFds`

Total number of file descriptors.

Definition at line 37 of file osapi-filesystems.h.

**11.318.2.4 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`

## 11.319 os\_fstat\_t Struct Reference

File system status.

```
#include <osapi-file.h>
```

### Data Fields

- `uint32 FileModeBits`
- `OS_time_t FileTime`
- `size_t FileSize`

#### 11.319.1 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.

#### 11.319.2 Field Documentation

**11.319.2.1 FileModeBits** `uint32` `os_fstat_t::FileModeBits`

Definition at line 66 of file osapi-file.h.

**11.319.2.2 FileSize** `size_t` `os_fstat_t::FileSize`

Definition at line 68 of file osapi-file.h.

**11.319.2.3 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`

## 11.320 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`

### 11.320.1 Detailed Description

OSAL heap properties.

See also

[OS\\_HeapGetInfo\(\)](#)

Definition at line 36 of file osapi-heap.h.

### 11.320.2 Field Documentation

#### 11.320.2.1 `free_blocks` `osal_blockcount_t OS_heap_prop_t::free_blocks`

Definition at line 39 of file osapi-heap.h.

#### 11.320.2.2 `free_bytes` `size_t OS_heap_prop_t::free_bytes`

Definition at line 38 of file osapi-heap.h.

#### 11.320.2.3 `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`

## 11.321 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`

### 11.321.1 Detailed Description

OSAL module address properties.

Definition at line 78 of file osapi-module.h.

### 11.321.2 Field Documentation

#### 11.321.2.1 bss\_address `cpuaddr` OS\_module\_address\_t::bss\_address

Definition at line 86 of file osapi-module.h.

#### 11.321.2.2 bss\_size `cpuaddr` OS\_module\_address\_t::bss\_size

Definition at line 87 of file osapi-module.h.

#### 11.321.2.3 code\_address `cpuaddr` OS\_module\_address\_t::code\_address

Definition at line 82 of file osapi-module.h.

#### 11.321.2.4 code\_size `cpuaddr` OS\_module\_address\_t::code\_size

Definition at line 83 of file osapi-module.h.

#### 11.321.2.5 data\_address `cpuaddr` OS\_module\_address\_t::data\_address

Definition at line 84 of file osapi-module.h.

#### 11.321.2.6 data\_size `cpuaddr` OS\_module\_address\_t::data\_size

Definition at line 85 of file osapi-module.h.

#### 11.321.2.7 flags `uint32` OS\_module\_address\_t::flags

Definition at line 81 of file osapi-module.h.

#### 11.321.2.8 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`

## 11.322 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`

### 11.322.1 Detailed Description

OSAL module properties.

Definition at line 91 of file osapi-module.h.

### 11.322.2 Field Documentation

#### 11.322.2.1 `addr` `OS_module_address_t` `OS_module_prop_t::addr`

Definition at line 97 of file osapi-module.h.

#### 11.322.2.2 `entry_point` `cpuaddr` `OS_module_prop_t::entry_point`

Definition at line 93 of file osapi-module.h.

#### 11.322.2.3 `filename` char `OS_module_prop_t::filename` [`OS_MAX_PATH_LEN`]

Definition at line 95 of file osapi-module.h.

#### 11.322.2.4 `host_module_id` `cpuaddr` `OS_module_prop_t::host_module_id`

Definition at line 94 of file osapi-module.h.

#### 11.322.2.5 `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`

## 11.323 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`

### 11.323.1 Detailed Description

OSAL mutex properties.

Definition at line 32 of file osapi-mutex.h.

### 11.323.2 Field Documentation

**11.323.2.1 creator** `osal_id_t` `OS_mut_sem_prop_t::creator`  
Definition at line 35 of file osapi-mutex.h.

**11.323.2.2 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`

## 11.324 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`

### 11.324.1 Detailed Description

OSAL queue properties.

Definition at line 32 of file osapi-queue.h.

### 11.324.2 Field Documentation

**11.324.2.1 creator** `osal_id_t` `OS_queue_prop_t::creator`  
Definition at line 35 of file osapi-queue.h.

**11.324.2.2 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`

## 11.325 OS\_rwlock\_prop\_t Struct Reference

OSAL rwlock properties.

```
#include <osapi-rwlock.h>
```

### Data Fields

- `char name [OS_MAX_API_NAME]`
- `osal_id_t creator`

### 11.325.1 Detailed Description

OSAL rwlock properties.

Definition at line 32 of file osapi-rwlock.h.

### 11.325.2 Field Documentation

#### 11.325.2.1 **creator** `osal_id_t OS_rwlock_prop_t::creator`

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

#### 11.325.2.2 **name** `char OS_rwlock_prop_t::name[OS_MAX_API_NAME]`

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

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-rwlock.h`

## 11.326 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.*

### 11.326.1 Detailed Description

Encapsulates a generic network address.

This is just an abstract buffer type that holds a network address. It is allocated for the worst-case size defined by `OS SOCKADDR MAX LEN`, and the real size is stored within.

Definition at line 110 of file osapi-sockets.h.

### 11.326.2 Field Documentation

#### 11.326.2.1 **ActualLength** `size_t OS_SockAddr_t::ActualLength`

Length of the actual address data.

Definition at line 112 of file osapi-sockets.h.

#### 11.326.2.2 **AddrData** `OS_SockAddrData_t OS_SockAddr_t::AddrData`

Abstract Address data.

Definition at line 113 of file osapi-sockets.h.

The documentation for this struct was generated from the following file:

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

## 11.327 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.*

### 11.327.1 Detailed Description

Storage buffer for generic network address.

This is a union type that helps to ensure a minimum alignment value for the data storage, such that it can be cast to the system-specific type without increasing alignment requirements.

Definition at line 96 of file osapi-sockets.h.

### 11.327.2 Field Documentation

#### 11.327.2.1 AlignPtr `void* OS_SockAddrData_t::AlignPtr`

Ensures pointer alignment.

Definition at line 100 of file osapi-sockets.h.

#### 11.327.2.2 AlignU32 `uint32 OS_SockAddrData_t::AlignU32`

Ensures uint32 alignment.

Definition at line 99 of file osapi-sockets.h.

#### 11.327.2.3 Buffer `uint8 OS_SockAddrData_t::Buffer[OS SOCKADDR_MAX_LEN]`

Ensures length of at least OS SOCKADDR\_MAX\_LEN.

Definition at line 98 of file osapi-sockets.h.

The documentation for this union was generated from the following file:

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

## 11.328 OS\_socket\_optval Union Reference

Socket option value.

```
#include <osapi-sockets.h>
```

## Data Fields

- `int32 IntVal`

### 11.328.1 Detailed Description

Socket option value.

This is used with `OS_SocketGetOption()` and `OS_SocketSetOption()` to store the option value that is get or set, respectively. Currently only integers values are relevant but defining as a union will allow other types to be transparently added in the future if needed.

Definition at line 150 of file osapi-sockets.h.

### 11.328.2 Field Documentation

#### 11.328.2.1 `IntVal int32 OS_socket_optval::IntVal`

Definition at line 152 of file osapi-sockets.h.

The documentation for this union was generated from the following file:

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

## 11.329 OS\_socket\_prop\_t Struct Reference

Encapsulates socket properties.

```
#include <osapi-sockets.h>
```

### Data Fields

- `char name [OS_MAX_API_NAME]`  
*Name of the socket.*
- `osal_id_t creator`  
*OSAL TaskID which opened the socket.*

#### 11.329.1 Detailed Description

Encapsulates socket properties.

This is for consistency with other OSAL resource types. Currently no extra properties are exposed here but this could change in a future revision of OSAL as needed.

Definition at line 123 of file osapi-sockets.h.

### 11.329.2 Field Documentation

#### 11.329.2.1 `creator osal_id_t OS_socket_prop_t::creator`

OSAL TaskID which opened the socket.

Definition at line 126 of file osapi-sockets.h.

#### 11.329.2.2 `name char OS_socket_prop_t::name [OS_MAX_API_NAME]`

Name of the socket.

Definition at line 125 of file osapi-sockets.h.

The documentation for this struct was generated from the following file:

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

## 11.330 OS\_static\_symbol\_record\_t Struct Reference

Associates a single symbol name with a memory address.

```
#include <osapi-module.h>
```

### Data Fields

- `const char * Name`
- `void(* Address )(void)`
- `const char * Module`

### 11.330.1 Detailed Description

Associates a single symbol name with a memory address.

If the OS\_STATIC\_SYMBOL\_TABLE feature is enabled, then an array of these structures should be provided by the application. When the application needs to find a symbol address, the static table will be checked in addition to (or instead of) the OS/library-provided lookup function.

This static symbol allows systems that do not implement dynamic module loading to maintain the same semantics as dynamically loaded modules.

Definition at line 113 of file osapi-module.h.

### 11.330.2 Field Documentation

#### 11.330.2.1 Address void(\* OS\_static\_symbol\_record\_t::Address) (void)

Definition at line 116 of file osapi-module.h.

#### 11.330.2.2 Module const char\* OS\_static\_symbol\_record\_t::Module

Definition at line 117 of file osapi-module.h.

#### 11.330.2.3 Name const char\* OS\_static\_symbol\_record\_t::Name

Definition at line 115 of file osapi-module.h.

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-module.h](#)

## 11.331 OS\_statvfs\_t Struct Reference

```
#include <osapi-filesystem.h>
```

### Data Fields

- `size_t block_size`
- `osal_blockcount_t total_blocks`
- `osal_blockcount_t blocks_free`

### 11.331.1 Detailed Description

Definition at line 49 of file osapi-filesystem.h.

### 11.331.2 Field Documentation

#### 11.331.2.1 block\_size size\_t OS\_statvfs\_t::block\_size

Block size of underlying FS

Definition at line 51 of file osapi-filesystem.h.

#### 11.331.2.2 blocks\_free osal\_blockcount\_t OS\_statvfs\_t::blocks\_free

Available blocks in underlying FS

Definition at line 53 of file osapi-filesystem.h.

**11.331.2.3 total\_blocks** `osal_blockcount_t OS_statvfs_t::total_blocks`

Total blocks in underlying FS

Definition at line 52 of file osapi-fs.h.

The documentation for this struct was generated from the following file:

- `osal/src/os/inc/osapi-fs.h`

## 11.332 OS\_task\_prop\_t Struct Reference

OSAL task properties.

```
#include <osapi-task.h>
```

### Data Fields

- `char name [OS_MAX_API_NAME]`
- `osal_id_t creator`
- `size_t stack_size`
- `osal_priority_t priority`

### 11.332.1 Detailed Description

OSAL task properties.

Definition at line 57 of file osapi-task.h.

### 11.332.2 Field Documentation

#### 11.332.2.1 creator

`osal_id_t OS_task_prop_t::creator`

Definition at line 60 of file osapi-task.h.

#### 11.332.2.2 name

`char OS_task_prop_t::name[OS_MAX_API_NAME]`

Definition at line 59 of file osapi-task.h.

#### 11.332.2.3 priority

`osal_priority_t OS_task_prop_t::priority`

Definition at line 62 of file osapi-task.h.

#### 11.332.2.4 stack\_size

`size_t OS_task_prop_t::stack_size`

Definition at line 61 of file osapi-task.h.

The documentation for this struct was generated from the following file:

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

## 11.333 OS\_time\_t Struct Reference

OSAL time interval structure.

```
#include <osapi-clock.h>
```

### Data Fields

- `int64 ticks`

### 11.333.1 Detailed Description

OSAL time interval structure.

This is used to represent a basic time interval.

When used with OS\_GetLocalTime/OS\_SetLocalTime, this represents the interval from the OS's epoch point, typically 01 Jan 1970 00:00:00 UTC on systems that have a persistent real time clock (RTC), or the system boot time if there is no RTC available.

Applications should not directly access fields within this structure, as the definition may change in future versions of OSAL. Instead, applications should use the accessor/conversion methods defined below.

Definition at line 45 of file osapi-clock.h.

### 11.333.2 Field Documentation

#### 11.333.2.1 ticks int64 OS\_time\_t::ticks

Ticks elapsed since reference point

Definition at line 47 of file osapi-clock.h.

Referenced by OS\_TimeAdd(), OS\_TimeAssembleFromMicroseconds(), OS\_TimeAssembleFromMilliseconds(), OS\_TimeAssembleFromNanoseconds(), OS\_TimeAssembleFromSubseconds(), OS\_TimeEqual(), OS\_TimeGetFractionalPart(), OS\_TimeGetSign(), OS\_TimeGetTotalMicroseconds(), OS\_TimeGetTotalMilliseconds(), OS\_TimeGetTotalNanoseconds(), OS\_TimeGetTotalSeconds(), and OS\_TimeSubtract().

The documentation for this struct was generated from the following file:

- [osal/src/os/inc/osapi-clock.h](#)

## 11.334 OS\_timebase\_prop\_t Struct Reference

Time base properties.

```
#include <osapi-timebase.h>
```

### Data Fields

- char [name \[OS\\_MAX\\_API\\_NAME\]](#)
- [osal\\_id\\_t creator](#)
- uint32 [nominal\\_interval\\_time](#)
- uint32 [freerun\\_time](#)
- uint32 [accuracy](#)

### 11.334.1 Detailed Description

Time base properties.

Definition at line 37 of file osapi-timebase.h.

### 11.334.2 Field Documentation

#### 11.334.2.1 accuracy uint32 OS\_timebase\_prop\_t::accuracy

Definition at line 43 of file osapi-timebase.h.

#### 11.334.2.2 creator osal\_id\_t OS\_timebase\_prop\_t::creator

Definition at line 40 of file osapi-timebase.h.

**11.334.2.3 freerun\_time** `uint32 OS_timebase_prop_t::freerun_time`

Definition at line 42 of file osapi-timebase.h.

**11.334.2.4 name** `char OS_timebase_prop_t::name[OS_MAX_API_NAME]`

Definition at line 39 of file osapi-timebase.h.

**11.334.2.5 nominal\_interval\_time** `uint32 OS_timebase_prop_t::nominal_interval_time`

Definition at line 41 of file osapi-timebase.h.

The documentation for this struct was generated from the following file:

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

## 11.335 OS\_timer\_prop\_t Struct Reference

Timer properties.

```
#include <osapi-timer.h>
```

### Data Fields

- `char name [OS_MAX_API_NAME]`
- `osal_id_t creator`
- `uint32 start_time`
- `uint32 interval_time`
- `uint32 accuracy`

### 11.335.1 Detailed Description

Timer properties.

Definition at line 37 of file osapi-timer.h.

### 11.335.2 Field Documentation

**11.335.2.1 accuracy** `uint32 OS_timer_prop_t::accuracy`

Definition at line 43 of file osapi-timer.h.

**11.335.2.2 creator** `osal_id_t OS_timer_prop_t::creator`

Definition at line 40 of file osapi-timer.h.

**11.335.2.3 interval\_time** `uint32 OS_timer_prop_t::interval_time`

Definition at line 42 of file osapi-timer.h.

**11.335.2.4 name** `char OS_timer_prop_t::name[OS_MAX_API_NAME]`

Definition at line 39 of file osapi-timer.h.

**11.335.2.5 start\_time uint32 OS\_timer\_prop\_t::start\_time**

Definition at line 41 of file osapi-timer.h.

The documentation for this struct was generated from the following file:

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

## 12 File Documentation

### 12.1 apps/cf/config/default\_cf\_extern\_typedefs.h File Reference

```
#include "cf_mission_cfg.h"
```

#### Data Structures

- struct **CF\_TxnFilenames**  
*Cache of source and destination filename.*

#### TypeDefs

- typedef struct **CF\_TxnFilenames CF\_TxnFilenames\_t**  
*Cache of source and destination filename.*
- typedef uint32 **CF\_EntityId\_t**  
*Entity id size.*
- typedef uint32 **CF\_TransactionSeq\_t**  
*transaction sequence number size*

#### Enumerations

- enum **CF\_CFDP\_Class\_t**{ **CF\_CFDP\_CLASS\_1** = 0 , **CF\_CFDP\_CLASS\_2** = 1 }  
*Values for CFDP file transfer class.*
- enum **CF\_QueueIdx\_t**{  
  **CF\_QueueIdx\_PEND** = 0 , **CF\_QueueIdx\_TX** = 1 , **CF\_QueueIdx\_RX** = 2 , **CF\_QueueIdx\_HIST** = 3 ,  
  **CF\_QueueIdx\_HIST\_FREE** = 4 , **CF\_QueueIdx\_FREE** = 5 , **CF\_QueueIdx\_NUM** = 6 }  
*CF queue identifiers.*

#### 12.1.1 Detailed Description

Declarations and prototypes for cf\_extern\_typedefs module

#### 12.1.2 Typedef Documentation

##### 12.1.2.1 **CF\_EntityId\_t** `typedef uint32 CF_EntityId_t`

Entity id size.

###### Description:

The maximum size of the entity id as expected for all CFDP packets. CF supports the spec's variable size of EID, where the actual size is selected at runtime, and therefore the size in CFDP PDUs may be smaller than the size specified here. This type only establishes the maximum size (and therefore maximum value) that an EID may be.

**Note**

This type is used in several CF commands, and so changing the size of this type will affect the following structs: CF\_ConfigTable\_t, configuration table - will change size of file CF\_ConfigPacket\_t, set config params command CF\_TxFileCmd\_t, transmit file command CF\_PlaybackDirCmd\_t, equivalent to above CF\_Transaction\_Payload\_t, any command that selects a transaction based on EID

**Limits**

Must be one of uint8, uint16, uint32, uint64.

Definition at line 94 of file default\_cf\_extern\_typedefs.h.

**12.1.2.2 CF\_TransactionSeq\_t** `typedef uint32 CF_TransactionSeq_t`  
transaction sequence number size**Description:**

The max size of the transaction sequence number as expected for all CFDP packets. CF supports the spec's variable size of TSN, where the actual size is selected at runtime, and therefore the size in CFDP PDUs may be smaller than the size specified here. This type only establishes the maximum size (and therefore maximum value) that a TSN may be.

**Note**

This type is used in several CF commands, and so changing the size of this type will affect the following structure: CF\_Transaction\_Payload\_t, any command that selects a transaction based on TSN

**Limits**

Must be one of uint8, uint16, uint32, uint64.

Definition at line 113 of file default\_cf\_extern\_typedefs.h.

**12.1.2.3 CF\_TxnFilenames\_t** `typedef struct CF_TxnFilenames CF_TxnFilenames_t`  
Cache of source and destination filename.

This pairs a source and destination file name together to be retained for future reference in the transaction/history

**12.1.3 Enumeration Type Documentation****12.1.3.1 CF\_CFDP\_Class\_t** `enum CF_CFDP_Class_t`  
Values for CFDP file transfer class.

The CFDP specification prescribes two classes/modes of file transfer protocol operation - unacknowledged/simple or acknowledged/reliable.

Defined per section 7.1 of CCSDS 727.0-B-5

**Enumerator**

<code>CF_CFDP_CLASS_1</code>	CFDP class 1 - Unreliable transfer.
<code>CF_CFDP_CLASS_2</code>	CFDP class 2 - Reliable transfer.

Definition at line 41 of file default\_cf\_extern\_typedefs.h.

### 12.1.3.2 CF\_QueueIdx\_t enum CF\_QueueIdx\_t

CF queue identifiers.

Enumerator

CF_QueueIdx_PEND	tx transactions that have not started
CF_QueueIdx_TX	tx transactions in progress
CF_QueueIdx_RX	rx transactions in progress
CF_QueueIdx_HIST	transaction history (completed)
CF_QueueIdx_HIST_FREE	unused transaction history structs
CF_QueueIdx_FREE	unused transaction structs
CF_QueueIdx_NUM	

Definition at line 50 of file default\_cf\_extern\_typedefs.h.

## 12.2 apps/cf/config/default\_cf\_fcncode\_values.h File Reference

### Macros

- #define CF\_CCVAL(x) CF\_FunctionCode\_##x

### Enumerations

- enum CF\_FunctionCode\_ {
 CF\_FunctionCode\_NOOP = 0 , CF\_FunctionCode\_RESET\_COUNTERS = 1 , CF\_FunctionCode\_TX\_FILE = 2 ,
 CF\_FunctionCode\_PLAYBACK\_DIR = 3 ,
 CF\_FunctionCode\_FREEZE = 4 , CF\_FunctionCode\_THAW = 5 , CF\_FunctionCode\_SUSPEND = 6 ,
 CF\_FunctionCode\_RESUME = 7 ,
 CF\_FunctionCode\_CANCEL = 8 , CF\_FunctionCode\_ABANDON = 9 , CF\_FunctionCode\_SET\_PARAM = 10 ,
 CF\_FunctionCode\_GET\_PARAM = 11 ,
 CF\_FunctionCode\_WRITE\_QUEUE = 15 , CF\_FunctionCode\_ENABLE\_DEQUEUE = 16 , CF\_FunctionCode\_DISABLE\_DEQUEUE =
 = 17 , CF\_FunctionCode\_ENABLE\_DIR\_POLLING = 18 ,
 CF\_FunctionCode\_DISABLE\_DIR\_POLLING = 19 , CF\_FunctionCode\_PURGE\_QUEUE = 21 , CF\_FunctionCode\_ENABLE\_ENG =
 = 22 , CF\_FunctionCode\_DISABLE\_ENGINE = 23 }

### 12.2.1 Detailed Description

Specification for the CFS CFDP (CF) command function codes

#### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

### 12.2.2 Macro Definition Documentation

#### 12.2.2.1 CF\_CCVAL #define CF\_CCVAL( x ) CF\_FunctionCode\_##x

Definition at line 36 of file default\_cf\_fcncode\_values.h.

### 12.2.3 Enumeration Type Documentation

#### 12.2.3.1 CF\_FunctionCode\_ `enum CF_FunctionCode_`

Enumerator

CF_FunctionCode_NOOP	
CF_FunctionCode_RESET_COUNTERS	
CF_FunctionCode_TX_FILE	
CF_FunctionCode_PLAYBACK_DIR	
CF_FunctionCode_FREEZE	
CF_FunctionCode_THAW	
CF_FunctionCode_SUSPEND	
CF_FunctionCode_RESUME	
CF_FunctionCode_CANCEL	
CF_FunctionCode_ABANDON	
CF_FunctionCode_SET_PARAM	
CF_FunctionCode_GET_PARAM	
CF_FunctionCode_WRITE_QUEUE	
CF_FunctionCode_ENABLE_DEQUEUE	
CF_FunctionCode_DISABLE_DEQUEUE	
CF_FunctionCode_ENABLE_DIR_POLLING	
CF_FunctionCode_DISABLE_DIR_POLLING	
CF_FunctionCode_PURGE_QUEUE	
CF_FunctionCode_ENABLE_ENGINE	
CF_FunctionCode_DISABLE_ENGINE	

Definition at line 38 of file default\_cf\_fnccode\_values.h.

### 12.3 apps/cf/config/default\_cf\_interface\_cfg\_values.h File Reference

#### Macros

- `#define CF_INTERFACE_CFGVAL(x) DEFAULT_CF_##x`

#### 12.3.1 Detailed Description

CFS CFDP (CF) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

#### 12.3.2 Macro Definition Documentation

```
12.3.2.1 CF_INTERFACE_CFGVAL #define CF_INTERFACE_CFGVAL( x ) DEFAULT_CF_##x
```

Definition at line 38 of file default\_cf\_interface\_cfg\_values.h.

## 12.4 apps/cf/config/default\_cf\_internal\_cfg\_values.h File Reference

### Macros

- `#define CF_INTERNAL_CFGVAL(x) DEFAULT_CF_##x`

#### 12.4.1 Detailed Description

CFS CFDP (CF) Application Private Config Definitions

This provides default values for configurable items that are internal to this module and do NOT affect the interface(s) of this module. Changes to items in this file only affect the local module and will be transparent to external entities that are using the public interface(s).

### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

#### 12.4.2 Macro Definition Documentation

```
12.4.2.1 CF_INTERNAL_CFGVAL #define CF_INTERNAL_CFGVAL( x ) DEFAULT_CF_##x
```

Definition at line 37 of file default\_cf\_internal\_cfg\_values.h.

## 12.5 apps/cf/config/default\_cf\_mission\_cfg.h File Reference

```
#include "cf_interface_cfg.h"
```

### Macros

- `#define CF_FILENAME_MAX_PATH (CFE_MISSION_MAX_PATH_LEN - CFE_MISSION_MAX_FILE_LEN)`  
*Maximum file path (not including file name)*

#### 12.5.1 Detailed Description

CFS CFDP (CF) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

#### 12.5.2 Macro Definition Documentation

**12.5.2.1 CF\_FILENAME\_MAX\_PATH** #define CF\_FILENAME\_MAX\_PATH (CFE\_MISSION\_MAX\_PATH\_LEN - CFE\_MISSION\_MAX\_FILE\_I  
Maximum file path (not including file name)

Limits:

Definition at line 43 of file default\_cf\_mission\_cfg.h.

## 12.6 apps/cf/config/default\_cf\_msg.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_fcncodes.h"
#include "cf_msgdefs.h"
#include "cf_msgstruct.h"
```

### Macros

- #define CF\_COMPOUND\_KEY (254)
- #define CF\_ALL\_CHANNELS (255)
- #define CF\_ALL\_POLLDIRS (CF\_ALL\_CHANNELS)

### 12.6.1 Detailed Description

Specification for the CFS CFDP (CF) command and telemetry message data types.

This is a compatibility header for the "cf\_msg.h" file that has traditionally provided the message definitions for cFS apps.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.6.2 Macro Definition Documentation

**12.6.2.1 CF\_ALL\_CHANNELS** #define CF\_ALL\_CHANNELS (255)

Definition at line 41 of file default\_cf\_msg.h.

**12.6.2.2 CF\_ALL\_POLLDIRS** #define CF\_ALL\_POLLDIRS (CF\_ALL\_CHANNELS)

Definition at line 42 of file default\_cf\_msg.h.

**12.6.2.3 CF\_COMPOUND\_KEY** #define CF\_COMPOUND\_KEY (254)

Definition at line 40 of file default\_cf\_msg.h.

## 12.7 apps/cf/config/default\_cf\_msgdefs.h File Reference

```
#include "common_types.h"
#include "cf_extern_typedefs.h"
```

## Data Structures

- struct [CF\\_HkCmdCounters](#)  
*Housekeeping command counters.*
- struct [CF\\_HkSent](#)  
*Housekeeping sent counters.*
- struct [CF\\_HkRecv](#)  
*Housekeeping received counters.*
- struct [CF\\_HkFault](#)  
*Housekeeping fault counters.*
- struct [CF\\_HkCounters](#)  
*Housekeeping counters.*
- struct [CF\\_HkChannel\\_Data](#)  
*Housekeeping channel data.*
- struct [CF\\_HkPacket\\_Payload](#)  
*Housekeeping packet.*
- struct [CF\\_EotPacket\\_Payload](#)  
*End of transaction packet.*
- union [CF\\_UnionArgs\\_Payload](#)  
*Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.*
- struct [CF\\_GetParam\\_Payload](#)  
*Get parameter command structure.*
- struct [CF\\_SetParam\\_Payload](#)  
*Set parameter command structure.*
- struct [CF\\_TxFile\\_Payload](#)  
*Transmit file command structure.*
- struct [CF\\_WriteQueue\\_Payload](#)  
*Write Queue command structure.*
- struct [CF\\_Transaction\\_Payload](#)  
*Transaction command structure.*

## Typedefs

- typedef struct [CF\\_HkCmdCounters](#) [CF\\_HkCmdCounters\\_t](#)  
*Housekeeping command counters.*
- typedef struct [CF\\_HkSent](#) [CF\\_HkSent\\_t](#)  
*Housekeeping sent counters.*
- typedef struct [CF\\_HkRecv](#) [CF\\_HkRecv\\_t](#)  
*Housekeeping received counters.*
- typedef struct [CF\\_HkFault](#) [CF\\_HkFault\\_t](#)  
*Housekeeping fault counters.*
- typedef struct [CF\\_HkCounters](#) [CF\\_HkCounters\\_t](#)  
*Housekeeping counters.*
- typedef struct [CF\\_HkChannel\\_Data](#) [CF\\_HkChannel\\_Data\\_t](#)  
*Housekeeping channel data.*
- typedef struct [CF\\_HkPacket\\_Payload](#) [CF\\_HkPacket\\_Payload\\_t](#)  
*Housekeeping packet.*
- typedef struct [CF\\_EotPacket\\_Payload](#) [CF\\_EotPacket\\_Payload\\_t](#)

*End of transaction packet.*

- **typedef union CF\_UnionArgs\_Payload CF\_UnionArgs\_Payload\_t**  
*Command payload argument union to support 4 uint8's, 2 uint16's or 1 uint32.*
- **typedef struct CF\_GetParam\_Payload CF\_GetParam\_Payload\_t**  
*Get parameter command structure.*
- **typedef struct CF\_SetParam\_Payload CF\_SetParam\_Payload\_t**  
*Set parameter command structure.*
- **typedef struct CF\_TxFile\_Payload CF\_TxFile\_Payload\_t**  
*Transmit file command structure.*
- **typedef struct CF\_WriteQueue\_Payload CF\_WriteQueue\_Payload\_t**  
*Write Queue command structure.*
- **typedef struct CF\_Transaction\_Payload CF\_Transaction\_Payload\_t**  
*Transaction command structure.*

## Enumerations

- **enum CF\_Reset\_t {**  
CF\_Reset\_all = 0 , CF\_Reset\_command = 1 , CF\_Reset\_fault = 2 , CF\_Reset\_up = 3 ,  
CF\_Reset\_down = 4 **}**  
*IDs for use for Reset cmd.*
- **enum CF\_Type\_t { CF\_Type\_all = 0 , CF\_Type\_up = 1 , CF\_Type\_down = 2 }**  
*Type IDs for use for Write Queue cmd.*
- **enum CF\_Queue\_t { CF\_Queue\_pend = 0 , CF\_Queue\_active = 1 , CF\_Queue\_history = 2 , CF\_Queue\_all = 3 }**  
*Queue IDs for use for Write Queue cmd.*
- **enum CF\_GetSet\_ValueID\_t {**  
CF\_GetSet\_ValueID\_ticks\_per\_second , CF\_GetSet\_ValueID\_rx\_crc\_calc\_bytes\_per\_wakeup , CF\_GetSet\_ValueID\_ack\_timer\_s ,  
CF\_GetSet\_ValueID\_nak\_timer\_s ,  
CF\_GetSet\_ValueID\_inactivity\_timer\_s , CF\_GetSet\_ValueID\_outgoing\_file\_chunk\_size , CF\_GetSet\_ValueID\_ack\_limit ,  
CF\_GetSet\_ValueID\_nak\_limit ,  
CF\_GetSet\_ValueID\_local\_eid , CF\_GetSet\_ValueID\_chan\_max\_outgoing\_messages\_per\_wakeup , CF\_GetSet\_ValueID\_MAX  
**}**  
*Parameter IDs for use with Get/Set parameter messages.*

### 12.7.1 Detailed Description

Specification for the CFS CFDP (CF) command and telemetry message payload and constant definitions.

## 12.8 apps/cf/config/default\_cf\_msgid\_values.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cf_topicids.h"
```

### Macros

- #define CFE\_PLATFORM\_CF\_CMD\_MIDVAL(x) CFE\_PLATFORM\_CMD\_TOPICID\_TO\_MIDV(CFE\_PLATFORM\_CF\_CMD\_MIDVAL(x))
- #define CFE\_PLATFORM\_CF\_TLM\_MIDVAL(x) CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV(CFE\_PLATFORM\_CF\_TLM\_TOPICID\_TO\_MIDV(CFE\_PLATFORM\_CF\_TLM\_MIDVAL(x)))

### 12.8.1 Detailed Description

CFS CFDP (CF) Application Message IDs

### 12.8.2 Macro Definition Documentation

**12.8.2.1 CFE\_PLATFORM\_CF\_CMD\_MIDVAL** #define CFE\_PLATFORM\_CF\_CMD\_MIDVAL(  
  x ) CFE\_PLATFORM\_CMD\_TOPICID\_TO\_MIDV(CFE\_MISSION\_CF\_##x##\_TOPICID)  
Definition at line 30 of file default\_cf\_msgid\_values.h.

**12.8.2.2 CFE\_PLATFORM\_CF\_TLM\_MIDVAL** #define CFE\_PLATFORM\_CF\_TLM\_MIDVAL(  
  x ) CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV(CFE\_MISSION\_CF\_##x##\_TOPICID)  
Definition at line 31 of file default\_cf\_msgid\_values.h.

## 12.9 apps/cf/config/default\_cf\_msgids.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cf_msgid_values.h"
```

### Macros

- #define CF\_CMD\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(CMD)  
*Message ID for commands.*
- #define CF\_SEND\_HK\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(SEND\_HK)  
*Message ID to request housekeeping telemetry.*
- #define CF\_WAKE\_UP\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(WAKE\_UP)  
*Message ID for waking up the processing cycle.*
- #define CF\_HK\_TLM\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(HK\_TLM)  
*Message ID for housekeeping telemetry.*
- #define CF\_EOT\_TLM\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(EOT\_TLM)  
*Message ID for end of transaction telemetry.*
- #define CF\_CH0\_TX\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(CH0\_TX)
- #define CF\_CH1\_TX\_MID CFE\_PLATFORM\_CF\_CMD\_MIDVAL(CH1\_TX)
- #define CF\_CH0\_RX\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(CH0\_RX)
- #define CF\_CH1\_RX\_MID CFE\_PLATFORM\_CF\_TLM\_MIDVAL(CH1\_RX)

### 12.9.1 Detailed Description

CFS CFDP (CF) Application Message IDs

## 12.10 apps/cf/config/default\_cf\_msgstruct.h File Reference

```
#include "cf_msgdefs.h"
#include "cf_mission_cfg.h"
#include "cfe_msg_hdr.h"
```

## Data Structures

- struct **CF\_HkPacket**  
*Housekeeping packet.*
- struct **CF\_EotPacket**  
*End of transaction packet.*
- struct **CF\_NoopCmd**  
*Noop command structure.*
- struct **CF\_EnableEngineCmd**  
*EnableEngine command structure.*
- struct **CF\_DisableEngineCmd**  
*DisableEngine command structure.*
- struct **CF\_ResetCountersCmd**  
*Reset command structure.*
- struct **CF\_FreezeCmd**  
*Freeze command structure.*
- struct **CF\_ThawCmd**  
*Thaw command structure.*
- struct **CF\_EnableDequeueCmd**  
*EnableDequeue command structure.*
- struct **CF\_DisableDequeueCmd**  
*DisableDequeue command structure.*
- struct **CF\_EnableDirPollingCmd**  
*EnableDirPolling command structure.*
- struct **CF\_DisableDirPollingCmd**  
*DisableDirPolling command structure.*
- struct **CF\_PurgeQueueCmd**  
*PurgeQueue command structure.*
- struct **CF\_GetParamCmd**  
*Get parameter command structure.*
- struct **CF\_SetParamCmd**  
*Set parameter command structure.*
- struct **CF\_TxFileCmd**  
*Transmit file command structure.*
- struct **CF\_WriteQueueCmd**  
*Write Queue command structure.*
- struct **CF\_PlaybackDirCmd**  
*Playback directory command structure.*
- struct **CF\_SuspendCmd**  
*Suspend command structure.*
- struct **CF\_ResumeCmd**  
*Resume command structure.*
- struct **CF\_CancelCmd**  
*Cancel command structure.*
- struct **CF\_AbandonCmd**  
*Abandon command structure.*
- struct **CF\_SendHkCmd**  
*Send Housekeeping Command.*
- struct **CF\_WakeupCmd**  
*Wake Up Command.*

## Typedefs

- `typedef struct CF_HkPacket CF_HkPacket_t`  
*Housekeeping packet.*
- `typedef struct CF_EotPacket CF_EotPacket_t`  
*End of transaction packet.*
- `typedef struct CF_NoopCmd CF_NoopCmd_t`  
*Noop command structure.*
- `typedef struct CF_EnableEngineCmd CF_EnableEngineCmd_t`  
*EnableEngine command structure.*
- `typedef struct CF_DisableEngineCmd CF_DisableEngineCmd_t`  
*DisableEngine command structure.*
- `typedef struct CF_ResetCountersCmd CF_ResetCountersCmd_t`  
*Reset command structure.*
- `typedef struct CF_FreezeCmd CF_FreezeCmd_t`  
*Freeze command structure.*
- `typedef struct CF_ThawCmd CF_ThawCmd_t`  
*Thaw command structure.*
- `typedef struct CF_EnableDequeueCmd CF_EnableDequeueCmd_t`  
*EnableDequeue command structure.*
- `typedef struct CF_DisableDequeueCmd CF_DisableDequeueCmd_t`  
*DisableDequeue command structure.*
- `typedef struct CF_EnableDirPollingCmd CF_EnableDirPollingCmd_t`  
*EnableDirPolling command structure.*
- `typedef struct CF_DisableDirPollingCmd CF_DisableDirPollingCmd_t`  
*DisableDirPolling command structure.*
- `typedef struct CF_PurgeQueueCmd CF_PurgeQueueCmd_t`  
*PurgeQueue command structure.*
- `typedef struct CF_GetParamCmd CF_GetParamCmd_t`  
*Get parameter command structure.*
- `typedef struct CF_SetParamCmd CF_SetParamCmd_t`  
*Set parameter command structure.*
- `typedef struct CF_TxFileCmd CF_TxFileCmd_t`  
*Transmit file command structure.*
- `typedef struct CF_WriteQueueCmd CF_WriteQueueCmd_t`  
*Write Queue command structure.*
- `typedef struct CF_PlaybackDirCmd CF_PlaybackDirCmd_t`  
*Playback directory command structure.*
- `typedef struct CF_SuspendCmd CF_SuspendCmd_t`  
*Suspend command structure.*
- `typedef struct CF_ResumeCmd CF_ResumeCmd_t`  
*Resume command structure.*
- `typedef struct CF_CancelCmd CF_CancelCmd_t`  
*Cancel command structure.*
- `typedef struct CF_AbandonCmd CF_AbandonCmd_t`  
*Abandon command structure.*
- `typedef struct CF_SendHkCmd CF_SendHkCmd_t`  
*Send Housekeeping Command.*
- `typedef struct CF_WakeupCmd CF_WakeupCmd_t`  
*Wake Up Command.*

### 12.10.1 Detailed Description

Specification for the CFS CFDP (CF) command and telemetry message data types.

#### Note

Constants and enumerated types related to these message structures are defined in cf\_msgdefs.h.

## 12.11 apps/cf/config/default\_cf\_platform\_cfg.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_internal_cfg.h"
```

### Macros

- `#define CF_CHANNEL_NUM_RX_CHUNKS_PER_TRANSACTION`  
*RX chunks per transaction (per channel)*
- `#define CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION`  
*TX chunks per transaction (per channel)*
- `#define CF_TOTAL_CHUNKS (CF_NAK_MAX_SEGMENTS * 4)`  
*Total number of chunks (tx, rx, all channels)*
- `#define CF_MISSION_REV 0`  
*Mission specific version number.*

### 12.11.1 Detailed Description

CFS CFDP (CF) Application Platform Configuration Header File

This is a compatibility header for the "platform\_cfg.h" file that has traditionally provided both public and private config definitions for each CFS app.

These definitions are now provided in two separate files, one for the public/mission scope and one for internal scope.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.11.2 Macro Definition Documentation

#### 12.11.2.1 CF\_CHANNEL\_NUM\_RX\_CHUNKS\_PER\_TRANSACTION `#define CF_CHANNEL_NUM_RX_CHUNKS_← PER_TRANSACTION`

##### Value:

```
{ CF_NAK_MAX_SEGMENTS, CF_NAK_MAX_SEGMENTS \ }
```

RX chunks per transaction (per channel)

##### Description:

Number of chunks per transaction per channel (RX).

CHUNKS - A chunk is a representation of a range (offset, size) of data received by a receiver.

Class 2 CFDP deals with NAK, so received data must be tracked for receivers in order to generate the NAK. The sender must also keep track of NAK requests and send new file data PDUs as a result. (array size must be CF\_NUM\_CHANNELS) CF\_CHANNEL\_NUM\_RX\_CHUNKS\_PER\_TRANSACTION is an array for each channel indicating the number of chunks per transaction CF\_CHANNEL\_NUM\_TX\_CHUNKS\_PER\_TRANSACTION is an array for each channel indicating the number of chunks to keep track of NAK requests from the receiver per transaction

Limits:

Definition at line 61 of file default\_cf\_platform\_cfg.h.

**12.11.2.2 CF\_CHANNEL\_NUM\_TX\_CHUNKS\_PER\_TRANSACTION** `#define CF_CHANNEL_NUM_TX_CHUNKS_PER_TRANSACTION`

**Value:**

```
{ CF_NAK_MAX_SEGMENTS, CF_NAK_MAX_SEGMENTS \ }
```

TX chunks per transaction (per channel)

**Description:**

Number of chunks per transaction per channel (TX).

Limits:

Definition at line 75 of file default\_cf\_platform\_cfg.h.

**12.11.2.3 CF\_MISSION\_REV** `#define CF_MISSION_REV 0`

Mission specific version number.

**Description:**

An application version number consists of four parts: major version number, minor version number, revision number and mission specific revision number. The mission specific revision number is defined here such that missions can manage as a configuration definition

Limits:

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

Definition at line 108 of file default\_cf\_platform\_cfg.h.

**12.11.2.4 CF\_TOTAL\_CHUNKS** `#define CF_TOTAL_CHUNKS (CF_NAK_MAX_SEGMENTS * 4)`

Total number of chunks (tx, rx, all channels)

**Description:**

Must be equal to the sum of all values input in CF\_CHANNEL\_NUM\_RX\_CHUNKS\_PER\_TRANSACTION and CF\_CHANNEL\_NUM\_TX\_CHUNKS\_PER\_TRANSACTION.

Limits:

Definition at line 92 of file default\_cf\_platform\_cfg.h.

## 12.12 apps/cf/config/default\_cf\_tbl.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_tbldefs.h"
#include "cf_tblstruct.h"
```

### 12.12.1 Detailed Description

Specification for the CFS CFDP (CF) table structures

#### Note

Constants and enumerated types related to these table structures are defined in cf\_tbldefs.h.

## 12.13 apps/cf/config/default\_cf\_tbldefs.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_extern_typedefs.h"
#include "cf_tblstruct.h"
```

### Data Structures

- struct **CF\_PollDir**  
*Configuration entry for directory polling.*
- struct **CF\_ChannelConfig**  
*Configuration entry for CFDP channel.*

### TypeDefs

- typedef struct **CF\_PollDir** **CF\_PollDir\_t**  
*Configuration entry for directory polling.*
- typedef struct **CF\_ChannelConfig** **CF\_ChannelConfig\_t**  
*Configuration entry for CFDP channel.*

### 12.13.1 Detailed Description

Specification for the CFS CFDP (CF) table related constant definitions.

#### Note

These Macro definitions have been put in this file (instead of cf\_tbl.h). DO NOT PUT ANY TYPEDEFS OR STRUCTURE DEFINITIONS IN THIS FILE! ADD THEM TO cf\_tbl.h IF NEEDED!

### 12.13.2 Typedef Documentation

#### 12.13.2.1 **CF\_ChannelConfig\_t** `typedef struct CF_ChannelConfig CF_ChannelConfig_t`

Configuration entry for CFDP channel.

#### 12.13.2.2 **CF\_PollDir\_t** `typedef struct CF_PollDir CF_PollDir_t`

Configuration entry for directory polling.

## 12.14 apps/cf/config/default\_cf\_tblstruct.h File Reference

```
#include "common_types.h"
#include "cf_mission_cfg.h"
#include "cf_tbldefs.h"
```

## Data Structures

- struct [CF\\_ConfigTable](#)  
*Top-level CFDP configuration structure.*

## Typedefs

- typedef struct [CF\\_ConfigTable](#) [CF\\_ConfigTable\\_t](#)  
*Top-level CFDP configuration structure.*

### 12.14.1 Detailed Description

Specification for the CFS CFDP (CF) table structures

Provides default definitions for HK table structures

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.14.2 Typedef Documentation

#### 12.14.2.1 [CF\\_ConfigTable\\_t](#) [typedef struct CF\\_ConfigTable CF\\_ConfigTable\\_t](#) Top-level CFDP configuration structure.

## 12.15 apps/cf/config/default\_cf\_topicid\_values.h File Reference

### Macros

- #define [CFE\\_MISSION\\_CF\\_TIDVAL\(x\)](#) DEFAULT\_CFE\_MISSION\_CF\_##x##\_TOPICID

### 12.15.1 Detailed Description

CFDP (CF) Application Topic IDs

### 12.15.2 Macro Definition Documentation

#### 12.15.2.1 [CFE\\_MISSION\\_CF\\_TIDVAL](#) [#define CFE\\_MISSION\\_CF\\_TIDVAL\(](#) [x \)](#) DEFAULT\_CFE\_MISSION\_CF\_##x##\_TOPICID

Definition at line 27 of file default\_cf\_topicid\_values.h.

## 12.16 apps/cf/config/eds\_cf\_extern\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_resourceid_typedef.h"
#include "cf_eds_typedefs.h"
```

### Macros

- #define [CF\\_QueueIdx\\_NUM](#) (1 + EdsDataType\_EdsEnum\_CF\_QueueIdx\_t\_MAX)
- #define [CF\\_GetSet\\_ValueID\\_MAX](#) (1 + EdsDataType\_EdsEnum\_CF\_GetSet\_ValueID\_t\_MAX)

**Typedefs**

- `typedef CF_QueueIdx_Enum_t CF_QueueIdx_t`
- `typedef CF_EntityId_Atom_t CF_EntityId_t`
- `typedef CF_TransactionSeq_Atom_t CF_TransactionSeq_t`
- `typedef CF_CFDP_Enum_t CF_CFDP_Class_t`
- `typedef CF_GetSet_ValueID_Enum_t CF_GetSet_ValueID_t`
- `typedef EdsDataType_BASE_TYPES_PathName_t CF_PathName_t`
- `typedef EdsDataType_BASE_TYPES_FileName_t CF_FileName_t`

**12.16.1 Detailed Description**

Declarations and prototypes for cf\_extern\_typedefs module

**12.16.2 Macro Definition Documentation**

**12.16.2.1 CF\_GetSet\_ValueID\_MAX** `#define CF_GetSet_ValueID_MAX (1 + EdsDataType_EdsEnum_CF_GetSet_ValueID_t_MAX)`

Definition at line 39 of file eds\_cf\_extern\_typedefs.h.

**12.16.2.2 CF\_QueueIdx\_NUM** `#define CF_QueueIdx_NUM (1 + EdsDataType_EdsEnum_CF_QueueIdx_t_MAX)`

Definition at line 38 of file eds\_cf\_extern\_typedefs.h.

**12.16.3 Typedef Documentation**

**12.16.3.1 CF\_CFDP\_Class\_t** `typedef CF_CFDP_Enum_t CF_CFDP_Class_t`

Definition at line 44 of file eds\_cf\_extern\_typedefs.h.

**12.16.3.2 CF\_EntityId\_t** `typedef CF_EntityId_Atom_t CF_EntityId_t`

Definition at line 41 of file eds\_cf\_extern\_typedefs.h.

**12.16.3.3 CF\_FileName\_t** `typedef EdsDataType_BASE_TYPES_FileName_t CF_FileName_t`

Definition at line 48 of file eds\_cf\_extern\_typedefs.h.

**12.16.3.4 CF\_GetSet\_ValueID\_t** `typedef CF_GetSet_ValueID_Enum_t CF_GetSet_ValueID_t`

Definition at line 45 of file eds\_cf\_extern\_typedefs.h.

**12.16.3.5 CF\_PathName\_t** `typedef EdsDataType_BASE_TYPES_PathName_t CF_PathName_t`

Definition at line 47 of file eds\_cf\_extern\_typedefs.h.

**12.16.3.6 CF\_QueueIdx\_t** `typedef CF_QueueIdx_Enum_t CF_QueueIdx_t`

Definition at line 36 of file eds\_cf\_extern\_typedefs.h.

**12.16.3.7 CF\_TransactionSeq\_t** `typedef CF_TransactionSeq_Atom_t CF_TransactionSeq_t`  
Definition at line 42 of file eds\_cf\_extern\_typedefs.h.

## 12.17 apps/cf/config/eds\_cf\_fcncode\_values.h File Reference

```
#include "cf_eds_cc.h"
```

### Macros

- `#define CF_CCVAL(x) EDS_CONTAINER_CF_##x##_CC`

#### 12.17.1 Detailed Description

Specification for the CFS CFDP (CF) command function codes

#### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

#### 12.17.2 Macro Definition Documentation

**12.17.2.1 CF\_CCVAL** `#define CF_CCVAL(`  
`x  ) EDS_CONTAINER_CF_##x##_CC`  
Definition at line 38 of file eds\_cf\_fcncode\_values.h.

## 12.18 apps/cf/config/eds\_cf\_interface\_cfg\_values.h File Reference

```
#include "cf_eds_designparameters.h"
```

### Macros

- `#define CF_INTERFACE_CFGVAL(x) EdsParam_CF_##x`

#### 12.18.1 Detailed Description

CFS CFDP (CF) Application Public Definitions

This provides default values for configurable items that affect the interface(s) of this module. This includes the CMD/TLM message interface, tables definitions, and any other data products that serve to exchange information with other entities.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

#### 12.18.2 Macro Definition Documentation

**12.18.2.1 CF\_INTERFACE\_CFGVAL** `#define CF_INTERFACE_CFGVAL(`  
`x  ) EdsParam_CF_##x`  
Definition at line 39 of file eds\_cf\_interface\_cfg\_values.h.

## 12.19 apps/cf/config/eds\_cf\_msgdefs.h File Reference

```
#include "cf_eds_typedefs.h"
#include "cf_fcnCodes.h"
```

### 12.19.1 Detailed Description

Specification for the CFS CFDP (CF) command and telemetry message constant definitions.

## 12.20 apps/cf/config/eds\_cf\_msgstruct.h File Reference

```
#include "cf_eds_typedefs.h"
```

### 12.20.1 Detailed Description

Specification for the CFS CFDP (CF) Command and Telemetry packet definition file.

#### Note

Constants and enumerated types related to these message structures are defined in cf\_msgdefs.h.

## 12.21 apps/cf/config/eds\_cf\_tbldefs.h File Reference

```
#include "cf_eds_typedefs.h"
```

### 12.21.1 Detailed Description

Specification for the CF table related constant definitions.

## 12.22 apps/cf/config/eds\_cf\_tblstruct.h File Reference

```
#include "cf_eds_typedefs.h"
```

### 12.22.1 Detailed Description

Specification for the CF table definitions.

## 12.23 apps/cf/config/eds\_cf\_topicid\_values.h File Reference

```
#include "cfe_mission_eds_designparameters.h"
```

### Macros

- #define CFE\_MISSION\_CF\_TIDVAL(x) EdsParam\_CFE\_MISSION\_CF\_##x##\_TOPICID

### 12.23.1 Detailed Description

CFDP (CF) Application Topic IDs

### 12.23.2 Macro Definition Documentation

**12.23.2.1 CFE\_MISSION\_CF\_TIDVAL** #define CFE\_MISSION\_CF\_TIDVAL ( x ) EdsParam\_CFE\_MISSION\_CF\_##x##\_TOPICID  
Definition at line 29 of file eds\_cf\_topicid\_values.h.

## 12.24 apps/cf/docs/dox\_src/cfs\_cf.dox File Reference

### 12.25 apps/cf/fsw/inc/cf\_eventids.h File Reference

#### Macros

- #define CF\_INIT\_INF\_EID 20  
*CF Initialization Event ID.*
- #define CF\_INIT\_TBL\_CHECK\_REL\_ERR\_EID 21  
*CF Check Table Release Address Failed Event ID.*
- #define CF\_INIT\_TBL\_CHECK\_MAN\_ERR\_EID 22  
*CF Check Table Manage Failed Event ID.*
- #define CF\_INIT\_TBL\_CHECK\_GA\_ERR\_EID 23  
*CF Check Table Get Address Failed Event ID.*
- #define CF\_INIT\_TBL\_REG\_ERR\_EID 24  
*CF Table Registration At Initialization Failed Event ID.*
- #define CF\_INIT\_TBL\_LOAD\_ERR\_EID 25  
*CF Table Load At Initialization Failed Event ID.*
- #define CF\_INIT\_TBL\_MANAGE\_ERR\_EID 26  
*CF Table Manage At Initialization Failed Event ID.*
- #define CF\_INIT\_TBL\_GETADDR\_ERR\_EID 27  
*CF Table Get Address At Initialization Failed Event ID.*
- #define CF\_MID\_ERR\_EID 28  
*CF Message ID Invalid Event ID.*
- #define CF\_INIT\_MSG\_RECV\_ERR\_EID 29  
*CF SB Receive Buffer Failed Event ID.*
- #define CF\_INIT\_SEM\_ERR\_EID 30  
*CF Channel Semaphore Initialization Failed Event ID.*
- #define CF\_CR\_CHANNEL\_PIPE\_ERR\_EID 31  
*CF Channel Create Pipe Failed Event ID.*
- #define CF\_INIT\_SUB\_ERR\_EID 32  
*CF Channel Message Subscription Failed Event ID.*
- #define CF\_INIT\_TPS\_ERR\_EID 33  
*CF Ticks Per Second Config Table Validation Failed Event ID.*
- #define CF\_INIT\_CRC\_ALIGN\_ERR\_EID 34  
*CF CRC Bytes Per Wakeup Config Table Validation Failed Event ID.*
- #define CF\_INIT\_OUTGOING\_SIZE\_ERR\_EID 35  
*CF Outgoing Chunk Size Config Table Validation Failed Event ID.*
- #define CF\_CR\_PIPE\_ERR\_EID 36  
*CF Create SB Command Pipe at Initialization Failed Event ID.*
- #define CF\_PDU\_MD\_RECVD\_INF\_EID 40  
*CF Metadata PDU Received Event ID.*
- #define CF\_PDU\_SHORT\_HEADER\_ERR\_EID 41  
*CF PDU Header Too Short Event ID.*
- #define CF\_PDU\_MD\_SHORT\_ERR\_EID 43

- #define CF\_PDU\_INVALID\_SRC\_LEN\_ERR\_EID 44  
*CF Metadata PDU Too Short Event ID.*
- #define CF\_PDU\_INVALID\_DST\_LEN\_ERR\_EID 45  
*CF Metadata PDU Source Filename Length Invalid Event ID.*
- #define CF\_PDU\_FD\_SHORT\_ERR\_EID 46  
*CF File Data PDU Too Short Event ID.*
- #define CF\_PDU\_EOF\_SHORT\_ERR\_EID 47  
*CF End-Of-File PDU Too Short Event ID.*
- #define CF\_PDU\_ACK\_SHORT\_ERR\_EID 48  
*CF Acknowledgment PDU Too Short Event ID.*
- #define CF\_PDU\_FIN\_SHORT\_ERR\_EID 49  
*CF Finished PDU Too Short Event ID.*
- #define CF\_PDU\_NAK\_SHORT\_ERR\_EID 50  
*CF Negative Acknowledgment PDU Too Short Event ID.*
- #define CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID 54  
*CF File Data PDU Unsupported Option Event ID.*
- #define CF\_PDU\_LARGE\_FILE\_ERR\_EID 55  
*CF PDU Header Large File Flag Set Event ID.*
- #define CF\_PDU\_TRUNCATION\_ERR\_EID 56  
*CF PDU Header Field Truncation.*
- #define CF\_RESET\_FREED\_XACT\_DBG\_EID 59  
*Attempt to reset a transaction that has already been freed.*
- #define CF\_CFDP\_RX\_DROPPED\_ERR\_EID 60  
*CF PDU Received Without Existing Transaction, Dropped Due To Max RX Reached Event ID.*
- #define CF\_CFDP\_INVALID\_DST\_ERR\_EID 61  
*CF PDU Received With Invalid Destination Entity ID Event ID.*
- #define CF\_CFDP\_IDLE\_MD\_ERR\_EID 62  
*CF Invalid Metadata PDU Received On Idle Transaction Event ID.*
- #define CF\_CFDP\_FD\_UNHANDLED\_ERR\_EID 63  
*CF Non-metadata File Directive PDU Received On Idle Transaction Event ID.*
- #define CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID 64  
*CF Transmission Request Rejected Due To Max Commanded TX Reached Event ID.*
- #define CF\_CFDP\_OPENDIR\_ERR\_EID 65  
*CF Playback/Polling Directory Open Failed Event ID.*
- #define CF\_CFDP\_DIR\_SLOT\_ERR\_EID 66  
*CF Playback Request Rejected Due to Max Playback Directories Reached Event ID.*
- #define CF\_CFDP\_NO\_MSG\_ERR\_EID 67  
*CF No Message Buffer Available Event ID.*
- #define CF\_CFDP\_CLOSE\_ERR\_EID 68  
*CF Close File Failed Event ID.*
- #define CF\_CFDP\_NO\_CHUNKLIST\_AVAIL\_EID 69  
*CF No chunklist available.*
- #define CF\_CFDP\_R\_REQUEST\_MD\_INF\_EID 70  
*CF Requesting RX Metadata Event ID.*
- #define CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID 71  
*CF Creating Temp File For RX Transaction.*

- #define CF\_CFDP\_R\_NAK\_LIMIT\_ERR\_EID 72  
*CF RX Transaction NAK Limit Reached Event ID.*
- #define CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_EID 73  
*CF RX Transaction ACK Limit Reached Event ID.*
- #define CF\_CFDP\_R\_CRC\_ERR\_EID 74  
*CF RX Transaction CRC Mismatch Event ID.*
- #define CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID 75  
*CF RX File Data PDU Seek Failed Event ID.*
- #define CF\_CFDP\_R\_SEEK\_CRC\_ERR\_EID 76  
*CF RX Class 2 CRC Seek Failed Event ID.*
- #define CF\_CFDP\_R\_WRITE\_ERR\_EID 77  
*CF RX File Data PDU Write Failed Event ID.*
- #define CF\_CFDP\_R\_SIZE\_MISMATCH\_ERR\_EID 78  
*CF RX End-Of-File PDU File Size Mismatch Event ID.*
- #define CF\_CFDP\_R\_PDU\_EOF\_ERR\_EID 79  
*CF Invalid End-Of-File PDU Event ID.*
- #define CF\_CFDP\_R\_CREAT\_ERR\_EID 80  
*CF RX Transaction File Create Failed Event ID.*
- #define CF\_CFDP\_R\_PDU\_FINACK\_ERR\_EID 81  
*CF Class 2 RX Transaction Invalid FIN-ACK PDU Event ID.*
- #define CF\_CFDP\_R\_EOF\_MD\_SIZE\_ERR\_EID 82  
*CF RX Class 2 Metadata PDU Size Mismatch Event ID.*
- #define CF\_CFDP\_R\_RENAME\_ERR\_EID 83  
*CF RX Class 2 Metadata PDU File Rename Failed Event ID.*
- #define CF\_CFDP\_R\_FILE\_RETAINED\_EID (84)  
*CF File retained.*
- #define CF\_CFDP\_R\_NOT\_RETAINED\_EID (85)  
*CF RX File not retained.*
- #define CF\_CFDP\_R\_READ\_ERR\_EID (86)  
*CF Class 2 CRC Read From File Failed Event ID.*
- #define CF\_CFDP\_R\_DC\_INV\_ERR\_EID 87  
*CF RX Invalid File Directive PDU Code Received Event ID.*
- #define CF\_CFDP\_R\_INACT\_TIMER\_ERR\_EID 88  
*CF RX Inactivity Timer Expired Event ID.*
- #define CF\_CFDP\_S\_START\_SEND\_INF\_EID 90  
*CF TX Initiated Event ID.*
- #define CF\_CFDP\_S\_SEEK\_FD\_ERR\_EID 91  
*CF TX File Data PDU Seek Failed Event ID.*
- #define CF\_CFDP\_S\_READ\_ERR\_EID 92  
*CF TX File Data PDU Read Failed Event ID.*
- #define CF\_CFDP\_S\_SEND\_FD\_ERR\_EID 93  
*CF TX File Data PDU Send Failed Event ID.*
- #define CF\_CFDP\_S\_ALREADY\_OPEN\_ERR\_EID 94  
*CF TX Metadata PDU File Already Open Event ID.*
- #define CF\_CFDP\_S\_OPEN\_ERR\_EID 95  
*CF TX Metadata PDU File Open Failed Event ID.*
- #define CF\_CFDP\_S\_SEEK\_END\_ERR\_EID 96

- #define CF\_CFDP\_S\_SEEK\_BEG\_ERR\_EID 97
  - CF TX Metadata PDU File Seek End Failed Event ID.*
- #define CF\_CFDP\_S\_SEND\_MD\_ERR\_EID 98
  - CF TX Metadata PDU Send Failed Event ID.*
- #define CF\_CFDP\_S\_INVALID\_SR\_ERR\_EID 100
  - CF TX Received NAK PDU Bad Segment Request Event ID.*
- #define CF\_CFDP\_S\_PDU\_NAK\_ERR\_EID 101
  - CF TX Received NAK PDU Invalid Event ID.*
- #define CF\_CFDP\_S\_PDU\_EOF\_ERR\_EID 102
  - CF TX Received EOF ACK PDU Invalid Event ID.*
- #define CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID 103
  - CF TX Received Early FIN PDU Event ID.*
- #define CF\_CFDP\_S\_DC\_INV\_ERR\_EID 104
  - CF Invalid TX File Directive PDU Code Event ID.*
- #define CF\_CFDP\_S\_NON\_FD\_PDU\_ERR\_EID 105
  - CF Received TX Non-File Directive PDU Event ID.*
- #define CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID 106
  - CF TX EOF PDU Send Limit Reached Event ID.*
- #define CF\_CFDP\_S\_INACT\_TIMER\_ERR\_EID 107
  - CF TX Inactivity Timer Expired Event ID.*
- #define CF\_CFDP\_S\_FILE\_MOVED\_EID (108)
  - CF TX File Moved.*
- #define CF\_CFDP\_S\_FILE\_REMOVED\_EID (109)
  - CF TX File Removed.*
- #define CF\_NOOP\_INF\_EID 110
  - CF NOOP Command Received Event ID.*
- #define CF\_RESET\_INF\_EID 111
  - CF Reset Counters Command Received Event ID.*
- #define CF\_CMD\_GETSET1\_INF\_EID 112
  - CF Set Parameter Command Received Event ID.*
- #define CF\_CMD\_GETSET2\_INF\_EID 113
  - CF Get Parameter Command Received Event ID.*
- #define CF\_CMD\_SUSPRES\_INF\_EID 114
  - CF Suspend/Resume Command Received Event ID.*
- #define CF\_CMD\_WQ\_INF\_EID 115
  - CF Write Queue Command Received Event ID.*
- #define CF\_CMD\_ENABLE\_ENGINE\_INF\_EID 116
  - CF Enable Engine Command Received Event ID.*
- #define CF\_CMD\_DISABLE\_ENGINE\_INF\_EID 117
  - CF Disable Engine Command Received Event ID.*
- #define CF\_CMD\_TX\_FILE\_INF\_EID 118
  - CF Transfer File Command Received Event ID.*
- #define CF\_CMD\_PLAYBACK\_DIR\_INF\_EID 119
  - CF Playback Directory Command Received Event ID.*
- #define CF\_CMD\_FREEZE\_INF\_EID 120
  - CF Freeze Command Received Event ID.*

- #define CF\_CMD\_THAW\_INF\_EID 121  
*CF Thaw Command Received Event ID.*
- #define CF\_CMD\_CANCEL\_INF\_EID 122  
*CF Cancel Command Received Event ID.*
- #define CF\_CMD\_ABANDON\_INF\_EID 123  
*CF Abandon Command Received Event ID.*
- #define CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID 124  
*CF Enable Dequeue Command Received Event ID.*
- #define CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID 125  
*CF Disable Dequeue Command Received Event ID.*
- #define CF\_CMD\_ENABLE\_POLLDIR\_INF\_EID 126  
*CF Enable Polldir Command Received Event ID.*
- #define CF\_CMD\_DISABLE\_POLLDIR\_INF\_EID 127  
*CF Disable Polldir Command Received Event ID.*
- #define CF\_CMD\_PURGE\_QUEUE\_INF\_EID 128  
*CF Purge Queue Command Received Event ID.*
- #define CF\_CMD\_RESET\_INVALID\_ERR\_EID 129  
*CF Reset Counters Command Invalid Event ID.*
- #define CF\_CMD\_CHAN\_PARAM\_ERR\_EID 130  
*CF Command Channel Invalid Event ID.*
- #define CF\_CMD\_TRANS\_NOT\_FOUND\_ERR\_EID 131  
*CF Command Transaction Invalid Event ID.*
- #define CF\_CMD\_TSN\_CHAN\_INVALID\_ERR\_EID 132  
*CF Command All Transaction Channel Invalid Event ID.*
- #define CF\_CMD\_SUSPRES\_SAME\_INF\_EID 133  
*CF Suspend/Resume Command For Single Transaction State Unchanged Event ID.*
- #define CF\_CMD\_SUSPRES\_CHAN\_ERR\_EID 134  
*CF Suspend/Resume Command No Matching Transaction Event ID.*
- #define CF\_CMD\_POLLDIR\_INVALID\_ERR\_EID 135  
*CF Enable/Disable Polling Directory Command Invalid Polling Directory Index Event ID.*
- #define CF\_CMD\_PURGE\_ARG\_ERR\_EID 136  
*CF Purge Queue Command Invalid Argument Event ID.*
- #define CF\_CMD\_WQ\_CHAN\_ERR\_EID 137  
*CF Write Queue Command Invalid Channel Event ID.*
- #define CF\_CMD\_WQ\_ARGS\_ERR\_EID 138  
*CF Write Queue Command Invalid Queue Event ID.*
- #define CF\_CMD\_WQ\_OPEN\_ERR\_EID 139  
*CF Write Queue Command File Open Failed Event ID.*
- #define CF\_CMD\_WQ\_WRITEQ\_RX\_ERR\_EID 140  
*CF Write Queue Command RX Active File Write Failed Event ID.*
- #define CF\_CMD\_WQ\_WRITEHIST\_RX\_ERR\_EID 141  
*CF Write Queue Command RX History File Write Failed Event ID.*
- #define CF\_CMD\_WQ\_WRITEQ\_TX\_ERR\_EID 142  
*CF Write Queue Command TX Active File Write Failed Event ID.*
- #define CF\_CMD\_WQ\_WRITEQ\_PEND\_ERR\_EID 143  
*CF Write Queue Command TX Pending File Write Failed Event ID.*
- #define CF\_CMD\_WQ\_WRITEHIST\_TX\_ERR\_EID 144

- #define CF\_CMD\_GETSET\_VALIDATE\_ERR\_EID 145  
*CF Write Queue Command TX History File Write Failed Event ID.*
- #define CF\_CMD\_GETSET\_PARAM\_ERR\_EID 146  
*CF Set Parameter Command Parameter Validation Failed Event ID.*
- #define CF\_CMD\_GETSET\_CHAN\_ERR\_EID 147  
*CF Set/Get Parameter Command Invalid Parameter ID Event ID.*
- #define CF\_CMD\_ENABLE\_ENGINE\_ERR\_EID 148  
*CF Enable Engine Command Failed Event ID.*
- #define CF\_CMD\_ENG\_ALREADY\_ENA\_INF\_EID 149  
*CF Enable Engine Command Engine Already Enabled Event ID.*
- #define CF\_CMD\_ENG\_ALREADY\_DIS\_INF\_EID 150  
*CF Disable Engine Command Engine Already Disabled Event ID.*
- #define CF\_CMD\_LEN\_ERR\_EID 151  
*CF Command Length Verification Failed Event ID.*
- #define CF\_CC\_ERR\_EID 152  
*CF Command Code Invalid Event ID.*
- #define CF\_CMD\_WHIST\_WRITE\_ERR\_EID 153  
*CF Write Entry To File Failed Event ID.*
- #define CF\_CMD\_BAD\_PARAM\_ERR\_EID 154  
*CF Playback Dir Or TX File Command Bad Parameter Event ID.*
- #define CF\_CMD\_CANCEL\_CHAN\_ERR\_EID 155  
*CF Cancel Command No Matching Transaction Event ID.*
- #define CF\_CMD\_ABANDON\_CHAN\_ERR\_EID 156  
*CF Abandon Command No Matching Transaction Event ID.*
- #define CF\_CMD\_TX\_FILE\_ERR\_EID 157  
*CF Transfer File Command Failed Event ID.*
- #define CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID 158  
*CF Playback Directory Command Failed Event ID.*
- #define CF\_CMD\_FREEZE\_ERR\_EID 159  
*CF Freeze Command Failed Event ID.*
- #define CF\_CMD\_THAW\_ERR\_EID 160  
*CF Thaw Command Failed Event ID.*
- #define CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID 161  
*CF Enable Dequeue Command Failed Event ID.*
- #define CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID 162  
*CF Disable Dequeue Command Failed Event ID.*
- #define CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID 163  
*CF Enable Polldir Command Failed Event ID.*
- #define CF\_CMD\_DISABLE\_POLLDIR\_ERR\_EID 164  
*CF Disable Polldir Command Failed Event ID.*
- #define CF\_CMD\_PURGE\_QUEUE\_ERR\_EID 165  
*CF Purge Queue Command Failed Event ID.*
- #define CF\_EID\_INF\_CFDP\_BUF\_EXCEED 166  
*CF Move Path Length Verification Too Long Event ID.*

### 12.25.1 Detailed Description

The CF Application event id definition header file

## 12.26 apps/cf/fsw/inc(cf\_fcncodes.h File Reference

```
#include "cf_fcncode_values.h"
```

### Macros

- #define CF\_NOOP\_CC CF\_CCVAL(NOOP)  
*No-op.*
- #define CF\_RESET\_CC CF\_CCVAL(RESET\_COUNTERS)  
*Reset counters.*
- #define CF\_TX\_FILE\_CC CF\_CCVAL(TX\_FILE)  
*Transmit file.*
- #define CF\_PLAYBACK\_DIR\_CC CF\_CCVAL(PLAYBACK\_DIR)  
*Playback a directory.*
- #define CF\_FREEZE\_CC CF\_CCVAL(FREEZE)  
*Freeze a channel.*
- #define CF\_THAW\_CC CF\_CCVAL(THAW)  
*Thaw a channel.*
- #define CF\_SUSPEND\_CC CF\_CCVAL(SUSPEND)  
*Suspend a transaction.*
- #define CF\_RESUME\_CC CF\_CCVAL(RESUME)  
*Resume a transaction.*
- #define CF\_CANCEL\_CC CF\_CCVAL(CANCEL)  
*Cancel a transaction.*
- #define CF\_ABANDON\_CC CF\_CCVAL(ABANDON)  
*Abandon a transaction.*
- #define CF\_SET\_PARAM\_CC CF\_CCVAL(SET\_PARAM)  
*Set parameter.*
- #define CF\_GET\_PARAM\_CC CF\_CCVAL(GET\_PARAM)  
*Get parameter.*
- #define CF\_WRITE\_QUEUE\_CC CF\_CCVAL(WRITE\_QUEUE)  
*Write queue.*
- #define CF\_ENABLE\_DEQUEUE\_CC CF\_CCVAL(ENABLE\_DEQUEUE)  
*Enable dequeue.*
- #define CF\_DISABLE\_DEQUEUE\_CC CF\_CCVAL(DISABLE\_DEQUEUE)  
*Disable dequeue.*
- #define CF\_ENABLE\_DIR\_POLLING\_CC CF\_CCVAL(ENABLE\_DIR\_POLLING)  
*Enable directory polling.*
- #define CF\_DISABLE\_DIR\_POLLING\_CC CF\_CCVAL(DISABLE\_DIR\_POLLING)  
*Disable directory polling.*
- #define CF\_PURGE\_QUEUE\_CC CF\_CCVAL(PURGE\_QUEUE)  
*Purge queue.*
- #define CF\_ENABLE\_ENGINE\_CC CF\_CCVAL(ENABLE\_ENGINE)  
*Enable engine.*

- #define CF\_DISABLE\_ENGINE\_CC CF\_CVAL(DISABLE\_ENGINE)  
*Disable engine.*
- #define CF\_NUM\_COMMANDS 24  
*Command code limit used for validity check and array sizing.*

### 12.26.1 Detailed Description

Specification for the CFS CFDP (CF) command function codes

#### Note

This file should be strictly limited to the command/function code (CC) macro definitions. Other definitions such as enums, typedefs, or other macros should be placed in the msgdefs.h or msg.h files.

## 12.27 apps/cf/fsw/inc/cf\_interface\_cfg.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cf_interface_cfg_values.h"
```

#### Macros

- #define CF\_NUM\_CHANNELS CF\_INTERFACE\_CVAL(NUM\_CHANNELS)  
*Number of channels.*
- #define DEFAULT\_CF\_NUM\_CHANNELS 2
- #define CF\_NAK\_MAX\_SEGMENTS CF\_INTERFACE\_CVAL(NAK\_MAX\_SEGMENTS)  
*Max NAK segments supported in a NAK PDU.*
- #define DEFAULT\_CF\_NAK\_MAX\_SEGMENTS 58
- #define CF\_MAX\_POLLING\_DIR\_PER\_CHAN CF\_INTERFACE\_CVAL(MAX\_POLLING\_DIR\_PER\_CHAN)  
*Max number of polling directories per channel.*
- #define DEFAULT\_CF\_MAX\_POLLING\_DIR\_PER\_CHAN 5
- #define CF\_MAX\_PDU\_SIZE CF\_INTERFACE\_CVAL(MAX\_PDU\_SIZE)  
*Max PDU size.*
- #define DEFAULT\_CF\_MAX\_PDU\_SIZE 512
- #define CF\_FILENAME\_MAX\_NAME CF\_INTERFACE\_CVAL(FILENAME\_MAX\_NAME)  
*Maximum file name length.*
- #define DEFAULT\_CF\_FILENAME\_MAX\_NAME CFE\_MISSION\_MAX\_FILE\_LEN
- #define CF\_FILENAME\_MAX\_LEN CF\_INTERFACE\_CVAL(FILENAME\_MAX\_LEN)  
*Max filename and path length.*
- #define DEFAULT\_CF\_FILENAME\_MAX\_LEN CFE\_MISSION\_MAX\_PATH\_LEN
- #define CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES CF\_INTERFACE\_CVAL(PDU\_ENCAPSULATION←\_EXTRA\_TRAILING\_BYTES)  
*Number of trailing bytes to add to CFDP PDU.*
- #define DEFAULT\_CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES 0

### 12.27.1 Detailed Description

CFS CFDP (CF) Application Mission Configuration Header File

#### 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.

## 12.28 apps/cf/fsw/inc(cf\_internal\_cfg.h File Reference

```
#include "cf_mission_cfg.h"
#include "cf_internal_cfg_values.h"
```

### Macros

- #define CF\_PIPE\_DEPTH CF\_INTERNAL\_CFGVAL(PIPE\_DEPTH)  
*Application Pipe Depth.*
- #define DEFAULT\_CF\_PIPE\_DEPTH 32
- #define CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN CF\_INTERNAL\_CFGVAL(MAX\_COMMANDED←\_PLAYBACK\_FILES\_PER\_CHAN)  
*Number of max commanded playback files per chan.*
- #define DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN 10
- #define CF\_MAX\_SIMULTANEOUS\_RX CF\_INTERNAL\_CFGVAL(MAX\_SIMULTANEOUS\_RX)  
*Max number of simultaneous file receives.*
- #define DEFAULT\_CF\_MAX\_SIMULTANEOUS\_RX 5
- #define CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN CF\_INTERNAL\_CFGVAL(MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN)  
*Max number of commanded playback directories per channel.*
- #define DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN 2
- #define CF\_NUM\_HISTORIES\_PER\_CHANNEL CF\_INTERNAL\_CFGVAL(NUM\_HISTORIES\_PER CHANNEL)  
*Number of histories per channel.*
- #define DEFAULT\_CF\_NUM\_HISTORIES\_PER\_CHANNEL 256
- #define CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK CF\_INTERNAL\_CFGVAL(NUM\_TRANSACTIONS PER\_PLAYBACK)  
*Number of transactions per playback directory.*
- #define DEFAULT\_CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK 5
- #define CF\_CONFIG\_TABLE\_NAME CF\_INTERNAL\_CFGVAL(CONFIG\_TABLE\_NAME)  
*Name of the CF Configuration Table.*
- #define DEFAULT\_CF\_CONFIG\_TABLE\_NAME "config\_table"
- #define CF\_CONFIG\_TABLE\_FILENAME CF\_INTERNAL\_CFGVAL(CONFIG\_TABLE\_FILENAME)  
*CF Configuration Table Filename.*
- #define DEFAULT\_CF\_CONFIG\_TABLE\_FILENAME "/cf(cf\_def\_config.tbl"
- #define CF\_R2\_CRC\_CHUNK\_SIZE CF\_INTERNAL\_CFGVAL(R2\_CRC\_CHUNK\_SIZE)  
*R2 CRC calc chunk size.*
- #define DEFAULT\_CF\_R2\_CRC\_CHUNK\_SIZE 1024
- #define CF\_RCVMSG\_TIMEOUT CF\_INTERNAL\_CFGVAL(RCVMSG\_TIMEOUT)  
*Number of milliseconds to wait for a SB message.*
- #define DEFAULT\_CF\_RCVMSG\_TIMEOUT 100
- #define CF\_STARTUP\_SEM\_MAX\_RETRIES CF\_INTERNAL\_CFGVAL(STARTUP\_SEM\_MAX\_RETRIES)  
*Limits the number of retries to obtain the CF throttle sem.*
- #define DEFAULT\_CF\_STARTUP\_SEM\_MAX\_RETRIES 25
- #define CF\_STARTUP\_SEM\_TASK\_DELAY CF\_INTERNAL\_CFGVAL(STARTUP\_SEM\_TASK\_DELAY)  
*Number of milliseconds to wait if CF throttle sem is not available.*
- #define DEFAULT\_CF\_STARTUP\_SEM\_TASK\_DELAY 100

### 12.28.1 Detailed Description

CFS CFDP (CF) Application Platform Configuration Header File

#### 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.

## 12.29 apps/cf/fsw/inc/cf\_perfids.h File Reference

#### Macros

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

### 12.29.1 Detailed Description

Define CF Performance IDs

## 12.30 apps/cf/fsw/inc/cf\_topicids.h File Reference

```
#include "cf_topicid_values.h"
```

## Macros

- #define CFE\_MISSION\_CF\_CMD\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CMD)  
• #define DEFAULT\_CFE\_MISSION\_CF\_CMD\_TOPICID 0xB3  
*Message ID for commands.*
- #define CFE\_MISSION\_CF\_SEND\_HK\_TOPICID CFE\_MISSION\_CF\_TIDVAL(SEND\_HK)  
• #define DEFAULT\_CFE\_MISSION\_CF\_SEND\_HK\_TOPICID 0xB4  
*Message ID to request housekeeping telemetry.*
- #define CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID CFE\_MISSION\_CF\_TIDVAL(WAKE\_UP)  
• #define DEFAULT\_CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID 0xB5  
*Message ID for waking up the processing cycle.*
- #define CFE\_MISSION\_CF\_HK\_TLM\_TOPICID CFE\_MISSION\_CF\_TIDVAL(HK\_TLM)  
• #define DEFAULT\_CFE\_MISSION\_CF\_HK\_TLM\_TOPICID 0xB0  
*Message ID for housekeeping telemetry.*
- #define CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID CFE\_MISSION\_CF\_TIDVAL(EOT\_TLM)  
• #define DEFAULT\_CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID 0xB3  
*Message ID for end of transaction telemetry.*
- #define CFE\_MISSION\_CF\_CH0\_TX\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CH0\_TX)  
• #define DEFAULT\_CFE\_MISSION\_CF\_CH0\_TX\_TOPICID 0xB4  
• #define CFE\_MISSION\_CF\_CH1\_TX\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CH1\_TX)  
• #define DEFAULT\_CFE\_MISSION\_CF\_CH1\_TX\_TOPICID 0xB5  
• #define CFE\_MISSION\_CF\_CH0\_RX\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CH0\_RX)  
• #define DEFAULT\_CFE\_MISSION\_CF\_CH0\_RX\_TOPICID 0xB6  
• #define CFE\_MISSION\_CF\_CH1\_RX\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CH1\_RX)  
• #define DEFAULT\_CFE\_MISSION\_CF\_CH1\_RX\_TOPICID 0xB7

### 12.30.1 Detailed Description

CFDP (CF) Application Topic IDs

### 12.30.2 Macro Definition Documentation

**12.30.2.1 CFE\_MISSION\_CF\_CH0\_RX\_TOPICID** #define CFE\_MISSION\_CF\_CH0\_RX\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CH0↔\_RX)

Definition at line 64 of file cf\_topicids.h.

**12.30.2.2 CFE\_MISSION\_CF\_CH0\_TX\_TOPICID** #define CFE\_MISSION\_CF\_CH0\_TX\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CH0↔\_TX)

Definition at line 58 of file cf\_topicids.h.

**12.30.2.3 CFE\_MISSION\_CF\_CH1\_RX\_TOPICID** #define CFE\_MISSION\_CF\_CH1\_RX\_TOPICID CFE\_MISSION\_CF\_TIDVAL(CH1↔\_RX)

Definition at line 67 of file cf\_topicids.h.

**12.30.2.4 CFE\_MISSION\_CF\_CH1\_TX\_TOPICID** #define CFE\_MISSION\_CF\_CH1\_TX\_TOPICID CFE\_MISSION\_CF\_TIDVAL (CH1↔\_TX)

Definition at line 61 of file cf\_topicids.h.

**12.30.2.5 CFE\_MISSION\_CF\_CMD\_TOPICID** #define CFE\_MISSION\_CF\_CMD\_TOPICID CFE\_MISSION\_CF\_TIDVAL (CMD)

Definition at line 34 of file cf\_topicids.h.

**12.30.2.6 CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID** #define CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID CFE\_MISSION\_CF\_TIDVAL (EOT↔\_TLM)

Definition at line 46 of file cf\_topicids.h.

**12.30.2.7 CFE\_MISSION\_CF\_HK\_TLM\_TOPICID** #define CFE\_MISSION\_CF\_HK\_TLM\_TOPICID CFE\_MISSION\_CF\_TIDVAL (HK↔\_TLM)

Definition at line 43 of file cf\_topicids.h.

**12.30.2.8 CFE\_MISSION\_CF\_SEND\_HK\_TOPICID** #define CFE\_MISSION\_CF\_SEND\_HK\_TOPICID CFE\_MISSION\_CF\_TIDVAL (SEND↔\_HK)

Definition at line 37 of file cf\_topicids.h.

**12.30.2.9 CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID** #define CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID CFE\_MISSION\_CF\_TIDVAL (WAKE↔\_UP)

Definition at line 40 of file cf\_topicids.h.

**12.30.2.10 DEFAULT\_CFE\_MISSION\_CF\_CH0\_RX\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_CH0\_RX\_TOPICID 0xB6

Definition at line 65 of file cf\_topicids.h.

**12.30.2.11 DEFAULT\_CFE\_MISSION\_CF\_CH0\_TX\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_CH0\_TX\_TOPICID 0xB4

Definition at line 59 of file cf\_topicids.h.

**12.30.2.12 DEFAULT\_CFE\_MISSION\_CF\_CH1\_RX\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_CH1\_RX\_TOPICID 0xB7

Definition at line 68 of file cf\_topicids.h.

**12.30.2.13 DEFAULT\_CFE\_MISSION\_CF\_CH1\_TX\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_CH1\_TX\_TOPICID 0xB5

Definition at line 62 of file cf\_topicids.h.

**12.30.2.14 DEFAULT\_CFE\_MISSION\_CF\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_CMD\_TOPICID 0xB3

Message ID for commands.

Definition at line 35 of file cf\_topicids.h.

**12.30.2.15 DEFAULT\_CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID 0xB3

Message ID for end of transaction telemetry.

Definition at line 47 of file cf\_topicids.h.

**12.30.2.16 DEFAULT\_CFE\_MISSION\_CF\_HK\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_HK\_TLM\_TOPICID 0xB0

Message ID for housekeeping telemetry.

Definition at line 44 of file cf\_topicids.h.

**12.30.2.17 DEFAULT\_CFE\_MISSION\_CF\_SEND\_HK\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_SEND\_HK\_TOPICID 0xB4

Message ID to request housekeeping telemetry.

Definition at line 38 of file cf\_topicids.h.

**12.30.2.18 DEFAULT\_CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID** #define DEFAULT\_CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID 0xB5

Message ID for waking up the processing cycle.

Definition at line 41 of file cf\_topicids.h.

## 12.31 apps/cf/fsw/src(cf\_app.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_version.h"
#include "cf_dispatch.h"
#include "cf_tbl.h"
#include <string.h>
```

### Functions

- void [CF\\_CheckTables](#) (void)
 

*Checks to see if a table update is pending, and perform it.*
- [CFE\\_Status\\_t CF\\_ValidateConfigTable](#) (void \*tbl\_ptr)
 

*Validation function for config table.*
- [CFE\\_Status\\_t CF\\_TableInit](#) (void)
 

*Load the table on application start.*
- [CFE\\_Status\\_t CF\\_AppInit](#) (void)

*CF app init function.*

- void [CF\\_AppMain](#) (void)

*CF app entry point.*

## Variables

- [CF\\_AppData\\_t CF\\_AppData](#)

*Singleton instance of the application global data.*

### 12.31.1 Detailed Description

The CF Application main application source file

This file contains the functions that initialize the application and link all logic and functionality to the CFS.

### 12.31.2 Function Documentation

#### 12.31.2.1 [CF\\_AppInit\(\)](#) [CFE\\_Status\\_t](#) CF\_AppInit (

    void )

CF app init function.

##### Description

Initializes all aspects of the CF application. Messages, pipes, events, table, and the CFDP engine.

##### Assumptions, External Events, and Notes:

This must only be called once.

##### Return values

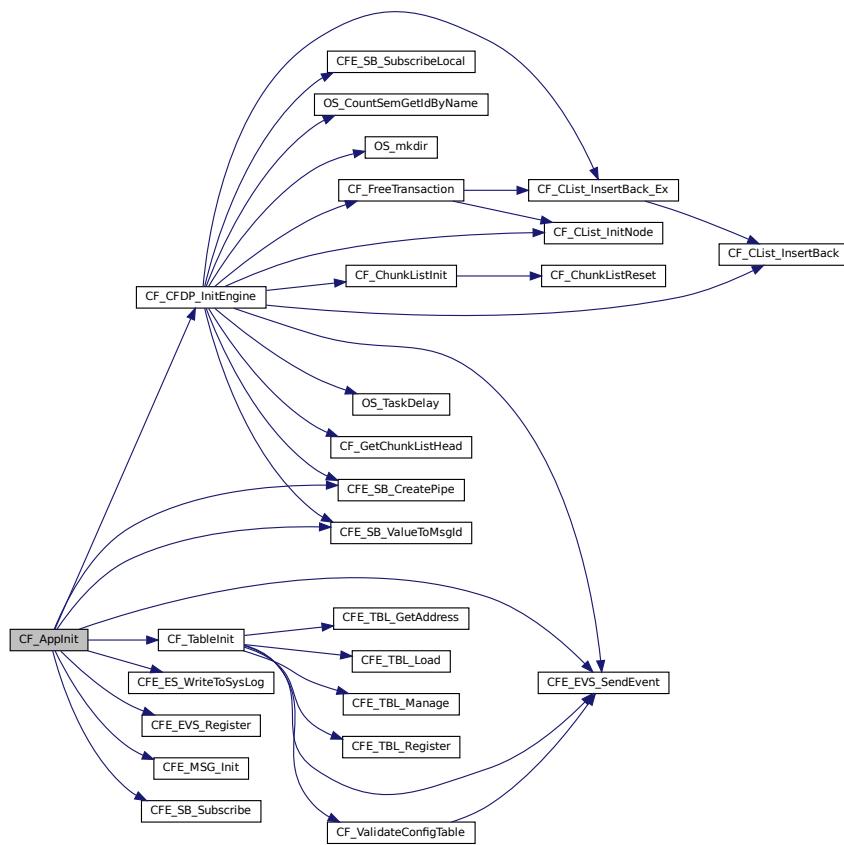
<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

Definition at line 189 of file cf\_app.c.

References CF\_AppData, CF\_CFDP\_InitEngine(), CF\_CMD\_MID, CF\_CR\_PIPE\_ERR\_EID, CF\_HK\_TLM\_MID, CF\_INIT\_INF\_EID, CF\_MAJOR\_VERSION, CF\_MINOR\_VERSION, CF\_MISSION\_REV, CF\_PIPE\_DEPTH, CF\_PIPE\_NAME, CF\_REVISION, CF\_SEND\_HK\_MID, CF\_TableInit(), CF\_WAKE\_UP\_MID, CFE\_ES\_RunStatus\_APP\_RUN, CFE\_ES\_WriteToSysLog(), CFE\_EVS\_EventFilter\_BINARY, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_Register(), CFE\_EVS\_SendEvent(), CFE\_MSG\_Init(), CFE\_SB\_CreatePipe(), CFE\_SB\_Subscribe(), CFE\_SB\_ValueToMsgId(), CFE\_SUCCESS, CF\_AppData\_t::CmdPipe, CF\_AppData\_t::hk, CF\_AppData\_t::RunStatus, and CF\_HkPacket::TelemetryHeader.

Referenced by CF\_AppMain().

Here is the call graph for this function:



**12.31.2.2 CF\_AppMain()** void CF\_AppMain ( void )

CF app entry point.

#### Description

Main entry point of CF application. Calls the init function and manages the app run loop.

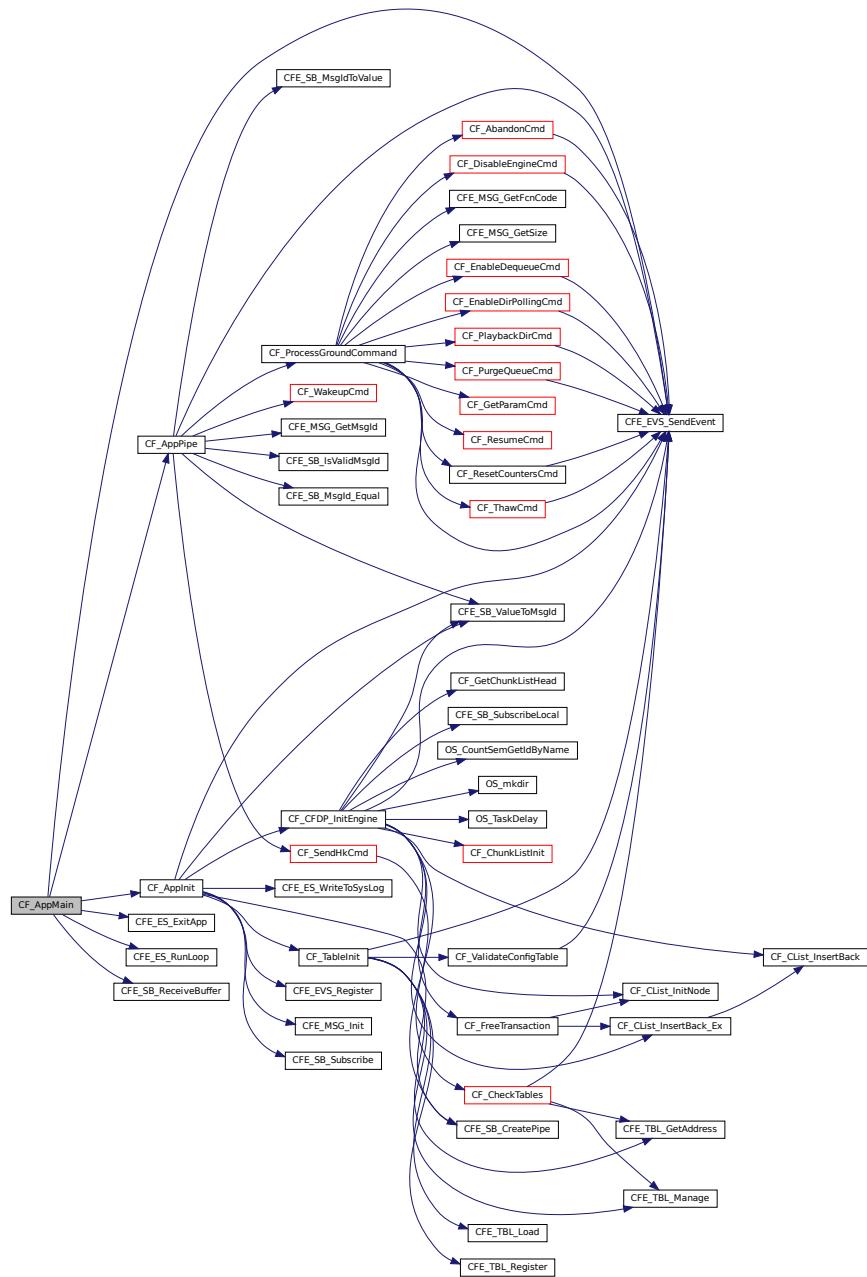
#### Assumptions, External Events, and Notes:

This must only be called once.

Definition at line 256 of file cf\_app.c.

References CF\_AppData, CF\_AppInit(), CF\_AppPipe(), CF\_INIT\_MSG\_RECV\_ERR\_EID, CF\_PERF\_ID\_APPMAIN, CF\_RCVMSG\_TIMEOUT, CFE\_ES\_ExitApp(), CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_ES\_RunLoop(), CFE\_ES\_RunStatus\_APP\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SB\_NO\_MESSAGE, CFE\_SB\_ReceiveBuffer(), CFE\_SB\_TIME\_OUT, CFE\_SUCCESS, CF\_AppData\_t::CmdPipe, and CF\_AppData\_t::RunStatus.

Here is the call graph for this function:



**12.31.2.3 CF\_CheckTables()** void CF\_CheckTables ( void )

Checks to see if a table update is pending, and perform it.

**Description**

Updates the table if the engine is disabled.

**Assumptions, External Events, and Notes:**

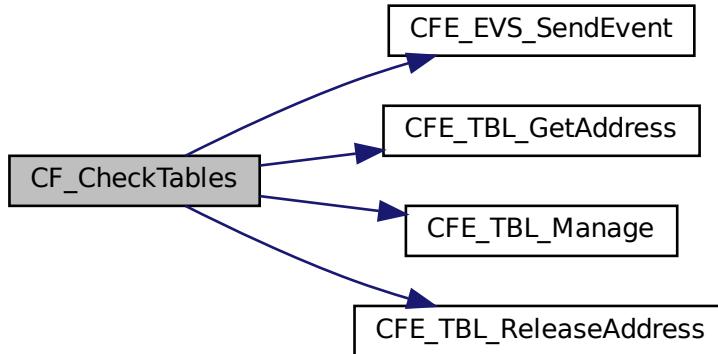
None

Definition at line 48 of file cf\_app.c.

References CF\_AppData, CF\_INIT\_TBL\_CHECK\_GA\_ERR\_EID, CF\_INIT\_TBL\_CHECK\_MAN\_ERR\_EID, CF\_INIT\_TBL\_CHECK\_REL\_ERR\_EID, CFE\_ES\_RunStatus\_APP\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_GetAddress(), CFE\_TBL\_Manage(), CFE\_TBL\_ReleaseAddress(), CF\_AppData\_t::config\_handle, CF\_AppData\_t::config\_table, CF\_Engine::enabled, CF\_AppData\_t::engine, and CF\_AppData\_t::RunStatus.

Referenced by CF\_SendHkCmd().

Here is the call graph for this function:



#### 12.31.2.4 CF\_TableInit() `CFE_Status_t CF_TableInit ( void )`

Load the table on application start.

**Assumptions, External Events, and Notes:**

None

**Return values**

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

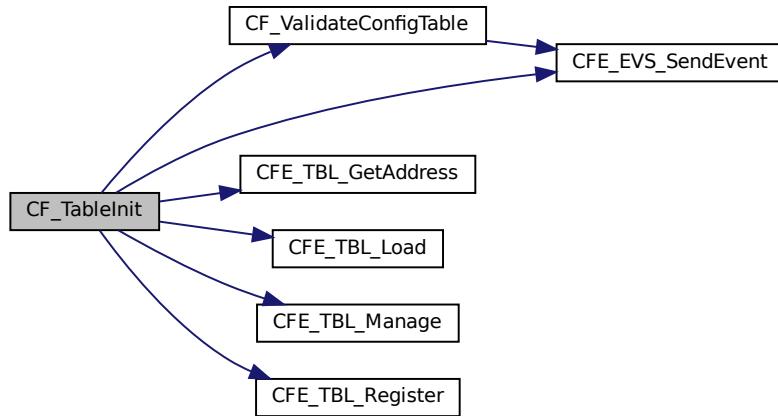
Definition at line 134 of file cf\_app.c.

References CF\_AppData, CF\_CONFIG\_TABLE\_FILENAME, CF\_CONFIG\_TABLE\_NAME, CF\_INIT\_TBL\_

GETADDR\_ERR\_EID, CF\_INIT\_TBL\_LOAD\_ERR\_EID, CF\_INIT\_TBL\_MANAGE\_ERR\_EID, CF\_INIT\_TBL\_REG\_ERR\_EID, CF\_ValidateConfigTable(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_GetAddress(), CFE\_TBL\_INFO\_UPDATED, CFE\_TBL\_Load(), CFE\_TBL\_Manage(), CFE\_TBL\_OPT\_LOAD\_DUMP, CFE\_TBL\_OPT\_SNGL\_BUFFER, CFE\_TBL\_Register(), CFE\_TBL\_SRC\_FILE, CF\_AppData\_t::config\_handle, and CF\_AppData\_t::config\_table.

Referenced by CF\_AppInit().

Here is the call graph for this function:



**12.31.2.5 CF\_ValidateConfigTable()** `CFE_Status_t CF_ValidateConfigTable ( void * tbl_ptr )`

Validation function for config table.

#### Description

Checks that the config table being loaded has correct data.

#### Assumptions, External Events, and Notes:

None

#### Return values

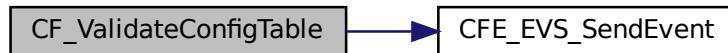
<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CFE_STATUS_VALIDATION_FAILURE</code>	if the config table fails one of the validation checks

Definition at line 101 of file cf\_app.c.

References CF\_INIT\_CRC\_ALIGN\_ERR\_EID, CF\_INIT\_OUTGOING\_SIZE\_ERR\_EID, CF\_INIT\_TPS\_ERR\_EID, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_STATUS\_VALIDATION\_FAILURE, CFE\_SUCCESS, CF\_ConfigTable::outgoing\_file\_chunk\_size, CF\_ConfigTable::rx\_crc\_calc\_bytes\_per\_wakeup, and CF\_ConfigTable::ticks\_per\_second.

Referenced by CF\_TableInit().

Here is the call graph for this function:



### 12.31.3 Variable Documentation

#### 12.31.3.1 CF\_AppData CF\_AppData\_t CF\_AppData

Singleton instance of the application global data.

Definition at line 40 of file cf\_app.c.

Referenced by CF\_AbandonCmd(), CF\_AppInit(), CF\_AppMain(), CF\_AppPipe(), CF\_CancelCmd(), CF\_CFDP\_AppendTlv(), CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_CycleEngine(), CF\_CFDP\_DisableEngine(), CF\_CFDP\_GetTempName(), CF\_CFDP\_InitEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_PlaybackDir(), CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_ReceiveMessage(), CF\_CFDP\_ReceivePdu(), CF\_CFDP\_RecvDrop(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvHold(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvPh(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_Send(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_StartRxTransaction(), CF\_CFDP\_TxFile(), CF\_CFDP\_TxFile\_Initiate(), CF\_CheckTables(), CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DisableEngineCmd(), CF\_DoEnableDisableDequeue(), CF\_DoEnableDisablePolldir(), CF\_DoFreezeThaw(), CF\_DoPurgeQueue(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(), CF\_FindTransactionBySequenceNumberAllChannels(), CF\_FreeTransaction(), CF\_FreezeCmd(), CF\_GetChannelFromTxn(), CF\_SetParamCmd(), CF\_InsertSortPrio(), CF\_MoveTransaction(), CF\_NoopCmd(), CF\_PlaybackDirCmd(), CF\_ProcessGroundCommand(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_SendHkCmd(), CF\_TableInit(), CF\_ThawCmd(), CF\_Timer\_Sec2Ticks(), CF\_TraverseAllTransactions\_All\_Channels(), CF\_TsnChanAction(), CF\_TxFileCmd(), CF\_ValidateMaxOutgoingCmd(), and CF\_WriteQueueCmd().

## 12.32 apps/cf/fsw/src(cf\_app.h File Reference

```

#include "cfe.h"
#include "cf_msg.h"
#include "cf_tbl.h"
#include "cf_msgids.h"
#include "cf_tbldefs.h"
#include "cf_mission_cfg.h"
#include "cf_platform_cfg.h"
#include "cf_cfdp.h"
#include "cf_clist.h"

```

## Data Structures

- struct `CF_AppData_t`

*The CF application global state structure.*

## Macros

- `#define CF_PIPE_NAME ("CF_CMD_PIPE")`  
*The name of the application command pipe for CF.*
- `#define CF_CHANNEL_PIPE_PREFIX ("CF_CHAN_")`  
*A common prefix for all data pipes for CF.*
- `#define CF_FILENAME_TRUNCATED '$'`  
*Marker used to flag filenames suspected of being truncated.*

## CF Error Codes

- `#define CF_ERROR -1`  
*Generic CF error return code.*
- `#define CF_PDU_METADATA_ERROR -2`  
*Invalid metadata PDU.*
- `#define CF_SHORT_PDU_ERROR -3`  
*PDU too short.*
- `#define CF_REC_PDU_FSIZE_MISMATCH_ERROR -4`  
*Receive PDU: EOF file size mismatch.*
- `#define CF_REC_PDU_BAD_EOF_ERROR -5`  
*Receive PDU: Invalid EOF packet.*
- `#define CF_SEND_PDU_NO_BUF_AVAIL_ERROR -6`  
*Send PDU: No send buffer available, throttling limit reached.*
- `#define CF_SEND_PDU_ERROR -7`  
*Send PDU: Send failed.*

## Functions

- `void CF_CheckTables (void)`  
*Checks to see if a table update is pending, and perform it.*
- `CFE_Status_t CF_ValidateConfigTable (void *tbl_ptr)`  
*Validation function for config table.*
- `CFE_Status_t CF_TableInit (void)`  
*Load the table on application start.*
- `CFE_Status_t CF_AppInit (void)`  
*CF app init function.*
- `void CF_AppMain (void)`  
*CF app entry point.*

## Variables

- `CF_AppData_t CF_AppData`  
*Singleton instance of the application global data.*

### 12.32.1 Detailed Description

The CF Application main application header file

### 12.32.2 Macro Definition Documentation

#### 12.32.2.1 CF\_CHANNEL\_PIPE\_PREFIX #define CF\_CHANNEL\_PIPE\_PREFIX ("CF\_CHAN\_")

A common prefix for all data pipes for CF.

Definition at line 67 of file cf\_app.h.

#### 12.32.2.2 CF\_ERROR #define CF\_ERROR -1

Generic CF error return code.

Definition at line 50 of file cf\_app.h.

#### 12.32.2.3 CF\_FILENAME\_TRUNCATED #define CF\_FILENAME\_TRUNCATED '\$'

Marker used to flag filenames suspected of being truncated.

Definition at line 72 of file cf\_app.h.

#### 12.32.2.4 CF\_PDU\_METADATA\_ERROR #define CF\_PDU\_METADATA\_ERROR -2

Invalid metadata PDU.

Definition at line 51 of file cf\_app.h.

#### 12.32.2.5 CF\_PIPE\_NAME #define CF\_PIPE\_NAME ("CF\_CMD\_PIPE")

The name of the application command pipe for CF.

Definition at line 62 of file cf\_app.h.

#### 12.32.2.6 CF\_REC\_PDU\_BAD\_EOF\_ERROR #define CF\_REC\_PDU\_BAD\_EOF\_ERROR -5

Receive PDU: Invalid EOF packet.

Definition at line 54 of file cf\_app.h.

#### 12.32.2.7 CF\_REC\_PDU\_FSIZE\_MISMATCH\_ERROR #define CF\_REC\_PDU\_FSIZE\_MISMATCH\_ERROR -4

Receive PDU: EOF file size mismatch.

Definition at line 53 of file cf\_app.h.

#### 12.32.2.8 CF\_SEND\_PDU\_ERROR #define CF\_SEND\_PDU\_ERROR -7

Send PDU: Send failed.

Definition at line 56 of file cf\_app.h.

#### 12.32.2.9 CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR #define CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR -6

Send PDU: No send buffer available, throttling limit reached.

Definition at line 55 of file cf\_app.h.

#### 12.32.2.10 CF\_SHORT\_PDU\_ERROR #define CF\_SHORT\_PDU\_ERROR -3

PDU too short.

Definition at line 52 of file cf\_app.h.

### 12.32.3 Function Documentation

**12.32.3.1 CF\_AppInit()** `CFE_Status_t CF_AppInit (`  
    `void )`

CF app init function.

#### Description

Initializes all aspects of the CF application. Messages, pipes, events, table, and the CFDP engine.

#### Assumptions, External Events, and Notes:

This must only be called once.

#### Return values

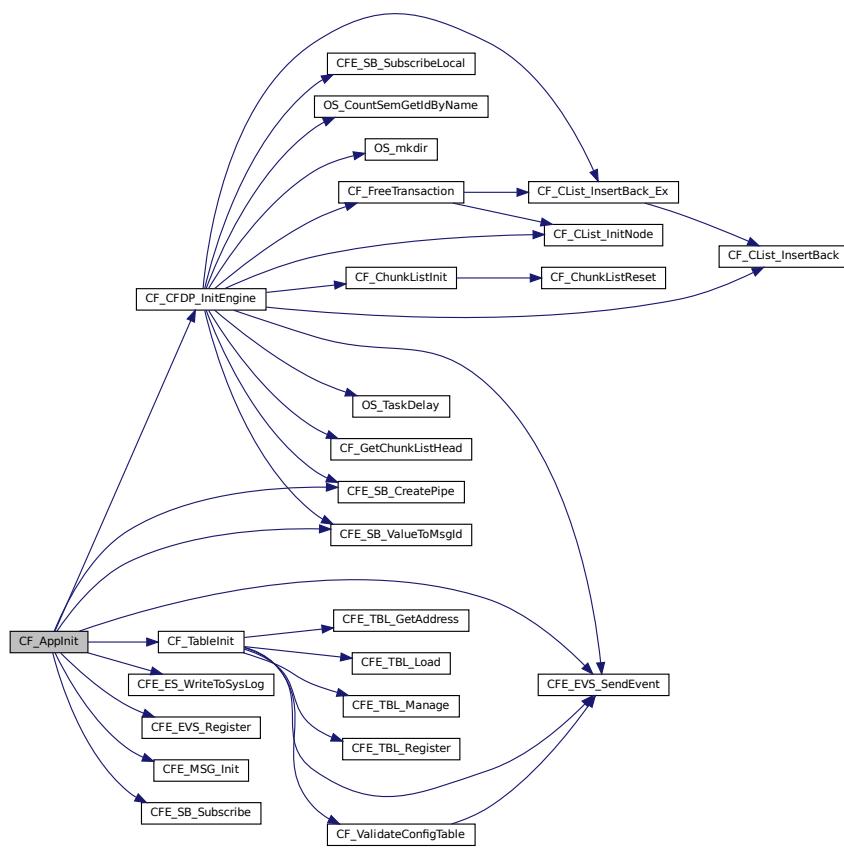
<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

Definition at line 189 of file cf\_app.c.

References CF\_AppData, CF\_CFDP\_InitEngine(), CF\_CMD\_MID, CF\_CR\_PIPE\_ERR\_EID, CF\_HK\_TLM\_MID, CF\_INIT\_INF\_EID, CF\_MAJOR\_VERSION, CF\_MINOR\_VERSION, CF\_MISSION\_REV, CF\_PIPE\_DEPTH, CF\_PIPE\_NAME, CF\_REVISION, CF\_SEND\_HK\_MID, CF\_TableInit(), CF\_WAKE\_UP\_MID, CFE\_ES\_RunStatus\_APP\_RUN, CFE\_ES\_WriteToSysLog(), CFE\_EVS\_EventFilter\_BINARY, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_Register(), CFE\_EVS\_SendEvent(), CFE\_MSG\_Init(), CFE\_SB\_CreatePipe(), CFE\_SB\_Subscribe(), CFE\_SB\_ValueToMsgId(), CFE\_SUCCESS, CF\_AppData\_t::CmdPipe, CF\_AppData\_t::hk, CF\_AppData\_t::RunStatus, and CF\_HkPacket::TelemetryHeader.

Referenced by CF\_AppMain().

Here is the call graph for this function:



**12.32.3.2 CF\_AppMain()** void CF\_AppMain ( void )

## CF app entry point.

## Description

Main entry point of CF application. Calls the init function and manages the app run loop.

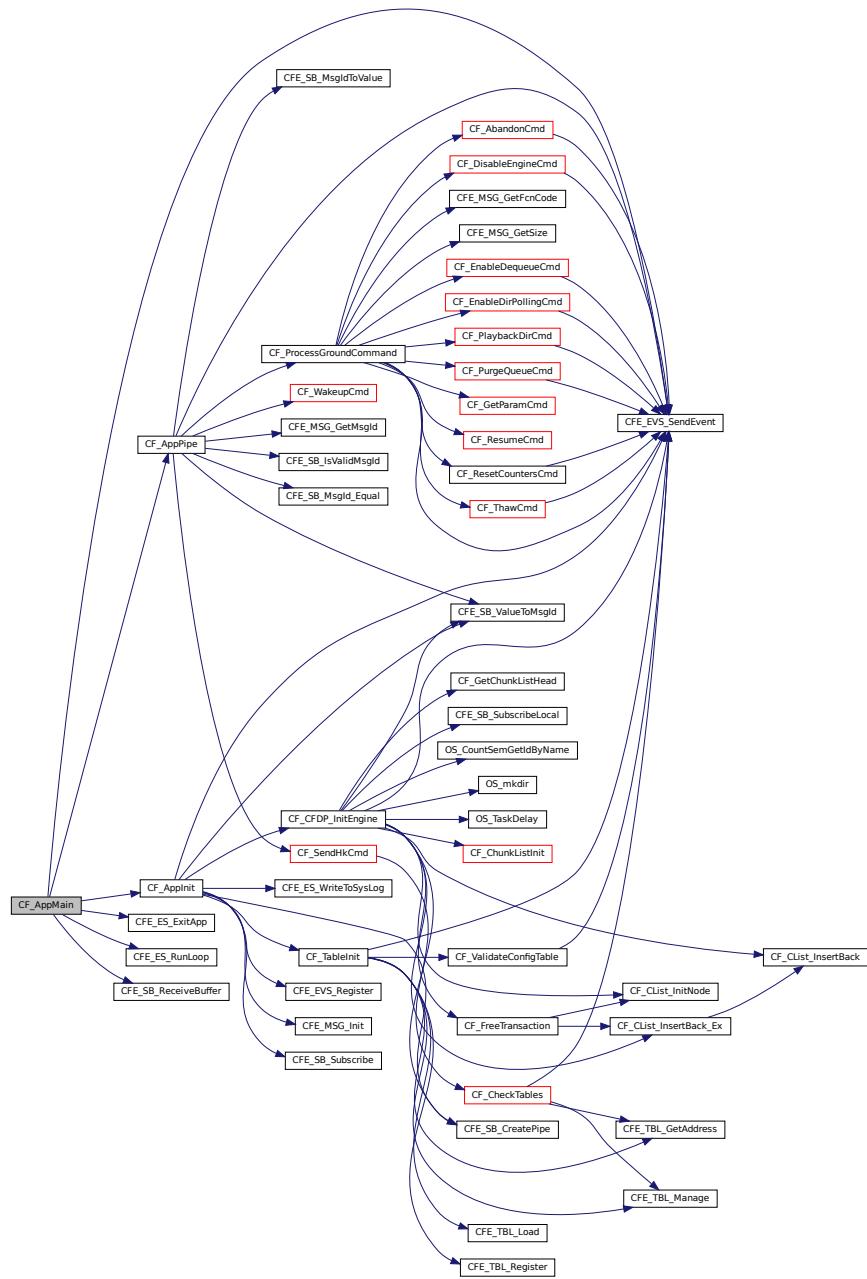
### **Assumptions, External Events, and Notes:**

This must only be called once.

Definition at line 256 of file cf\_app.c.

References CF\_AppData, CF\_AppInit(), CF\_AppPipe(), CF\_INIT\_MSG\_RECV\_ERR\_EID, CF\_PERF\_ID\_APPMAIN, CF\_RCVMSG\_TIMEOUT, CFE\_ES\_ExitApp(), CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_ES\_RunLoop(), CFE\_ES\_RunStatus\_APP\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SB\_NO\_ MESSAGE, CFE\_SB\_ReceiveBuffer(), CFE\_SB\_TIME\_OUT, CFE\_SUCCESS, CF\_AppData\_t::CmdPipe, and CF\_AppData t::RunStatus.

Here is the call graph for this function:



```
12.32.3.3 CF_CheckTables() void CF_CheckTables ( void )
```

Checks to see if a table update is pending, and perform it.

**Description**

Updates the table if the engine is disabled.

**Assumptions, External Events, and Notes:**

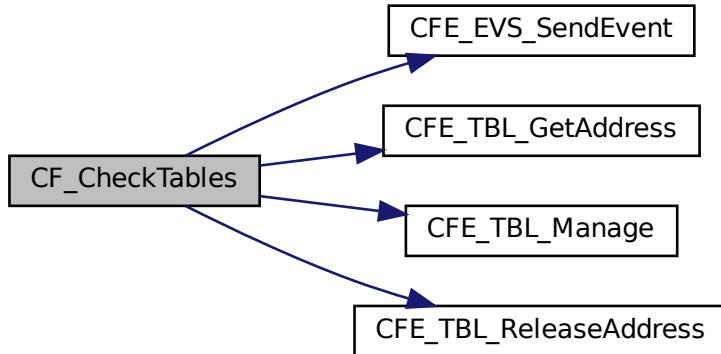
None

Definition at line 48 of file cf\_app.c.

References CF\_AppData, CF\_INIT\_TBL\_CHECK\_GA\_ERR\_EID, CF\_INIT\_TBL\_CHECK\_MAN\_ERR\_EID, CF\_INIT\_TBL\_CHECK\_REL\_ERR\_EID, CFE\_ES\_RunStatus\_APP\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_GetAddress(), CFE\_TBL\_Manage(), CFE\_TBL\_ReleaseAddress(), CF\_AppData\_t::config\_handle, CF\_AppData\_t::config\_table, CF\_Engine::enabled, CF\_AppData\_t::engine, and CF\_AppData\_t::RunStatus.

Referenced by CF\_SendHkCmd().

Here is the call graph for this function:



#### 12.32.3.4 CF\_TableInit() `CFE_Status_t CF_TableInit( void )`

Load the table on application start.

**Assumptions, External Events, and Notes:**

None

**Return values**

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<i>Returns</i>	anything else on error.

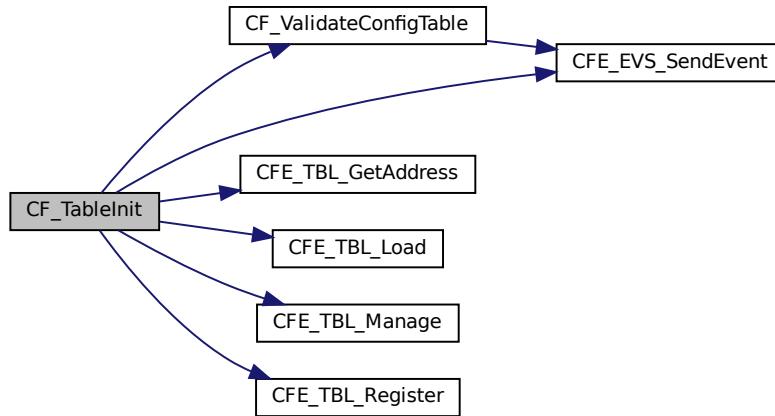
Definition at line 134 of file cf\_app.c.

References CF\_AppData, CF\_CONFIG\_TABLE\_FILENAME, CF\_CONFIG\_TABLE\_NAME, CF\_INIT\_TBL\_

GETADDR\_ERR\_EID, CF\_INIT\_TBL\_LOAD\_ERR\_EID, CF\_INIT\_TBL\_MANAGE\_ERR\_EID, CF\_INIT\_TBL\_REG\_ERR\_EID, CF\_ValidateConfigTable(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CFE\_TBL\_GetAddress(), CFE\_TBL\_INFO\_UPDATED, CFE\_TBL\_Load(), CFE\_TBL\_Manage(), CFE\_TBL\_OPT\_LOAD\_DUMP, CFE\_TBL\_OPT\_SNGL\_BUFFER, CFE\_TBL\_Register(), CFE\_TBL\_SRC\_FILE, CF\_AppData\_t::config\_handle, and CF\_AppData\_t::config\_table.

Referenced by CF\_AppInit().

Here is the call graph for this function:



**12.32.3.5 CF\_ValidateConfigTable()** `CFE_Status_t CF_ValidateConfigTable ( void * tbl_ptr )`

Validation function for config table.

#### Description

Checks that the config table being loaded has correct data.

#### Assumptions, External Events, and Notes:

None

#### Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CFE_STATUS_VALIDATION_FAILURE</code>	if the config table fails one of the validation checks

Definition at line 101 of file cf\_app.c.

References `CF_INIT_CRC_ALIGN_ERR_EID`, `CF_INIT_OUTGOING_SIZE_ERR_EID`, `CF_INIT_TPS_ERR_EID`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_STATUS_VALIDATION_FAILURE`, `CFE_SUCCESS`, `CF_ConfigTable::outgoing_file_chunk_size`, `CF_ConfigTable::rx_crc_calc_bytes_per_wakeup`, and `CF_ConfigTable::ticks_per_second`.

Referenced by `CF_TableInit()`.

Here is the call graph for this function:



#### 12.32.4 Variable Documentation

##### 12.32.4.1 CF\_AppData CF\_AppData\_t CF\_AppData [extern]

Singleton instance of the application global data.

Definition at line 40 of file cf\_app.c.

Referenced by CF\_AbandonCmd(), CF\_AppInit(), CF\_AppMain(), CF\_AppPipe(), CF\_CancelCmd(), CF\_CFDP\_AppendTlv(), CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_CycleEngine(), CF\_CFDP\_DisableEngine(), CF\_CFDP\_GetTempName(), CF\_CFDP\_InitEngine(), CF\_CFDP\_MsgOutGet(), CF\_CFDP\_PlaybackDir(), CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_ReceiveMessage(), CF\_CFDP\_ReceivePdu(), CF\_CFDP\_RecvDrop(), CF\_CFDP\_RecvFd(), CF\_CFDP\_RecvHold(), CF\_CFDP\_RecvMd(), CF\_CFDP\_RecvPh(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_Send(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), CF\_CFDP\_StartRxTransaction(), CF\_CFDP\_TxFile(), CF\_CFDP\_TxFile\_Initiate(), CF\_CheckTables(), CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_DisableEngineCmd(), CF\_DoEnableDisableDequeue(), CF\_DoEnableDisablePolldir(), CF\_DoFreezeThaw(), CF\_DoPurgeQueue(), CF\_DoSuspRes(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_EnableEngineCmd(), CF\_FindTransactionBySequenceNumberAllChannels(), CF\_FreeTransaction(), CF\_FreezeCmd(), CF\_GetChannelFromTxn(), CF\_SetParamCmd(), CF\_InsertSortPrio(), CF\_MoveTransaction(), CF\_NoopCmd(), CF\_PlaybackDirCmd(), CF\_ProcessGroundCommand(), CF\_PurgeQueueCmd(), CF\_ResetCountersCmd(), CF\_SendHkCmd(), CF\_TableInit(), CF\_ThawCmd(), CF\_Timer\_Sec2Ticks(), CF\_TraverseAllTransactions\_All\_Channels(), CF\_TsnChanAction(), CF\_TxFileCmd(), CF\_ValidateMaxOutgoingCmd(), and CF\_WriteQueueCmd().

#### 12.33 apps/cf/fsw/src(cf\_assert.h File Reference

```
#include "cfe.h"
```

#### Macros

##### CF assert macro

*CF Assert statements within the code are primarily informational for developers, as the conditions within them should always be true. Barring any unforeseen bugs in the code, they should never get triggered. However, if the code is modified, these conditions could happen, so it is still worthwhile to keep these statements in the source code, so they can be enabled if necessary.*

The debug build assert translates CF\_ASSERT to the system assert. Note that asserts may still get disabled if building with NDEBUG flag set, even if CF\_DEBUG\_BUILD flag is enabled.

It should be impossible to get any conditions which are asserted, so it should be safe to turn these off via the normal build assert. This is the configuration that the code should be normally tested and verified in.

- #define CF\_ASSERT(x) /\* no-op \*/  
    Normal build assert.
- #define CF\_TRACE(...)

### 12.33.1 Detailed Description

The CF Application CF\_ASSERT macro

### 12.33.2 Macro Definition Documentation

#### 12.33.2.1 CF\_ASSERT #define CF\_ASSERT (

    x ) /\* no-op \*/

Normal build assert.

Definition at line 68 of file cf\_assert.h.

#### 12.33.2.2 CF\_TRACE #define CF\_TRACE (

    ... )

Definition at line 69 of file cf\_assert.h.

## 12.34 apps/cf/fsw/src/cf\_cfdp.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_r.h"
#include "cf_cfdp_s.h"
#include "cf_cfdp_dispatch.h"
#include "cf_cfdp_sbintf.h"
#include <string.h>
#include "cf_assert.h"
```

### Functions

- void CF\_CFDP\_EncodeStart (CF\_EncoderState\_t \*penc, void \*msgbuf, CF\_Logical\_PduBuffer\_t \*ph, size\_t encaps\_hdr\_size, size\_t total\_size)  
*Initiate the process of encoding a new PDU to send.*
- void CF\_CFDP\_DecodeStart (CF\_DecoderState\_t \*pdec, const void \*msgbuf, CF\_Logical\_PduBuffer\_t \*ph, size\_t encaps\_hdr\_size, size\_t total\_size)  
*Initiate the process of decoding a received PDU.*
- void CF\_CFDP\_ArmAckTimer (CF\_Transaction\_t \*txn)

*Arm the ACK timer.*

- static `CF_CFDP_Class_t CF_CFDP_GetClass` (const `CF_Transaction_t *txn`)
- static bool `CF_CFDP_IsSender` (`CF_Transaction_t *txn`)
- void `CF_CFDP_ArmInactTimer` (`CF_Transaction_t *txn`)

*Arms the inactivity timer.*

- bool `CF_CFDP_CheckAckNakCount` (`CF_Transaction_t *txn, uint8 *counter`)

*Increment ack/nak counter and check limit.*

- void `CF_CFDP_DispatchRecv` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Dispatch received packet to its handler.*

- static `CF_ChunkWrapper_t *CF_CFDP_FindUnusedChunks` (`CF_Channel_t *chan, CF_Direction_t dir`)
- static void `CF_CFDP_SetPduLength` (`CF_Logical_PduBuffer_t *ph`)
- `CF_Logical_PduBuffer_t *CF_CFDP_ConstructPduHeader` (const `CF_Transaction_t *txn, CF_CFDP_FileDirective_t directive_code, CF_EntityId_t src_eid, CF_EntityId_t dst_eid, bool towards_sender, CF_TransactionSeq_t tsn, bool silent`)

*Build the PDU header in the output buffer to prepare to send a packet.*

- `CFE_Status_t CF_CFDP_SendMd` (`CF_Transaction_t *txn`)

*Build a metadata PDU for transmit.*

- `CFE_Status_t CF_CFDP_SendFd` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Send a previously-assembled filedata PDU for transmit.*

- void `CF_CFDP_AppendTlv` (`CF_Logical_TlvList_t *ptlv_list, CF_CFDP_TlvType_t tlv_type`)

*Appends a single TLV value to the logical PDU data.*

- `CFE_Status_t CF_CFDP_SendEof` (`CF_Transaction_t *txn`)

*Build an EOF PDU for transmit.*

- `CFE_Status_t CF_CFDP_SendAck` (`CF_Transaction_t *txn, CF_CFDP_FileDirective_t dir_code`)

*Build an ACK PDU for transmit.*

- `CFE_Status_t CF_CFDP_SendFin` (`CF_Transaction_t *txn`)

*Build a FIN PDU for transmit.*

- void `CF_CFDP_SendNak` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Send a previously-assembled NAK PDU for transmit.*

- `CFE_Status_t CF_CFDP_RecvPh` (`uint8 chan_num, CF_Logical_PduBuffer_t *ph`)

*Unpack a basic PDU header from a received message.*

- `CFE_Status_t CF_CFDP_RecvMd` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Unpack a metadata PDU from a received message.*

- `CFE_Status_t CF_CFDP_RecvFd` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Unpack a file data PDU from a received message.*

- `CFE_Status_t CF_CFDP_RecvEof` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Unpack an EOF PDU from a received message.*

- `CFE_Status_t CF_CFDP_RecvAck` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Unpack an ACK PDU from a received message.*

- `CFE_Status_t CF_CFDP_RecvFin` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Unpack an FIN PDU from a received message.*

- `CFE_Status_t CF_CFDP_RecvNak` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Unpack a NAK PDU from a received message.*

- void `CF_CFDP_RecvDrop` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Receive state function to ignore a packet.*

- void `CF_CFDP_RecvHold` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

*Receive state function during holdover period.*

- void `CF_CFDP_RecvInit` (`CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph`)

- Receive state function to process new rx transaction.*
- void `CF_CFDP_AllocChunkList` (`CF_Transaction_t` \*`txn`)  
*Allocates a chunk list for the transaction.*
  - void `CF_CFDP_SetupTxTransaction` (`CF_Transaction_t` \*`txn`)  
*Sets up a new TX transaction.*
  - void `CF_CFDP_SetupRxTransaction` (`CF_Transaction_t` \*`txn`, `CF_Logical_PduBuffer_t` \*`ph`)  
*Sets up a new RX transaction based on PDU.*
  - void `CF_CFDP_ReceivePdu` (`CF_Channel_t` \*`chan`, `CF_Logical_PduBuffer_t` \*`ph`)  
*Receive PDU processing entry point.*
  - `CFE_Status_t CF_CFDP_InitEngine` (void)  
*Initialization function for the CFDP engine.*
  - void `CF_CFDP_S_Tick_NewData` (`CF_Transaction_t` \*`txn`)  
*Tick processor to send new file data.*
  - bool `CF_CFDP_StartFirstPending` (`CF_Channel_t` \*`chan`)  
*Pulls next TX transaction from the PEND queue.*
  - `CF_CListTraverse_Status_t CF_CFDP_DoTick` (`CF_CListNode_t` \*`node`, void \*`context`)  
*List traversal function that calls a r or s tick function.*
  - void `CF_CFDP_CompleteTick` (`CF_Transaction_t` \*`txn`)  
*Complete tick processing on a transaction.*
  - void `CF_CFDP_TickTransactions` (`CF_Channel_t` \*`chan`)  
*Call R and then S tick functions for all active transactions.*
  - void `CF_CFDP_InitTxnTxFile` (`CF_Transaction_t` \*`txn`, `CF_CFDP_Class_t` `cfdp_class`, `uint8` `keep`, `uint8` `chan`, `uint8` `priority`)  
*Helper function to set tx file state in a transaction.*
  - static void `CF_CFDP_TxFile_Initiate` (`CF_Transaction_t` \*`txn`, `CF_CFDP_Class_t` `cfdp_class`, `uint8` `keep`, `uint8` `chan`, `uint8` `priority`, `CF_EntityId_t` `dest_id`)
  - `CFE_Status_t CF_CFDP_TxFile` (const char \*`src_filename`, const char \*`dst_filename`, `CF_CFDP_Class_t` `cfdp_class`, `uint8` `keep`, `uint8` `chan_num`, `uint8` `priority`, `CF_EntityId_t` `dest_id`)  
*Begin transmit of a file.*
  - `CF_Transaction_t` \* `CF_CFDP_StartRxTransaction` (`uint8` `chan_num`)  
*Helper function to start a new RX transaction.*
  - static `CFE_Status_t CF_CFDP_PlaybackDir_Initiate` (`CF_Playback_t` \*`pb`, const char \*`src_filename`, const char \*`dst_filename`, `CF_CFDP_Class_t` `cfdp_class`, `uint8` `keep`, `uint8` `chan`, `uint8` `priority`, `CF_EntityId_t` `dest_id`)
  - `CFE_Status_t CF_CFDP_PlaybackDir` (const char \*`src_filename`, const char \*`dst_filename`, `CF_CFDP_Class_t` `cfdp_class`, `uint8` `keep`, `uint8` `chan`, `uint8` `priority`, `uint16` `dest_id`)  
*Begin transmit of a directory.*
  - void `CF_CFDP_ProcessPlaybackDirectory` (`CF_Channel_t` \*`chan`, `CF_Playback_t` \*`pb`)  
*Step each active playback directory.*
  - static void `CF_CFDP_UpdatePollPbCounted` (`CF_Playback_t` \*`pb`, int `up`, `uint8` \*`counter`)
  - static void `CF_CFDP_ProcessPlaybackDirectories` (`CF_Channel_t` \*`chan`)
  - void `CF_CFDP_ProcessPollingDirectories` (`CF_Channel_t` \*`chan`)  
*Kick the dir playback if timer elapsed.*
  - void `CF_CFDP_CycleEngine` (void)  
*Cycle the engine. Called once per wakeup.*
  - void `CF_CFDP_FinishTransaction` (`CF_Transaction_t` \*`txn`, bool `keep_history`)  
*Finish a transaction.*
  - void `CF_CFDP_RecycleTransaction` (`CF_Transaction_t` \*`txn`)  
*Recover resources associated with a transaction.*

- void `CF_CFDP_SetTxnStatus (CF_Transaction_t *txn, CF_TxnStatus_t txn_stat)`  
*Helper function to store transaction status code only.*
- `CF_TxnStatus_t CF_CFDP_GetTxnStatus (const CF_Transaction_t *txn)`  
*Helper function to retrieve transaction status code only.*
- `CF_TxnStatus_t CF_CFDP_TxnIsOK (const CF_Transaction_t *txn)`  
*Helper function to check if a transaction is errored.*
- void `CF_CFDP_SendEotPkt (CF_Transaction_t *txn)`  
*Send an end of transaction packet.*
- int `CF_CFDP_CopyStringFromLV (char *buf, size_t buf_maxsz, const CF_Logical_Lv_t *src_lv)`  
*Copy string data from a lv (length, value) pair.*
- void `CF_CFDP_CancelTransaction (CF_Transaction_t *txn)`  
*Cancels a transaction.*
- `CF_CListTraverse_Status_t CF_CFDP_CloseFiles (CF_CListNode_t *node, void *context)`  
*List traversal function to close all files in all active transactions.*
- void `CF_CFDP_DisableEngine (void)`  
*Disables the CFDP engine and resets all state in it.*
- void `CF_CFDP_GetTempName (const CF_History_t *hist, char *FileNameBuf, size_t FileNameSize)`  
*Get temporary file name.*
- const char \* `CF_CFDP_GetMoveTarget (const char *dest_dir, const char *subject_file, char *dest_buf, size_t dest_size)`  
*Get move target.*

### 12.34.1 Detailed Description

The CF Application main CFDP engine and PDU parsing implementation

This file contains two sets of functions. The first is what is needed to deal with CFDP PDUs. Specifically validating them for correctness and ensuring the byte-order is correct for the target. The second is incoming and outgoing CFDP PDUs pass through here. All receive CFDP PDU logic is performed here and the data is passed to the R (rx) and S (tx) logic.

### 12.34.2 Function Documentation

#### 12.34.2.1 `CF_CFDP_AllocChunkList()` void CF\_CFDP\_AllocChunkList ( `CF_Transaction_t * txn` )

Allocates a chunk list for the transaction.

All transactions use a chunklist to track gaps in the file as it is sent or received. This is used for NAK generation and processing in class 2 or just simply checking if the whole file is received in class 1.

The chunk lists come from a pool and are re-used

Only active transactions need a chunklist. Pending TX transactions will not get one assigned until they become active.

#### Parameters

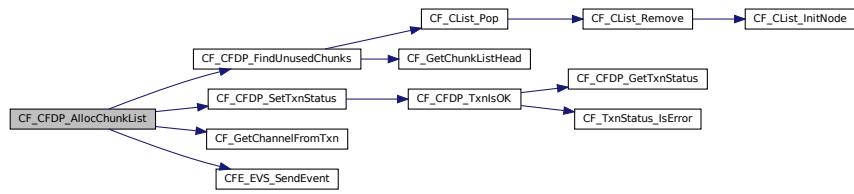
<code>txn</code>	Transaction pointer
------------------	---------------------

Definition at line 978 of file cf\_cfdp.c.

References `CF_CFDP_FindUnusedChunks()`, `CF_CFDP_NO_CHUNKLIST_AVAIL_EID`, `CF_CFDP_SetTxnStatus()`, `CF_GetChannelFromTxn()`, `CF_TxnStatus_NO_RESOURCE`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CF_Transaction::chunks`, `CF_History::dir`, `CF_Transaction::history`, and `CF_History::seq_num`.

Referenced by `CF_CFDP_SetupRxTransaction()`, and `CF_CFDP_SetupTxTransaction()`.

Here is the call graph for this function:



**12.34.2.2 CF\_CFDP\_AppendTlv()** `void CF_CFDP_AppendTlv ( CF_Logical_TlvList_t * ptlv_list, CF_CFDP_TlvType_t tlv_type )`

Appends a single TLV value to the logical PDU data.

This function implements common functionality between SendEof and SendFin which append a TLV value specifying the faulting entity ID.

**Assumptions, External Events, and Notes:**

`ptlv_list` must not be NULL. Only `CF_CFDP_TLV_TYPE_ENTITY_ID` type is currently implemented

#### Parameters

<code>ptlv_list</code>	TLV list from current PDU buffer.
<code>tlv_type</code>	Type of TLV to append. Currently must be <code>CF_CFDP_TLV_TYPE_ENTITY_ID</code> .

Definition at line 433 of file cf\_cfdp.c.

References `CF_AppData`, `CF_CFDP_GetValueEncodedSize()`, `CF_CFDP_TLV_TYPE_ENTITY_ID`, `CF_PDU_MAX_TLV`, `CF_AppData_t::config_table`, `CF_Logical_Tlv::data`, `CF_Logical_TlvData::data_ptr`, `CF_Logical_TlvData::eid`, `CF_Logical_Tlv::length`, `CF_ConfigTable::local_eid`, `CF_Logical_TlvList::num_tlv`, `CF_Logical_TlvList::tlv`, and `CF_Logical_Tlv::type`.

Referenced by `CF_CFDP_SendEof()`, and `CF_CFDP_SendFin()`.

Here is the call graph for this function:



**12.34.2.3 CF\_CFDP\_ArmAckTimer()** `void CF_CFDP_ArmAckTimer ( CF_Transaction_t * txn )`

Arm the ACK timer.

## Description

Helper function to arm the ACK timer and set the flag.

### Assumptions, External Events, and Notes:

*txn* must not be NULL.

## Parameters

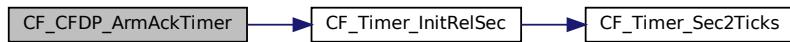
<i>txn</i>	Pointer to the transaction state
------------	----------------------------------

Definition at line 123 of file cf\_cfdp.c.

References CF\_Transaction::ack\_timer, CF\_Flags\_Common::ack\_timer\_armed, CF\_ChannelConfig::ack\_timer\_s, CF\_AppData, CF\_Timer\_InitRelSec(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_StateFlags::com, CF\_AppData\_t::config\_table, and CF\_Transaction::flags.

Referenced by CF\_CFDP\_R2\_Recv(), CF\_CFDP\_R\_Tick\_Maintenance(), CF\_CFDP\_S\_Tick\_Maintenance(), and CF\_CFDP\_SendNak().

Here is the call graph for this function:



### 12.34.2.4 CF\_CFDP\_ArmInactTimer()

```
void CF_CFDP_ArmInactTimer (
    CF_Transaction_t * txn )
```

Arms the inactivity timer.

This is invoked whenever activity occurs on a transaction. The inactivity timer will be reset to the value in the configuration.

## Parameters

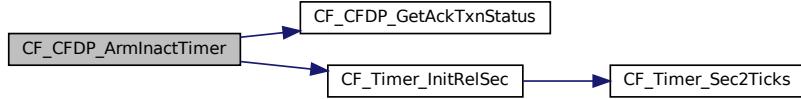
<i>txn</i>	Transaction pointer
------------	---------------------

Definition at line 157 of file cf\_cfdp.c.

References CF\_ChannelConfig::ack\_timer\_s, CF\_AppData, CF\_CFDP\_AckTxnStatus\_ACTIVE, CF\_CFDP\_GetAckTxnStatus(), CF\_Timer\_InitRelSec(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_AppData\_t::config\_table, CF\_Transaction::inactivity\_timer, and CF\_ChannelConfig::inactivity\_timer\_s.

Referenced by CF\_CFDP\_DispatchRecv(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_SetupRxTransaction(), and CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



**12.34.2.5 CF\_CFDP\_CancelTransaction()** void CF\_CFDP\_CancelTransaction ( CF\_Transaction\_t \* txn )

Cancels a transaction.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

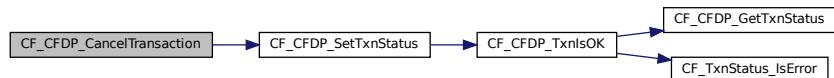
txn	Pointer to the transaction state
-----	----------------------------------

Definition at line 2116 of file cf\_cfdp.c.

References CF\_Flags\_Common::canceled, CF\_CFDP\_SetTxnStatus(), CF\_TxnStatus\_CANCEL\_REQUEST\_← RECEIVED, CF\_StateFlags::com, and CF\_Transaction::flags.

Referenced by CF\_Cancel\_TxnCmd().

Here is the call graph for this function:



**12.34.2.6 CF\_CFDP\_CheckAckNakCount()** bool CF\_CFDP\_CheckAckNakCount ( CF\_Transaction\_t \* txn, uint8 \* counter )

Increment ack/nak counter and check limit.

**Description**

Checks the counter against the configured limit. If still under limit, increment counter and return true. If reached limit, generate event and return false.

**Assumptions, External Events, and Notes:**

**Parameters**

<i>txn</i>	the transaction object
<i>counter</i>	pointer to the respective ack/nak counter

**Return values**

<i>true</i>	Within limit, another ACK/NAK is allowed
<i>false</i>	Reached limit, another ACK/NAK is not allowed

Definition at line 188 of file cf\_cfdp.c.

References CF\_HkFault::ack\_limit, CF\_ChannelConfig::ack\_limit, CF\_AppData, CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_<→ EID, CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID, CF\_Direction\_TX, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_AppData\_t::config\_table, CF\_HkChannel\_Data::counters, CF\_History::dir, CF\_HkCounters::fault, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_History::peer\_eid, and CF\_History::seq\_num.

Referenced by CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_<→ SubstateRecvEof(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), and CF\_CFDP\_S\_SubstateRecvFin().

Here is the call graph for this function:



**12.34.2.7 CF\_CFDP\_CloseFiles()** [CF\\_CListTraverse\\_Status\\_t](#) CF\_CFDP\_CloseFiles (   
*CF\_CListNode\_t* \* *node*,   
*void* \* *context* )

List traversal function to close all files in all active transactions.

This helper is used in conjunction with [CF\\_CList\\_Traverse\(\)](#).

**Assumptions, External Events, and Notes:**

*node* must not be NULL.

**Parameters**

<i>node</i>	List node pointer
<i>context</i>	Opaque pointer, not used in this function

**Returns**

integer traversal code

### Return values

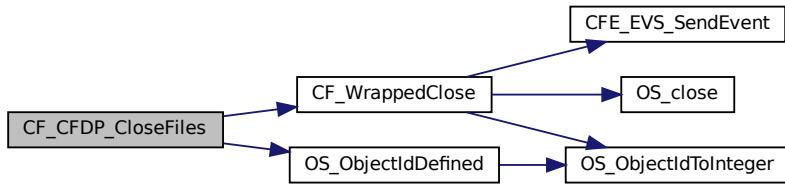
Always	CF_LIST_CONT indicate list traversal should not exit early.
--------	---

Definition at line 2132 of file cf\_cfdp.c.

References CF\_CLIST\_CONT, CF\_WrappedClose(), container\_of, CF\_Transaction::fd, and OS\_ObjectIdDefined().

Referenced by CF\_CFDP\_DisableEngine().

Here is the call graph for this function:



### 12.34.2.8 CF\_CFDP\_CompleteTick()

```
void CF_CFDP_CompleteTick (
```

```
    CF_Transaction_t * txn )
```

Complete tick processing on a transaction.

#### Description

Checks if transmit operations have been blocked due to TX limits If so, snapshot this TXN as a resume point for next cycle. If not, does nothing.

#### Assumptions, External Events, and Notes:

#### Parameters

<code>txn</code>	the transaction object
------------------	------------------------

Definition at line 1357 of file cf\_cfdp.c.

References CF\_GetChannelFromTxn(), CF\_Channel::tick\_resume, and CF\_Channel::tx\_blocked.

Referenced by CF\_CFDP\_R\_Tick(), and CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



```

12.34.2.9 CF_CFDP_ConstructPduHeader() CF_Logical_PduBuffer_t* CF_CFDP_ConstructPduHeader (
    const CF_Transaction_t * txn,
    CF_CFDP_FileDirective_t directive_code,
    CF_EntityId_t src_eid,
    CF_EntityId_t dst_eid,
    bool towards_sender,
    CF_TransactionSeq_t tsn,
    bool silent )

```

Build the PDU header in the output buffer to prepare to send a packet.

#### Assumptions, External Events, and Notes:

*txn* must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction object
<i>directive_code</i>	Code to use for file directive headers (set to 0 for data)
<i>src_eid</i>	Value to set in source entity ID field
<i>dst_eid</i>	Value to set in destination entity ID field
<i>towards_sender</i>	Whether this is transmitting toward the sender entity
<i>tsn</i>	Transaction sequence number to put into PDU
<i>silent</i>	If true, suppress error event if no message buffer available

#### Returns

Pointer to PDU buffer which may be filled with additional data

#### Return values

NULL	if no message buffer available
------	--------------------------------

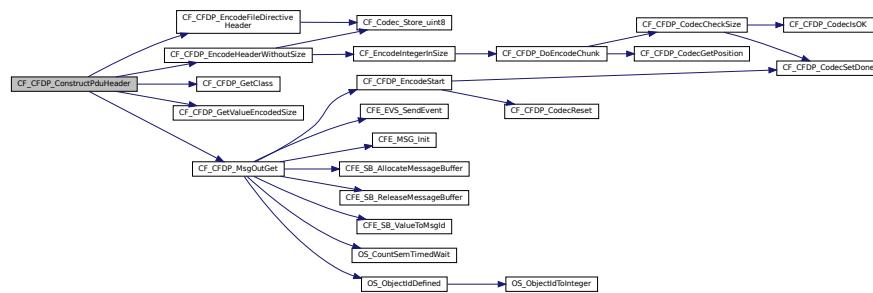
Definition at line 296 of file cf\_cfdp.c.

References CF\_CFDP\_CLASS\_1, CF\_CFDP\_EncodeFileDirectiveHeader(), CF\_CFDP\_EncodeHeaderWithoutSize(), CF\_CFDP\_GetClass(), CF\_CFDP\_GetValueEncodedSize(), CF\_CFDP\_MsgOutGet(), CF\_Logical\_PduHeader::destination\_eid, CF\_Logical\_PduHeader::direction, CF\_Logical\_PduFileDirectiveHeader::directive\_code, CF\_Logical\_PduHeader::eid\_length, CF\_Logical\_PduBuffer::fdirective, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::pdu\_type, CF\_Logical\_PduBuffer::penc, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical

\_PduHeader::source\_eid, CF\_Logical\_PduHeader::txm\_mode, CF\_Logical\_PduHeader::txn\_seq\_length, and CF\_Logical\_PduHeader::version.

Referenced by CF\_CFDP\_R\_SendNak(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFin(), and CF\_CFDP\_SendMd().

Here is the call graph for this function:



### 12.34.2.10 CF\_CFDP\_CopyStringFromLV()

```
int CF_CFDP_CopyStringFromLV (
    char * buf,
    size_t buf_maxsz,
    const CF_Logical_Lv_t * src_lv )
```

Copy string data from a lv (length, value) pair.

This copies a string value from an LV pair inside a PDU buffer. In CF this is used for file names embedded within PDUs.

#### Note

This function assures that the output string is terminated appropriately, such that it can be used as a normal C string. As such, the buffer size must be at least 1 byte larger than the maximum string length.

#### Assumptions, External Events, and Notes:

src\_lv must not be NULL. buf must not be NULL.

#### Parameters

<i>buf</i>	Pointer to buffer to store string
<i>buf_maxsz</i>	Total size of buffer pointer to by buf (usable size is 1 byte less, for termination)
<i>src_lv</i>	Pointer to LV pair from logical PDU buffer

#### Returns

The resulting string length, NOT including termination character

#### Return values

<i>CF_ERROR</i>	on error
-----------------	----------

Definition at line 2096 of file cf\_cfdp.c.

References CF\_ERROR, CF\_Logical\_Lv::data\_ptr, and CF\_Logical\_Lv::length.  
Referenced by CF\_CFDP\_RecvMd().

#### 12.34.2.11 CF\_CFDP\_CycleEngine()

```
void CF_CFDP_CycleEngine (
    void )
```

Cycle the engine. Called once per wakeup.

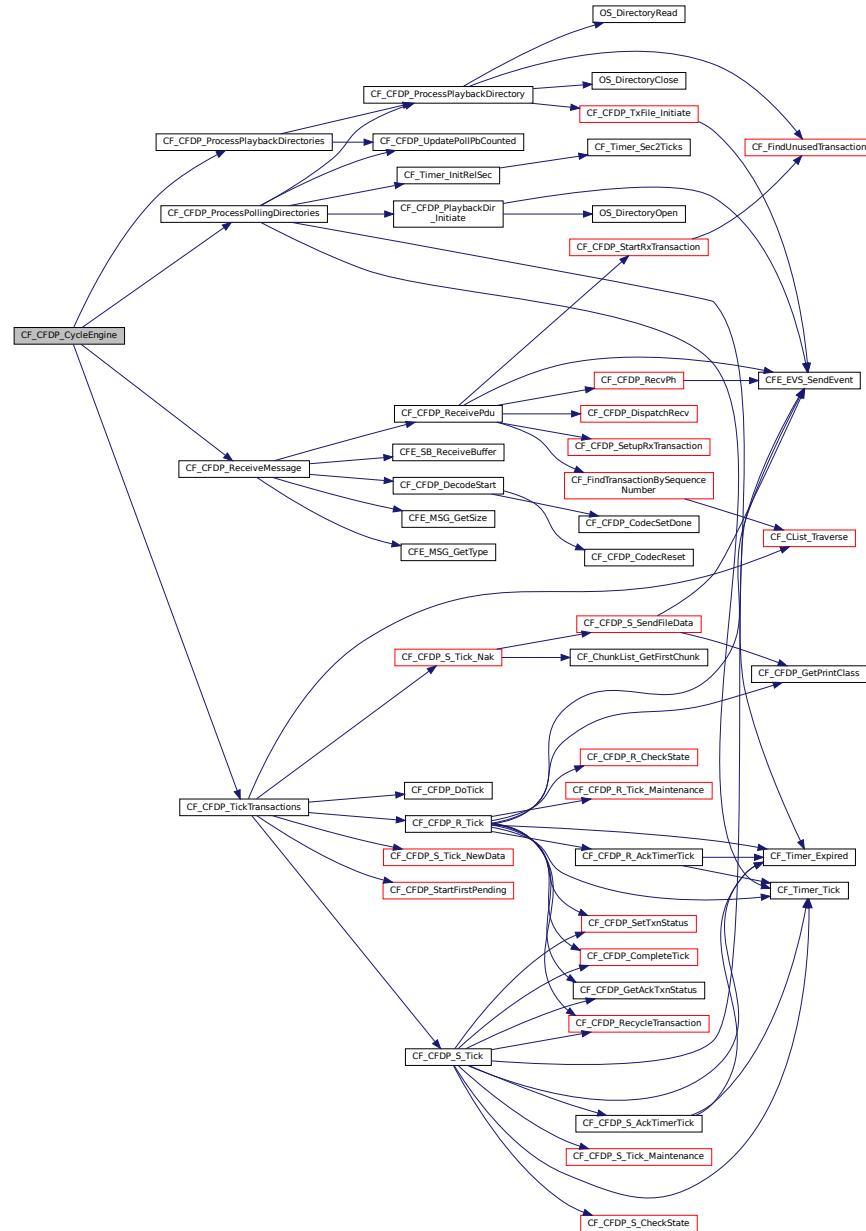
**Assumptions, External Events, and Notes:**

None

Definition at line 1862 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_ReceiveMessage(), CF\_CFDP\_TickTransactions(), CF\_NUM\_CHANNELS, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_HkChannel\_Data::frozen, CF\_AppData\_t::hk, CF\_Channel::outgoing\_counter, CF\_HkPacket::Payload, and CF\_Channel::tx\_blocked.  
Referenced by CF\_WakeupCmd().

Here is the call graph for this function:



**12.34.2.12 CF\_CFDP\_DecodeStart()** void CF\_CFDP\_DecodeStart (

```
CF_DecoderState_t * pdec,  
const void * msgbuf,  
CF_Logical_PduBuffer_t * ph,  
size_t encaps_hdr_size,  
size_t total_size )
```

Initiate the process of decoding a received PDU.

This resets the decoder and PDU buffer to initial values, and prepares for decoding a new PDU that was received from a remote entity.

#### Parameters

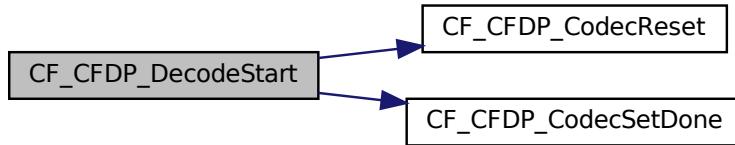
<i>pdec</i>	Decoder state structure, will be reset-initialized by this call to point to msgbuf.
<i>msgbuf</i>	Pointer to encapsulation message, in this case a CFE software bus message
<i>ph</i>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<i>encap_hdr_size</i>	Offset of first CFDP PDU octet within buffer
<i>total_size</i>	Total size of msgbuf encapsulation structure (decoding cannot exceed this)

Definition at line 89 of file cf\_cfdp.c.

References CF\_DecoderState::base, CF\_CFDP\_CodecReset(), CF\_CFDP\_CodecSetDone(), CF\_DecoderState::codec\_state, CF\_CodecState::max\_size, and CF\_Logical\_PduBuffer::pdec.

Referenced by CF\_CFDP\_ReceiveMessage().

Here is the call graph for this function:



**12.34.2.13 CF\_CFDP\_DisableEngine()** void CF\_CFDP\_DisableEngine ( void )

Disables the CFDP engine and resets all state in it.

#### Assumptions, External Events, and Notes:

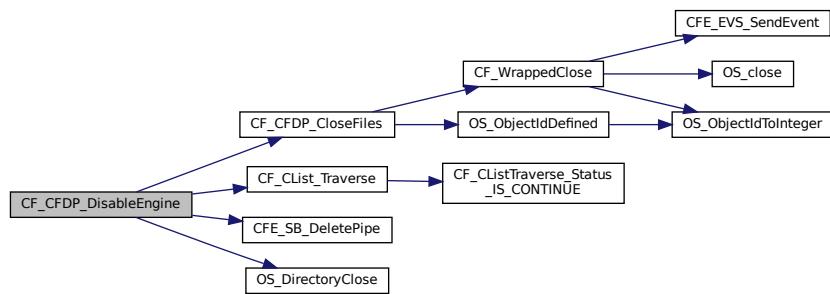
None

Definition at line 2148 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_CloseFiles(), CF\_CList\_Traverse(), CF\_MAX\_COMMANDED←\_PLAYBACK\_DIRECTORIES\_PER\_CHAN, CF\_MAX\_POLLING\_DIR\_PER\_CHAN, CF\_NUM\_CHANNELS, CF←\_Queueldx\_RX, CF\_Queueldx\_TX, CFE\_SB\_DeletePipe(), CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Playback::dir\_id, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_AppData\_t::hk, OS\_DirectoryClose(), CF\_Hk←Packet::Payload, CF\_Poll::pb, CF\_Channel::pipe, CF\_Channel::playback, CF\_Channel::poll, CF\_HkChannel\_Data::q←\_size, and CF\_Channel::qs.

Referenced by CF\_DisableEngineCmd().

Here is the call graph for this function:



**12.34.2.14 CF\_CFDP\_DispatchRecv()** void CF\_CFDP\_DispatchRecv ( CF\_Transaction\_t \* *txn*, CF\_Logical\_PduBuffer\_t \* *ph* )

Dispatch received packet to its handler.

This dispatches the PDU to the appropriate handler based on the transaction state

#### Assumptions, External Events, and Notes:

*txn* must not be null. It must be an initialized transaction.

#### Parameters

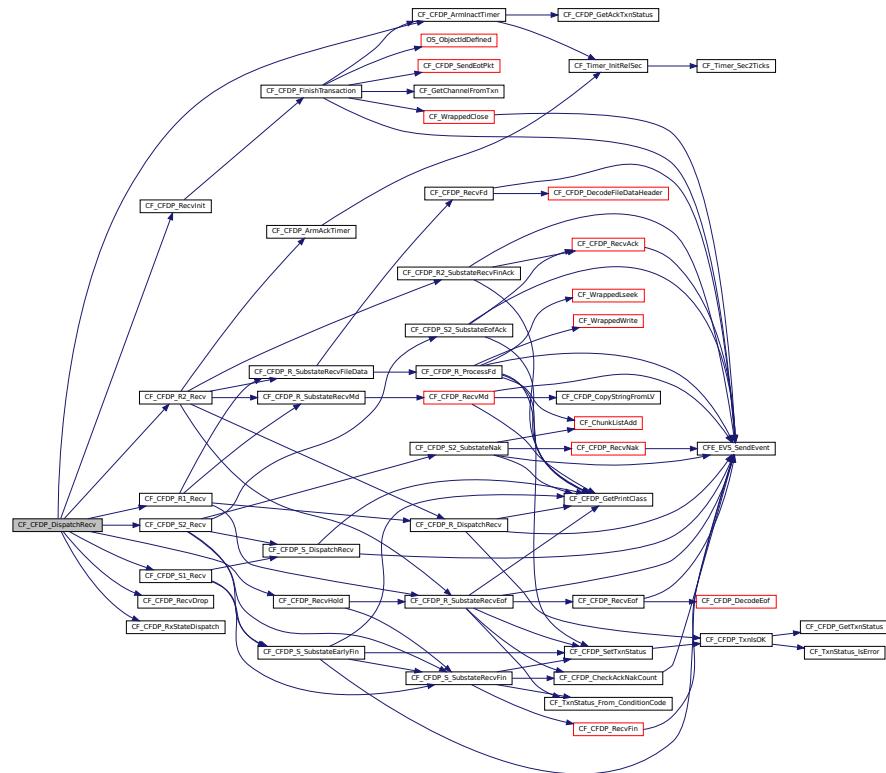
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 228 of file cf\_cfdp.c.

References CF\_CFDP\_ArmlnactTimer(), CF\_CFDP\_R1\_Recv(), CF\_CFDP\_R2\_Recv(), CF\_CFDP\_RecvDrop(), CF←\_CFDP\_RecvHold(), CF\_CFDP\_RecvInit(), CF\_CFDP\_RxStateDispatch(), CF\_CFDP\_S1\_Recv(), CF\_CFDP\_S2←Recv(), CF\_TxnState\_DROP, CF\_TxnState\_HOLD, CF\_TxnState\_INIT, CF\_TxnState\_R1, CF\_TxnState\_R2, CF\_Txn←State\_S1, CF\_TxnState\_S2, and CF\_CFDP\_TxnRecvDispatchTable\_t::rx.

Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



**12.34.2.15 CF\_CFDP\_DoTick()** CF\_CListTraverse\_Status\_t CF\_CFDP\_DoTick (

```
CF_CListNode_t * node,  
void * context )
```

List traversal function that calls a r or s tick function.

This helper is used in conjunction with [CF\\_CList\\_Traverse\(\)](#).

### **Assumptions, External Events, and Notes:**

node must not be NULL, context must not be NULL.

## Parameters

<i>node</i>	Pointer to list node
<i>context</i>	Pointer to CF_CFDP_Tick_args_t object (passed through)

## Returns

## integer traversal code

## Return values

***CF\_CLIST\_EXIT*** when it's found, which terminates list traversal

## Return values

<code>CF_CLIST_CONT</code>	when it's isn't found, which causes list traversal to continue
----------------------------	--

Definition at line 1324 of file cf\_cfdp.c.

References `CF_CLIST_CONT`, `CF_CLIST_EXIT`, `CF_CFDP_Tick_args::chan`, `CF_StateFlags::com`, `container_of`, `CF_Transaction::flags`, `CF_CFDP_Tick_args::fn`, `CF_CFDP_Tick_args::resume_point`, `CF_Flags_Common::suspended`, and `CF_Channel::tx_blocked`.

Referenced by `CF_CFDP_TickTransactions()`.

### 12.34.2.16 `CF_CFDP_EncodeStart()` `void CF_CFDP_EncodeStart (`

```
    CF_EncoderState_t * penc,
    void * msgbuf,
    CF_Logical_PduBuffer_t * ph,
    size_t encaps_hdr_size,
    size_t total_size )
```

Initiate the process of encoding a new PDU to send.

This resets the encoder and PDU buffer to initial values, and prepares for encoding a new PDU for sending to a remote entity.

#### Parameters

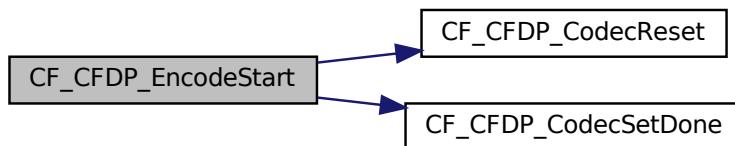
<code>penc</code>	Encoder state structure, will be reset-initialized by this call to point to <code>msgbuf</code> .
<code>msgbuf</code>	Pointer to encapsulation message, in this case a CFE software bus message
<code>ph</code>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<code>encap_hdr_size</code>	Offset of first CFDP PDU octet within buffer
<code>total_size</code>	Allocated size of <code>msgbuf</code> encapsulation structure (encoding cannot exceed this)

Definition at line 55 of file cf\_cfdp.c.

References `CF_EncoderState::base`, `CF_CFDP_CodecReset()`, `CF_CFDP_CodecSetDone()`, `CF_EncoderState::codec_state`, `CF_CodecState::max_size`, and `CF_Logical_PduBuffer::penc`.

Referenced by `CF_CFDP_MsgOutGet()`.

Here is the call graph for this function:



### 12.34.2.17 `CF_CFDP_FindUnusedChunks()` `static CF_ChunkWrapper_t* CF_CFDP_FindUnusedChunks (`

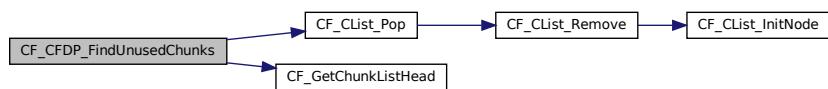
```
    CF_Channel_t * chan,
    CF_Direction_t dir ) [static]
```

Definition at line 247 of file cf\_cfdp.c.

References CF\_ASSERT, CF\_CList\_Pop(), CF\_GetChunkListHead(), and container\_of.

Referenced by CF\_CFDP\_AllocChunkList().

Here is the call graph for this function:



#### 12.34.2.18 CF\_CFDP\_FinishTransaction() void CF\_CFDP\_FinishTransaction (

```

    CF_Transaction_t * txn,
    bool keep_history )
  
```

Finish a transaction.

This marks the transaction as completed and puts it into a holdover state. After the inactivity timer expires, the resources will be recycled and become available for re-use.

Holdover is necessary because even though locally we consider the transaction to be complete, there may be undelivered PDUs still in network queues that get delivered to us late. By holding this transaction for a bit longer, we can still associate those PDUs with this transaction/seq\_num and appropriately handle them as dupes/spurious deliveries.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

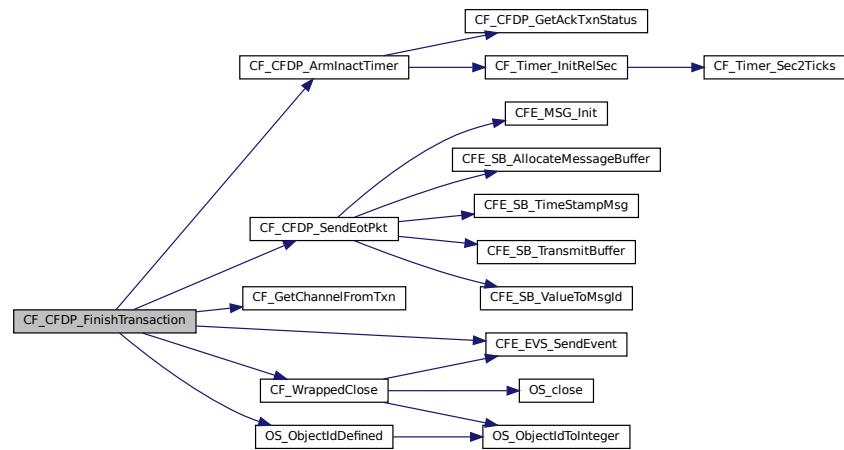
<i>txn</i>	Pointer to the transaction object
<i>keep_history</i>	Whether the transaction info should be preserved in history

Definition at line 1897 of file cf\_cfdp.c.

References CF\_ASSERT, CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_SendEotPkt(), CF\_Direction\_TX, CF\_GetChannelFromTxn(), CF\_QueueIdx\_FREE, CF\_RESET\_FREED\_XACT\_DBG\_EID, CF\_TRACE, CF\_TxnState\_HOLD, CF\_WrappedClose(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_SendEvent(), CF\_Flags\_Tx::cmd\_tx, CF\_StateFlags::com, CF\_History::dir, CF\_Transaction::fd, CF\_Transaction::flags, CF\_Transaction::history, CF\_Flags\_Common::keep\_history, CF\_Channel::num\_cmd\_tx, CF\_Playback::num\_ts, OS\_OBJECT\_ID\_UNDEFINED, OS\_ObjectIdDefined(), CF\_Transaction::pb, CF\_Flags\_Common::q\_index, CF\_History::seq\_num, CF\_Transaction::state, and CF\_StateFlags::tx.

Referenced by CF\_Abandon\_TxnCmd(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_RecvInit(), CF\_CFDP\_S\_CheckState(), and CF\_PurgeTransaction().

Here is the call graph for this function:



**12.34.2.19 CF\_CFDP\_GetClass()** static CF\_CFDP\_Class\_t CF\_CFDP\_GetClass ( const CF\_Transaction\_t \* txn ) [inline], [static]

Definition at line 134 of file cf\_cfdp.c.

References CF\_Assert, CF\_QueueIdx\_FREE, CF\_TxnState\_R2, CF\_TxnState\_S2, CF\_StateFlags::com, CF\_Transaction::flags, CF\_Flags\_Common::q\_index, and CF\_Transaction::state.

Referenced by CF\_CFDP\_ConstructPduHeader(), and CF\_CFDP\_SendNak().

**12.34.2.20 CF\_CFDP\_GetMoveTarget()** const char\* CF\_CFDP\_GetMoveTarget (

```

    const char * dest_dir,
    const char * subject_file,
    char * dest_buf,
    size_t dest_size
  )

```

Get move target.

Gets the full path of the destination file after move to dest\_dir

**Assumptions, External Events, and Notes:**

None

#### Parameters

<i>dest_dir</i>	Directory to move file into
<i>subject_file</i>	Full path of file to move
<i>dest_buf</i>	Buffer to store result
<i>dest_size</i>	Size of result buffer

#### Return values

<i>NULL</i>	if the result is not valid (i.e. dest_dir not set)
-------------	--

## Return values

<i>dest_buf</i>	if result is valid
-----------------	--------------------

Definition at line 2209 of file cf\_cfdp.c.

References CF\_EID\_INF\_CFDP\_BUF\_EXCEED, CF\_FILENAME\_TRUNCATED, CFE\_EVS\_EventType\_INFORMATION, and CFE\_EVS\_SendEvent().

Referenced by CF\_CFDP\_S\_HandleFileRetention().

Here is the call graph for this function:



**12.34.2.21 CF\_CFDP\_GetTempName()** void CF\_CFDP\_GetTempName (

<i>const CF_History_t * hist,</i>
<i>char * FileNameBuf,</i>
<i>size_t FileNameSize )</i>

Get temporary file name.

Gets the full path of the temporary file for a transaction

## Assumptions, External Events, and Notes:

None

## Parameters

<i>hist</i>	History pointer
<i>FileNameBuf</i>	Buffer to store result
<i>FileNameSize</i>	Size of result buffer

Definition at line 2197 of file cf\_cfdp.c.

References CF\_AppData, CF\_FILENAME\_MAX\_PATH, CF\_AppData\_t::config\_table, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_ConfigTable::tmp\_dir.

Referenced by CF\_CFDP\_R\_HandleFileRetention(), and CF\_CFDP\_R\_Init().

**12.34.2.22 CF\_CFDP\_GetTxnStatus()** CF\_TxnStatus\_t CF\_CFDP\_GetTxnStatus (

<i>const CF_Transaction_t * txn )</i>
---------------------------------------

Helper function to retrieve transaction status code only.

This retrieves the status from the history block.

## Assumptions, External Events, and Notes:

txn must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

**Returns**

Status Code value set within transaction

Definition at line 2028 of file cf\_cfdp.c.

References CF\_TxnStatus\_NO\_RESOURCE, CF\_Transaction::history, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_TxnIsOK().

**12.34.2.23 CF\_CFDP\_InitEngine()** [CFE\\_Status\\_t](#) CF\_CFDP\_InitEngine ( void )

Initialization function for the CFDP engine.

**Description**

Performs all initialization of the CFDP engine

**Assumptions, External Events, and Notes:**

Only called once.

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
-----------------------------	--

**Returns**

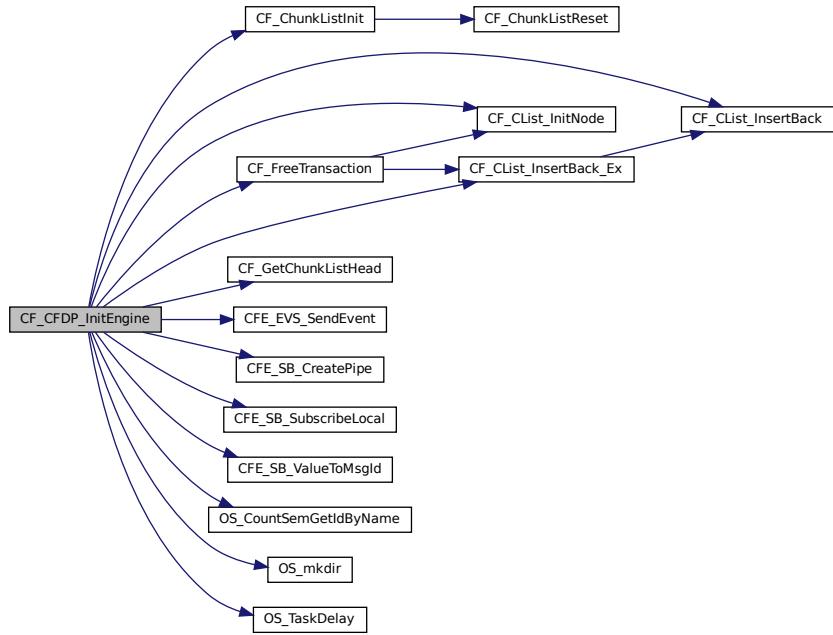
anything else on error.

Definition at line 1135 of file cf\_cfdp.c.

References CF\_AppData, CF\_ASSERT, CF\_CHANNEL\_NUM\_RX\_CHUNKS\_PER\_TRANSACTION, CF\_CHANNEL\_NUM\_TX\_CHUNKS\_PER\_TRANSACTION, CF\_CHANNEL\_PIPE\_PREFIX, CF\_ChunkListInit(), CF\_CList\_InitNode(), CF\_CList\_InsertBack(), CF\_CList\_InsertBack\_Ex(), CF\_CR\_CHANNEL\_PIPE\_ERR\_EID, CF\_Direction\_NUM, CF\_FreeTransaction(), CF\_GetChunkListHead(), CF\_INIT\_SEM\_ERR\_EID, CF\_INIT\_SUB\_ERR\_EID, CF\_NUM\_CHANNELS, CF\_NUM\_CHUNKS\_ALL\_CHANNELS, CF\_NUM\_HISTORIES\_PER\_CHANNEL, CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL, CF\_QueueIdx\_HIST\_FREE, CF\_STARTUP\_SEM\_MAX\_RETRIES, CF\_STARTUP\_SEM\_TASK\_DELAY, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SB\_CreatePipe(), CFE\_SB\_SubscribeLocal(), CFE\_SB\_ValueToMsgId(), CFE\_SUCCESS, CF\_ConfigTable::chan, CF\_Engine::channels, CF\_Engine::chunk\_mem, CF\_ChunkWrapper::chunks, CF\_Engine::chunks, CF\_History::cl\_node, CF\_ChunkWrapper::cl\_node, CF\_AppData\_t::config\_table, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_Engine::histories, CF\_ChannelConfig::mid\_input, OS\_CountSemGetIdByName(), OS\_ERR\_NAME\_NOT\_FOUND, OS\_mkdir(), OS\_SUCCESS, OS\_TaskDelay(), CF\_Channel::pipe, CF\_ChannelConfig::pipe\_depth\_input, CF\_Channel::sem\_id, CF\_ChannelConfig::sem\_name, CF\_ConfigTable::tmp\_dir, and CF\_Engine::transactions.

Referenced by CF\_AppInit(), and CF\_EnableEngineCmd().

Here is the call graph for this function:



```
12.34.2.24 CF_CFDP_InitTxnTxFile() void CF_CFDP_InitTxnTxFile (
    CF_Transaction_t * txn,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority )
```

Helper function to set tx file state in a transaction.

This sets various fields inside a newly-allocated transaction structure appropriately for sending a file.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction state
<i>cfdp_class</i>	Set to class 1 or class 2
<i>keep</i>	Whether to keep the local file
<i>chan</i>	CF channel number
<i>priority</i>	Priority of transfer

Definition at line 1486 of file cf\_cfdp.c.

References CF\_TxnState\_S1, CF\_TxnState\_S2, CF\_Transaction::chan\_num, CF\_Transaction::keep, CF\_Transaction::priority, CF\_Transaction::reliable\_mode, and CF\_Transaction::state.

Referenced by CF\_CFDP\_TxFile\_Initiate().

**12.34.2.25 CF\_CFDP\_IsSender()** static bool CF\_CFDP\_IsSender (   
     CF\_Transaction\_t \* txn ) [inline], [static]

Definition at line 145 of file cf\_cfdp.c.

References CF\_ASSERT, CF\_Direction\_TX, CF\_History::dir, and CF\_Transaction::history.

Referenced by CF\_CFDP\_SendAck().

**12.34.2.26 CF\_CFDP\_PlaybackDir()** CFE\_Status\_t CF\_CFDP\_PlaybackDir (   
     const char \* src\_filename,   
     const char \* dst\_filename,   
     CF\_CFDP\_Class\_t cfdp\_class,   
     uint8 keep,   
     uint8 chan,   
     uint8 priority,   
     uint16 dest\_id )

Begin transmit of a directory.

#### Description

This function sets up CF\_Playback\_t structure with state so it can become part of the directory polling done at each engine cycle.

#### Assumptions, External Events, and Notes:

src\_filename must not be NULL. dst\_filename must not be NULL.

#### Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename
<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

#### Return values

<b>CFE_SUCCESS</b>	Successful execution. Operation was performed successfully
--------------------	--

#### Returns

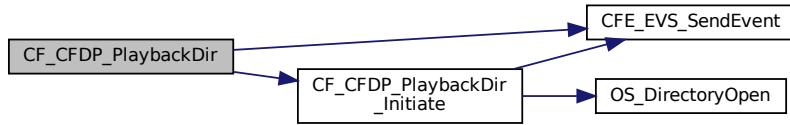
CFE\_SUCCESS on success. CF\_ERROR on error.

Definition at line 1647 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_DIR\_SLOT\_ERR\_EID, CF\_CFDP\_PlaybackDir\_Initiate(), CF\_ERROR, CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Engine::channels, CF\_AppData\_t::engine, and CF\_Channel::playback.

Referenced by CF\_PlaybackDirCmd().

Here is the call graph for this function:



#### 12.34.2.27 CF\_CFDP\_PlaybackDir\_Initiate()

```

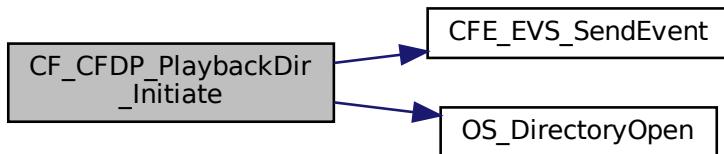
static CFE_Status_t CF_CFDP_PlaybackDir_Initiate (
    CF_Playback_t * pb,
    const char * src_filename,
    const char * dst_filename,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority,
    CF_EntityId_t dest_id ) [static]
  
```

Definition at line 1607 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_OPENDIR\_ERR\_EID, CF\_Playback::cfdp\_class, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Playback::dest\_id, CF\_Playback::dir\_id, CF\_HkFault::directory\_read, CF\_Playback::diopen, CF\_TxnFilenames::dst\_filename, CF\_HkCounters::fault, CF\_Playback::fnames, CF\_AppData\_t::hk, CF\_Playback::keep, OS\_DirectoryOpen(), OS\_SUCCESS, CF\_HkPacket::Payload, CF\_Playback::priority, and CF\_TxnFilenames::src\_filename.

Referenced by CF\_CFDP\_PlaybackDir(), and CF\_CFDP\_ProcessPollingDirectories().

Here is the call graph for this function:



#### 12.34.2.28 CF\_CFDP\_ProcessPlaybackDirectories()

```

static void CF_CFDP_ProcessPlaybackDirectories (
    CF_Channel_t * chan ) [static]
  
```

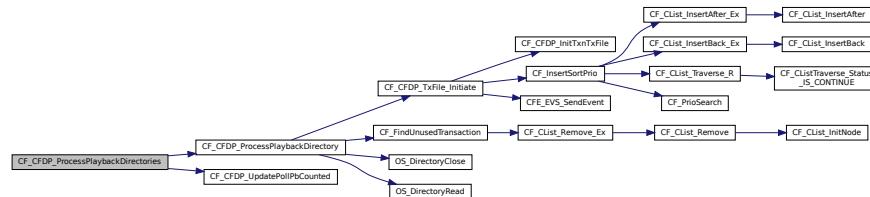
Definition at line 1774 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_UpdatePollPbCounted(), CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN, CF\_HkPacket\_Payload::channel\_hk,

CF\_Engine::channels, CF\_AppData\_t::engine, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Channel::playback, and CF\_HkChannel\_Data::playback\_counter.

Referenced by CF\_CFDP\_CycleEngine().

Here is the call graph for this function:



### 12.34.2.29 CF\_CFDP\_ProcessPlaybackDirectory()

```
void CF_CFDP_ProcessPlaybackDirectory (
    CF_Channel_t * chan,
    CF_Playback_t * pb )
```

Step each active playback directory.

#### Description

Check if a playback directory needs iterated, and if so does, and if a valid file is found initiates playback on it.

#### Assumptions, External Events, and Notes:

chan must not be NULL, pb must not be NULL.

#### Parameters

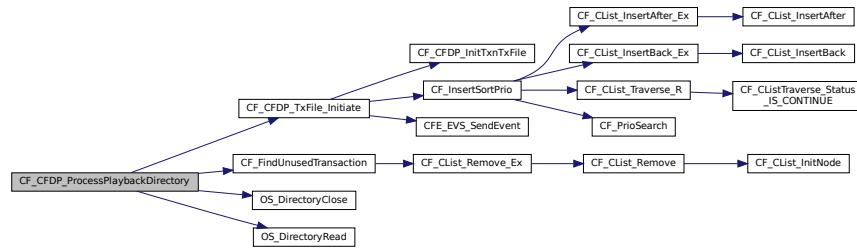
<i>chan</i>	The channel associated with the playback
<i>pb</i>	The playback state

Definition at line 1677 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_TxFile\_Initiate(), CF\_Direction\_TX, CF\_FILENAME\_MAX\_NAME, CF\_FILENAME\_MAX\_PATH, CF\_FindUnusedTransaction(), CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK, CF\_PERF\_ID\_DIRREAD, CF\_Playback::cfdp\_class, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CF\_Engine::channels, CF\_Playback::dest\_id, CF\_Playback::dir\_id, CF\_Playback::diropen, CF\_TxnFilenames::dst\_filename, CF\_AppData\_t::engine, os\_dirent\_t::FileName, CF\_History::fnames, CF\_Playback::fnames, CF\_Transaction::history, CF\_Playback::keep, CF\_Playback::num\_ts, OS\_DirectoryClose(), OS\_DirectoryRead(), OS\_DIRENTRY\_NAME, OS\_SUCCESS, CF\_Transaction::pb, CF\_Playback::pending\_file, CF\_Playback::priority, and CF\_TxnFilenames::src\_filename.

Referenced by CF\_CFDP\_ProcessPlaybackDirectories(), and CF\_CFDP\_ProcessPollingDirectories().

Here is the call graph for this function:



### 12.34.2.30 CF\_CFDP\_ProcessPollingDirectories()

```
void CF_CFDP_ProcessPollingDirectories (
    CF_Channel_t * chan )
```

Kick the dir playback if timer elapsed.

#### Description

This function waits for the polling directory interval timer, and if it has expired, starts a playback in the polling directory.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

#### Parameters

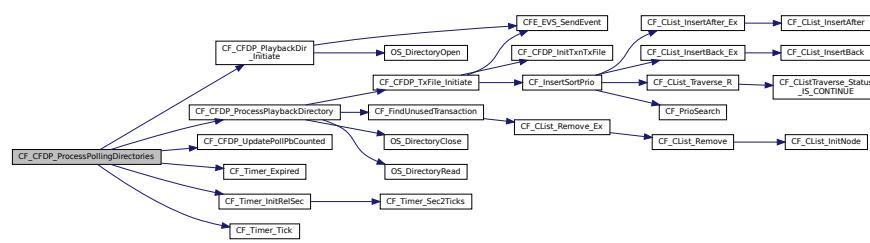
<code>chan</code>	The channel associated with the playback
-------------------	--

Definition at line 1793 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_UpdatePollPbCounted(), CF\_MAX\_POLLING\_DIR\_PER\_CHAN, CF\_Timer\_Expired(), CF\_Timer\_InitRelSec(), CF\_Timer\_Tick(), CF\_PollDir::cfdp\_class, CF\_ConfigTable::chan, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_AppData\_t::config\_table, CF\_PollDir::dest\_eid, CF\_PollDir::dst\_dir, CF\_PollDir::enabled, CF\_AppData\_t::engine, CF\_AppData\_t::hk, CF\_PollDir::interval\_sec, CF\_Poll::interval\_timer, CF\_Playback::num\_ts, CF\_HkPacket::Payload, CF\_Poll::pb, CF\_Channel::poll, CF\_HkChannel\_Data::poll\_counter, CF\_ChannelConfig::polldir, CF\_PollDir::priority, CF\_PollDir::src\_dir, and CF\_Poll::timer\_set.

Referenced by CF\_CFDP\_CycleEngine().

Here is the call graph for this function:



```
12.34.2.31 CF_CFDP_ReceivePdu() void CF_CFDP_ReceivePdu (
    CF_Channel_t * chan,
    CF_Logical_PduBuffer_t * ph )
```

Receive PDU processing entry point.

Invoked from the transport interface (e.g. software bus or equivalent) after reception of a PDU from the network.

**Assumptions, External Events, and Notes:**

None

**Parameters**

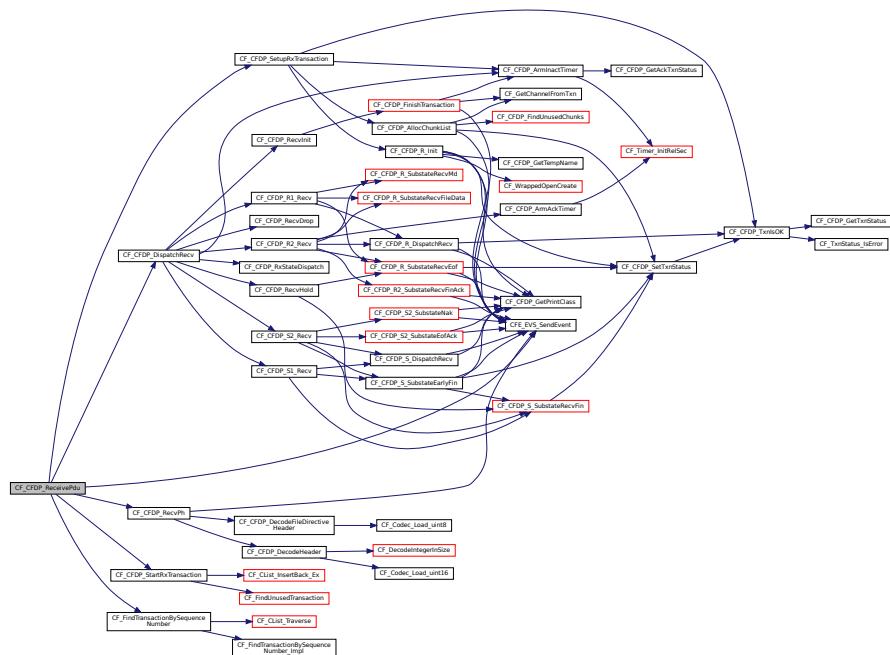
<i>chan</i>	Channel pointer
<i>ph</i>	Received PDU buffer

Definition at line 1076 of file cf\_cfdp.c.

References CF\_AppData, CF\_ASSERT, CF\_CFDP\_DispatchRecv(), CF\_CFDP\_INVALID\_DST\_ERR\_EID, CF\_CFDP\_RecvPh(), CF\_CFDP\_RX\_DROPPED\_ERR\_EID, CF\_CFDP\_SetupRxTransaction(), CF\_CFDP\_StartRxTransaction(), CF\_FindTransactionBySequenceNumber(), CF\_TxnState\_UNDEF, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Engine::channels, CF\_AppData\_t::config\_table, CF\_Logical\_PduHeader::destination\_eid, CF\_AppData\_t::engine, CF\_ConfigTable::local\_eid, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical\_PduHeader::source\_eid, and CF\_Transaction::state.

Referenced by CF\_CFDP\_ReceiveMessage().

Here is the call graph for this function:



```
12.34.2.32 CF_CFDP_RecvAck() CFE_Status_t CF_CFDP_RecvAck (
```

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Unpack an ACK PDU from a received message.

This should only be invoked for buffers that have been identified as an acknowledgment PDU.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

#### Returns

integer status code

#### Return values

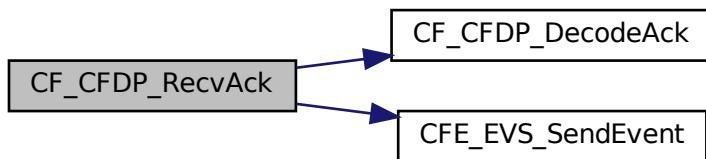
<i>CFE_SUCCESS</i>	on success
<i>CF_SHORT_PDU_ERROR</i>	on error

Definition at line 840 of file cf\_cfdp.c.

References CF\_Logical\_IntHeader::ack, CF\_CFDP\_DecodeAck(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_ACK\_SHORT\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Logical\_PduBuffer::int\_header, and CF\_Logical\_PduBuffer::pdec.

Referenced by CF\_CFDP\_R2\_SubstateRecvFinAck(), and CF\_CFDP\_S2\_SubstateEofAck().

Here is the call graph for this function:



**12.34.2.33 CF\_CFDP\_RecvDrop()** void CF\_CFDP\_RecvDrop (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Receive state function to ignore a packet.

**Description**

This function signature must match all receive state functions. The parameter *txn* is ignored here.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 909 of file cf\_cfdp.c.

References CF\_AppData, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_HkRecv::dropped, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_HkCounters::recv.  
Referenced by CF\_CFDP\_DispatchRecv().

```
12.34.2.34 CF_CFDP_RecvEof() CFE_Status_t CF_CFDP_RecvEof (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Unpack an EOF PDU from a received message.

This should only be invoked for buffers that have been identified as an end of file PDU.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

**Returns**

integer status code

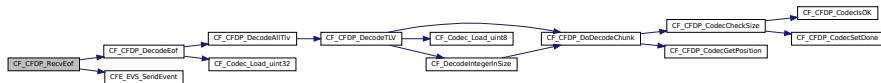
**Return values**

CFE_SUCCESS	on success
CF_SHORT_PDU_ERROR	on error

Definition at line 818 of file cf\_cfdp.c.

References CF\_CFDP\_DecodeEof(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_EOF\_SHORT\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVT\_EventType\_ERROR, CFE\_EVT\_SendEvent(), CFE\_SUCCESS, CF\_Logical\_IntHeader::eof, CF\_Logical\_PduBuffer::int\_header, and CF\_Logical\_PduBuffer::pdec.  
Referenced by CF\_CFDP\_R\_SubstateRecvEof().

Here is the call graph for this function:



**12.34.2.35 CF\_CFDP\_RecvFd()** `CFE_Status_t CF_CFDP_RecvFd (`  
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph )`

Unpack a file data PDU from a received message.

This should only be invoked for buffers that have been identified as a file data PDU.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

#### Returns

integer status code

#### Return values

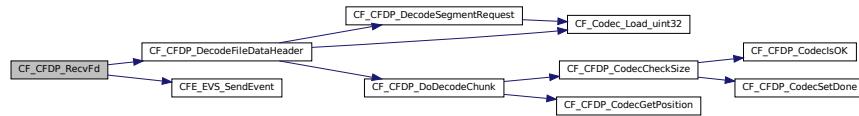
<code>CFE_SUCCESS</code>	on success
<code>CF_ERROR</code>	for general errors
<code>CF_SHORT_PDU_ERROR</code>	PDU too short

Definition at line 774 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_DecodeFileDataHeader(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_ERROR, CF\_PDU\_FD\_SHORT\_ERR\_EID, CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduHeader::crc\_flag, CF\_Logical\_PduFileDataHeader::data\_len, CF\_HkRecv::error, CF\_Logical\_IntHeader::fd, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdec, CF\_Logical\_PduBuffer::pdu\_header, CF\_HkCounters::recv, and CF\_Logical\_PduHeader::segment\_meta\_flag.

Referenced by CF\_CFDP\_R\_SubstateRecvFileData().

Here is the call graph for this function:



**12.34.2.36 CF\_CFDP\_RecvFin()** `CFE_Status_t CF_CFDP_RecvFin (`  
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph )`

Unpack an FIN PDU from a received message.

This should only be invoked for buffers that have been identified as a final PDU.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

#### Returns

integer status code

#### Return values

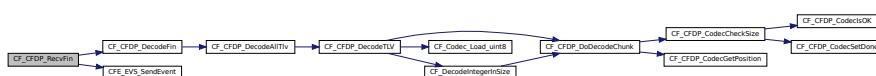
<code>CFE_SUCCESS</code>	on success
<code>CF_SHORT_PDU_ERROR</code>	on error

Definition at line 863 of file cf\_cfdp.c.

References CF\_CFDP\_DecodeFin(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_FIN\_SHORT\_ERR\_←  
 EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_←  
 Logical\_IntHeader::fin, CF\_Logical\_PduBuffer::int\_header, and CF\_Logical\_PduBuffer::pdec.

Referenced by CF\_CFDP\_S\_SubstateRecvFin().

Here is the call graph for this function:



**12.34.2.37 CF\_CFDP\_RecvHold()** `void CF_CFDP_RecvHold (`  
`CF_Transaction_t * txn,`

CF\_Logical\_PduBuffer\_t \* ph )

Receive state function during holdover period.

## Description

This function signature must match all receive state functions. Handles any possible spurious PDUs that might come in after the transaction is considered done. This can happen if ACKs were lost in transmission causing the sender to retransmit PDUs even though we already completed the transaction.

## Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

## Parameters

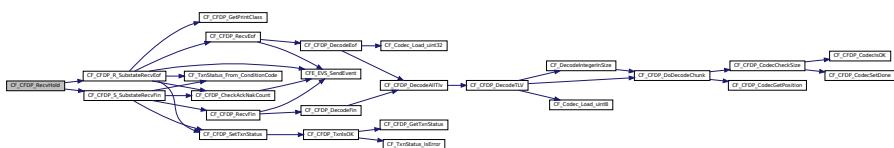
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 920 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_R\_SubstateRecv, CF\_Eof(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_Direction\_TX, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_History::dir, CF\_Logical\_PduFileDirectiveHeader::directive\_code, CF\_Logical\_PduBuffer::fdirective, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_HkCounters::recv, and CF\_HkRecv::spurious.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.34.2.38 CF\_CFDP\_RecvInit()** void CF\_CFDP\_RecvInit (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Receive state function to process new rx transaction.

### Description

An idle transaction has never had message processing performed on it. Typically, the first packet received for a transaction would be the metadata PDU. There's a special case for R2 where the metadata PDU could be missed, and filedata comes in instead. In that case, an R2 transaction must still be started.

## Assumptions, External Events, and Notes:

`txn` must not be `NULL`. There must be a received message.

### Parameters

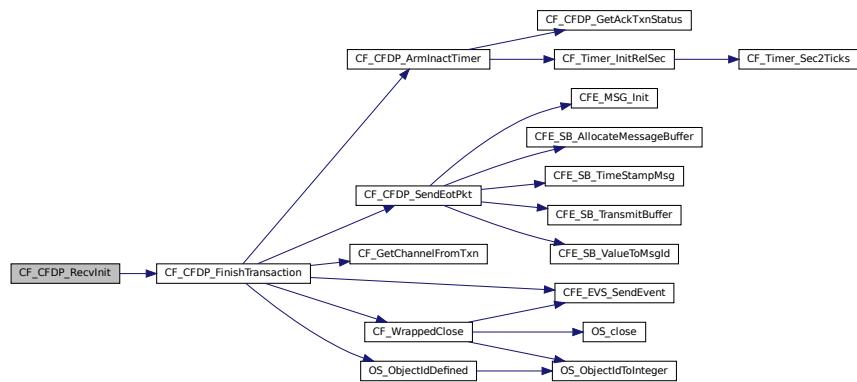
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 965 of file cf\_cfdp.c.

References CF\_CFDP\_FinishTransaction().

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.34.2.39 CF\_CFDP\_RecvMd()** *CFE\_Status\_t* CF\_CFDP\_RecvMd ( *CF\_Transaction\_t* \* *txn*, *CF\_Logical\_PduBuffer\_t* \* *ph* )

Unpack a metadata PDU from a received message.

This should only be invoked for buffers that have been identified as a metadata PDU.

### Assumptions, External Events, and Notes:

*txn* must not be NULL.

### Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

### Returns

integer status code

### Return values

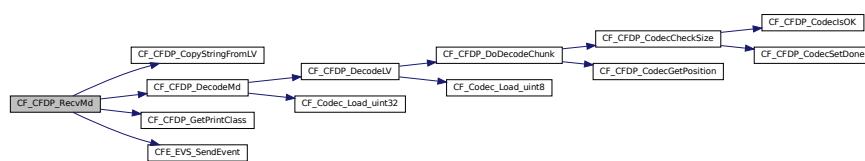
<i>CFE_SUCCESS</i>	on success
<i>CF_PDU_METADATA_ERROR</i>	on error

Definition at line 701 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_CopyStringFromLV(), CF\_CFDP\_DecodeMd(), CF\_CFDP\_GetPrintClass(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_INVALID\_DST\_LEN\_ERR\_EID, CF\_PDU\_INVALID\_SRC\_LEN\_ERR\_EID, CF\_PDU\_MD\_RECVD\_INF\_EID, CF\_PDU\_MD\_SHORT\_ERR\_EID, CF\_PDU\_METADATA\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_Flags\_Common::close\_req, CF\_Logical\_PduMd::close\_req, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_Logical\_PduMd::dest\_filename, CF\_TxnFilenames::dst\_filename, CF\_HkRecv::error, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::fsiz, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_Lv::length, CF\_Logical\_IntHeader::md, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdec, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_Logical\_PduMd::size, CF\_Logical\_PduMd::source\_filename, CF\_History::src\_eid, and CF\_TxnFilenames::src\_filename.

Referenced by CF\_CFDP\_R\_SubstateRecvMd().

Here is the call graph for this function:



#### 12.34.2.40 CF\_CFDP\_RecvNak() CFE\_Status\_t CF\_CFDP\_RecvNak (

```

    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph
)
```

Unpack a NAK PDU from a received message.

This should only be invoked for buffers that have been identified as a negative/non-acknowledgment PDU.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

txn	Pointer to the transaction state
ph	The logical PDU buffer being received

#### Returns

integer status code

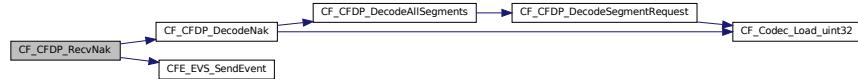
#### Return values

CFE_SUCCESS	on success
CF_SHORT_PDU_ERROR	on error

Definition at line 887 of file cf\_cfdp.c.

References CF\_CFDP\_DecodeNak(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_NAK\_SHORT\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_

Logical\_PduBuffer::int\_header, CF\_Logical\_IntHeader::nak, and CF\_Logical\_PduBuffer::pdec.  
Referenced by CF\_CFDP\_S2\_SubstateNak().  
Here is the call graph for this function:



**12.34.2.41 CF\_CFDP\_RecvPh()** `CFE_Status_t CF_CFDP_RecvPh ( uint8 chan_num, CF_Logical_PduBuffer_t * ph )`

Unpack a basic PDU header from a received message.

#### Description

This interprets the common PDU header and the file directive header (if applicable) and populates the logical PDU buffer.

#### Assumptions, External Events, and Notes:

A new message has been received.

#### Parameters

<code>chan_num</code>	The channel number for statistics purposes
<code>ph</code>	The logical PDU buffer being received

#### Returns

integer status code

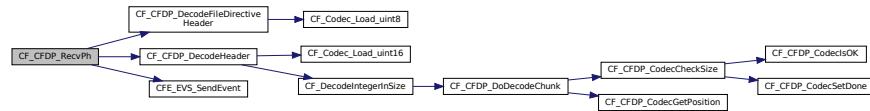
#### Return values

<code>CFE_SUCCESS</code>	on success
<code>CF_ERROR</code>	for general errors
<code>CF_SHORT_PDU_ERROR</code>	if PDU too short

Definition at line 641 of file cf\_cfdp.c.

References CF\_AppData, CF\_ASSERT, CF\_CFDP\_DecodeFileDirectiveHeader(), CF\_CFDP\_DecodeHeader(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_ERROR, CF\_NUM\_CHANNELS, CF\_PDU\_LARGE\_FILE\_ERR\_EID, CF\_PDU\_SHORT\_HEADER\_ERR\_EID, CF\_PDU\_TRUNCATION\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_HkRecv::error, CF\_Logical\_PduBuffer::fdirective, CF\_AppData\_t::hk, CF\_Logical\_PduHeader::large\_flag, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdec, CF\_HkRecv::pdu, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::pdu\_type, and CF\_HkCounters::recv.  
Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



#### 12.34.2.42 CF\_CFDP\_RecycleTransaction()

```
void CF_CFDP_RecycleTransaction (
```

```
    CF_Transaction_t * txn )
```

Recover resources associated with a transaction.

Wipes all data in the transaction struct and returns everything to its relevant FREE list so it can be used again.

Notably, should any PDUs arrive after this that is related to this transaction, these PDUs will not be identifiable, and no longer associative to this transaction.

##### Assumptions, External Events, and Notes:

It is imperative that nothing uses the `txn` struct after this call, as it will now be invalid. This is effectively like `free()`.

##### Parameters

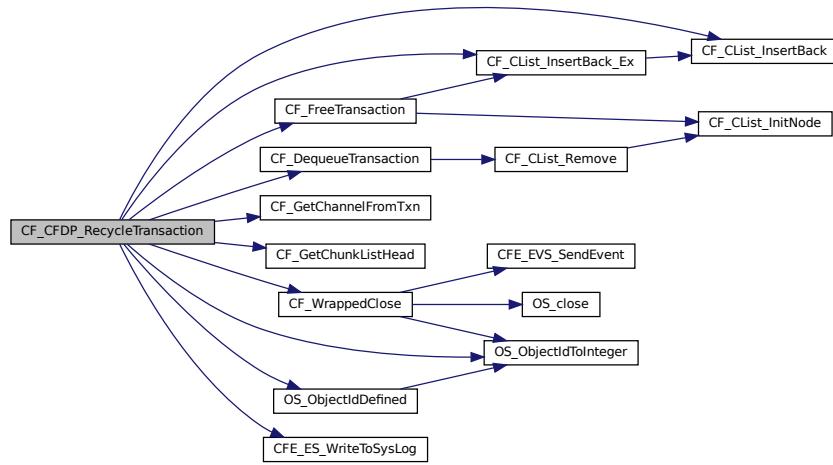
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 1954 of file cf\_cfdp.c.

References `CF_CList_InsertBack()`, `CF_CList_InsertBack_Ex()`, `CF_DequeueTransaction()`, `CF_FreeTransaction()`, `CF_GetChannelFromTxn()`, `CF_GetChunkListHead()`, `CF_QueueIdx_HIST`, `CF_QueueIdx_HIST_FREE`, `CF_TRACE`, `CF_WrappedClose()`, `CFE_ES_WriteToSysLog()`, `CF_Transaction::chan_num`, `CF_Transaction::chunks`, `CF_History::cl_node`, `CF_ChunkWrapper::cl_node`, `CF_StateFlags::com`, `CF_History::dir`, `CF_Transaction::fd`, `CF_Transaction::flags`, `CF_Transaction::history`, `CF_Flags_Common::keep_history`, `OS_OBJECT_ID_UNDEFINED`, `OS_ObjectIdDefined()`, `OS_ObjectIdToInteger()`, and `CF_History::seq_num`.

Referenced by `CF_CFDP_R_Tick()`, `CF_CFDP_S_Tick()`, and `CF_CFDP_SetupTxTransaction()`.

Here is the call graph for this function:



#### 12.34.2.43 CF\_CFDP\_S\_Tick\_NewData()

```
void CF_CFDP_S_Tick_NewData (
    CF_Transaction_t * txn )
```

Tick processor to send new file data.

This helper is used in conjunction with [CF\\_CList\\_Traverse\(\)](#) as part of Tick Processing.

##### Description

This sends new file data, which constitutes chunks of the file that have never been transmitted before.

This attempts to fill the output pipe as much as possible - it will create new PDUs until a transmission limit is reached. As such, it should run only after control traffic is handled. This will consume any unused bandwidth at the end of each wakeup.

##### Assumptions, External Events, and Notes:

*txn* must not be NULL.

##### Parameters

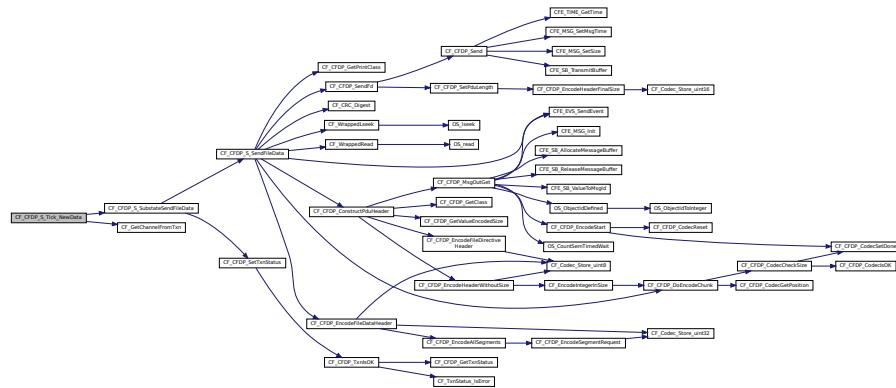
<i>txn</i>	Txn
------------	-----

Definition at line 1251 of file cf\_cfdp.c.

References [CF\\_CFDP\\_S\\_SubstateSendFileData\(\)](#), [CF\\_GetChannelFromTxn\(\)](#), [CF\\_PERF\\_ID\\_PDUSENT](#), [CF\\_TxSubState\\_DATA\\_NORMAL](#), [CFE\\_ES\\_PerfLogEntry](#), [CFE\\_ES\\_PerfLogExit](#), [CF\\_Transaction::chan\\_num](#), [CF\\_StateFlags::com](#), [CF\\_Transaction::flags](#), [CF\\_Channel::outgoing\\_counter](#), [CF\\_Transaction::state\\_data](#), [CF\\_StateData::sub\\_state](#), and [CF\\_Flags\\_Common::suspended](#).

Referenced by [CF\\_CFDP\\_TickTransactions\(\)](#).

Here is the call graph for this function:



**12.34.2.44 CF\_CFDP\_SendAck()** `CFE_Status_t CF_CFDP_SendAck (`  
    `CF_Transaction_t * txn,`  
    `CF_CFDP_FileDirective_t dir_code )`

Build an ACK PDU for transmit.

## **Assumptions, External Events, and Notes:**

**txn** must not be `NULL`.

## Note

`CF_CFDP_SendAck()` takes a `CF_TransactionSeq_t` instead of getting it from transaction history because of the special case where a FIN-ACK must be sent for an unknown transaction. It's better for long term maintenance to not build an incomplete `CF_History_t` for it.

## Parameters

<i>txn</i>	Pointer to the transaction object
<i>dir_code</i>	File directive code being ACK'ed

## Returns

CFE\_Status\_t status code

## Return values

<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

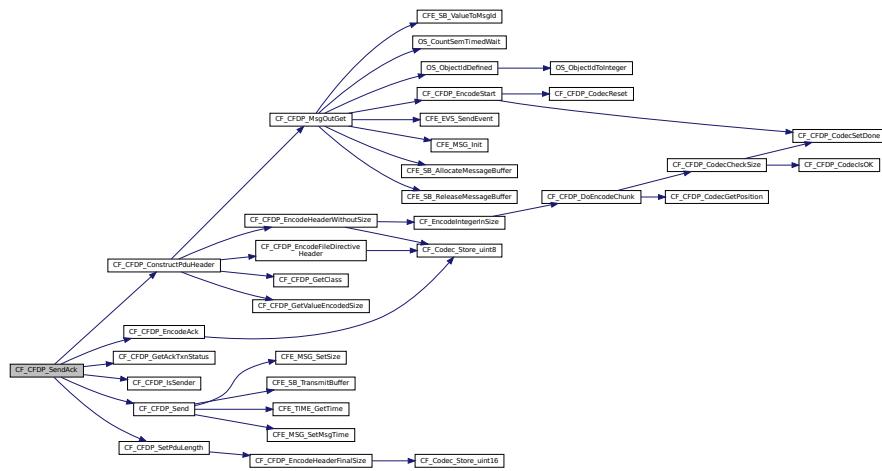
Definition at line 512 of file cf\_cfdp.c.

References CF\_Logical\_IntHeader::ack, CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::ack\_subtype\_code, CF\_Logical\_PduAck::cc, CF\_AppData, CF\_ASSERT, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeAck(), CF\_CFDP\_FileDirective\_ACK, CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_FileDirective\_FIN, CF\_

CFDP\_GetAckTxnStatus(), CF\_CFDP\_IsSender(), CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_SEND\_↔ PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TRACE, CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_AppData\_t::config↔ table, CF\_Transaction::history, CF\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_StateData::peer\_cc, CF\_History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_History::seq\_num, CF\_Transaction::state\_data, and CF↔ Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_R\_Tick\_Maintenance(), and CF\_CFDP\_S\_Tick\_Maintenance().

Here is the call graph for this function:



#### 12.34.2.45 CF\_CFDP\_SendEof()

```
CFE_Status_t CF_CFDP_SendEof (
    CFE_Transaction_t * txn )
```

Build an EOF PDU for transmit.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

txn	Pointer to the transaction object
-----	-----------------------------------

#### Returns

CFE\_Status\_t status code

#### Return values

CFE_SUCCESS	on success.
CF_SEND_PDU_NO_BUF_AVAIL_ERROR	if message buffer cannot be obtained.

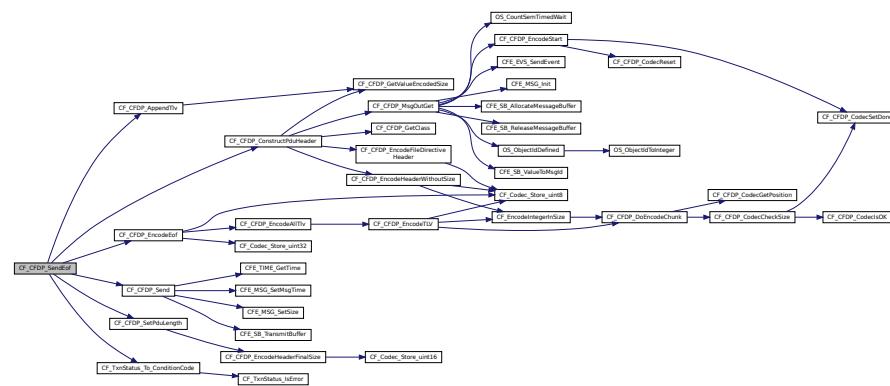
Definition at line 470 of file cf\_cfdp.c.

References CF\_Logical\_PduEof::cc, CF\_AppData, CF\_CFDP\_AppendTlv(), CF\_CFDP\_ConditionCode\_NO\_ERROR, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeEof(), CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_Send(), CF↔ CFDP\_SetPduLength(), CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TRACE,

CF\_TxnStatus\_To\_ConditionCode(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_AppData\_t::config\_table, CF\_Transaction::crc, CF\_Logical\_PduEof::crc, CF\_Logical\_IntHeader::eof, CF\_Transaction::fsize, CF\_Transaction::history, CF\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_Crc::result, CF\_History::seq\_num, CF\_Logical\_PduEof::size, CF\_Logical\_PduEof::tlv\_list, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_S\_Tick\_Maintenance().

Here is the call graph for this function:



#### 12.34.2.46 CF\_CFDP\_SendEotPkt()

```
void CF_CFDP_SendEotPkt (
    CF_Transaction_t * txn )
```

Send an end of transaction packet.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

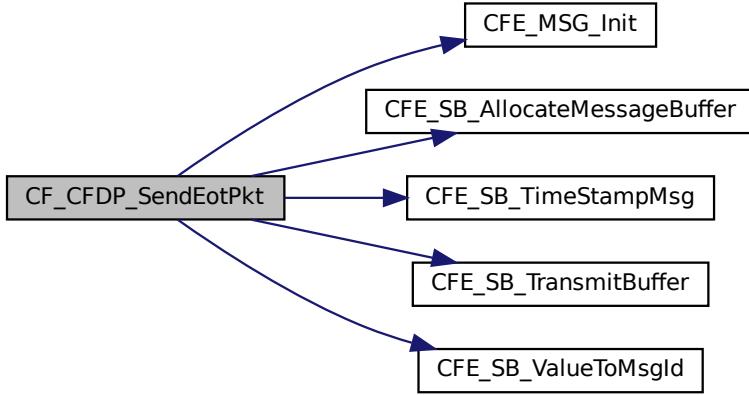
txn	Pointer to the transaction object
-----	-----------------------------------

Definition at line 2055 of file cf\_cfdp.c.

References CF\_EOT\_TLM\_MID, CFE\_MSG\_Init(), CFE\_SB\_AllocateMessageBuffer(), CFE\_SB\_TimeStampMsg(), CFE\_SB\_TransmitBuffer(), CFE\_SB\_ValueToMsgId(), CF\_Transaction::chan\_num, CF\_EotPacket\_Payload::channel, CF\_Transaction::crc, CF\_EotPacket\_Payload::crc\_result, CF\_History::dir, CF\_EotPacket\_Payload::direction, CF\_EotPacket\_Payload::fnames, CF\_History::fnames, CF\_EotPacket\_Payload::fsize, CF\_Transaction::fsize, CF\_Transaction::history, CF\_EotPacket::Payload, CF\_EotPacket\_Payload::peer\_eid, CF\_History::peer\_eid, CF\_Crc::result, CF\_EotPacket\_Payload::seq\_num, CF\_History::seq\_num, CF\_EotPacket\_Payload::src\_eid, CF\_History::src\_eid, CF\_EotPacket\_Payload::state, CF\_Transaction::state, CF\_EotPacket::TelemetryHeader, CF\_EotPacket\_Payload::txn\_stat, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_FinishTransaction().

Here is the call graph for this function:



**12.34.2.47 CF\_CFDP\_SendFd()** `CFE_Status_t CF_CFDP_SendFd(`  
 `CF_Transaction_t * txn,`  
 `CF_Logical_PduBuffer_t * ph )`

Send a previously-assembled filedata PDU for transmit.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to logical PDU buffer content

#### Note

Unlike other "send" routines, the file data PDU must be acquired and filled by the caller prior to invoking this routine. This routine only sends the PDU that was previously allocated and assembled. As such, the typical failure possibilities do not apply to this call.

#### Returns

`CFE_Status_t` status code

#### Return values

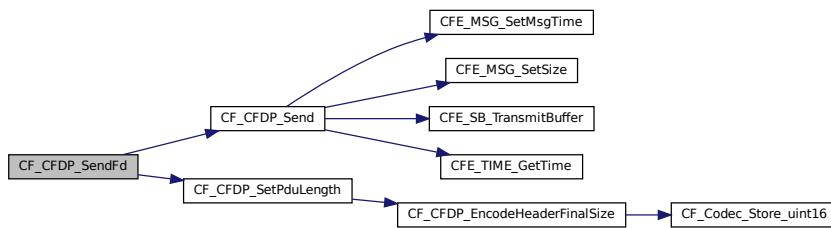
<code>CFE_SUCCESS</code>	on success. (error checks not yet implemented)
--------------------------	--

Definition at line 413 of file cf\_cfdp.c.

References CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CFE\_SUCCESS, and CF\_Transaction::chan\_num.

Referenced by CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:



#### 12.34.2.48 CF\_CFDP\_SendFin() [CFE\\_Status\\_t](#) CF\_CFDP\_SendFin ( [CF\\_Transaction\\_t](#) \* *txn* )

Build a FIN PDU for transmit.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

##### Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

##### Returns

[CFE\\_Status\\_t](#) status code

##### Return values

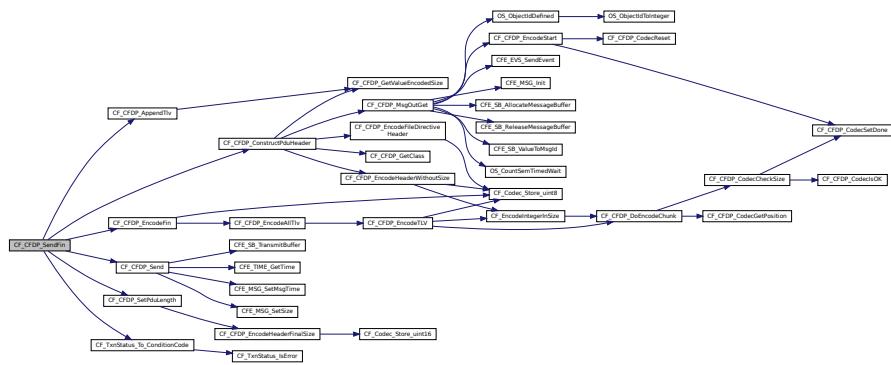
<a href="#">CFE_SUCCESS</a>	on success.
<a href="#">CF_SEND_PDU_NO_BUF_AVAIL_ERROR</a>	if message buffer cannot be obtained.

Definition at line 568 of file cf\_cfdp.c.

References CF\_Logical\_PduFin::cc, CF\_AppData, CF\_CFDP\_AppendTlv(), CF\_CFDP\_ConditionCode\_NO\_ERROR, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeFin(), CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TRACE, CF\_TxnStatus\_To\_ConditionCode(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_AppData\_t::config\_table, CF\_Logical\_PduFin::delivery\_code, CF\_Logical\_PduFin::file\_status, CF\_Logical\_IntHeader::fin, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::history, CF\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_History::seq\_num, CF\_Transaction::state\_data, CF\_Logical\_PduFin::tlv\_list, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_R\_Tick\_Maintenance().

Here is the call graph for this function:



**12.34.2.49 CF\_CFDP\_SendMd()** `CFE_Status_t CF_CFDP_SendMd ( CF_Transaction_t * txn )`

Build a metadata PDU for transmit.

## Assumptions, External Events, and Notes

text must not be RELE.

## Parameters

*txn* Pointer to the transaction object

## Returns

CFE\_Status\_t status code

## Return values

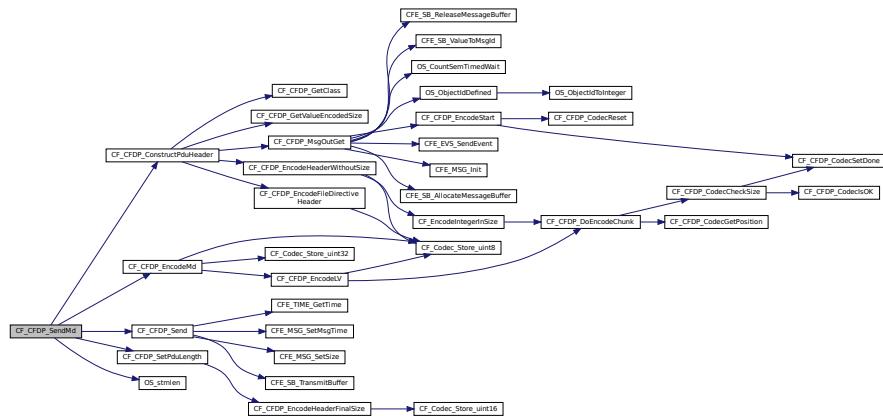
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 366 of file cf\_cfdp.c.

References CF\_AppData, CF\_Assert, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeMd(), CF\_CFDP\_File↔  
 Directive\_METADATA, CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR,  
 CF\_TRACE, CF\_TxnState\_S1, CF\_TxnState\_S2, CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_Logical\_Pdu↔  
 Md::checksum\_type, CF\_Flags\_Common::close\_req, CF\_Logical\_PduMd::close\_req, CF\_StateFlags::com, CF\_App↔  
 Data\_t::config\_table, CF\_Logical\_Lv::data\_ptr, CF\_Logical\_PduMd::dest\_filename, CF\_TxnFilenames::dst\_filename,  
 CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::fsize, CF\_Transaction::history, CF\_Logical\_PduBuffer↔  
 ::int\_header, CF\_Logical\_Lv::length, CF\_ConfigTable::local\_eid, CF\_Logical\_IntHeader::md, OS\_strlen(), CF↔  
 History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_History::seq\_num, CF\_Logical\_PduMd::size, CF\_Logical\_Pdu↔  
 Md::source\_filename, CF\_TxnFilenames::src\_filename, and CF\_Transaction::state.

Referenced by CF CFDP S Tick Maintenance().

Here is the call graph for this function:



**12.34.2.50 CF\_CFDP\_SendNak()** void CF\_CFDP\_SendNak (   
`CF_Transaction_t * txn,`   
`CF_Logical_PduBuffer_t * ph )`

Send a previously-assembled NAK PDU for transmit.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to logical PDU buffer content

#### Note

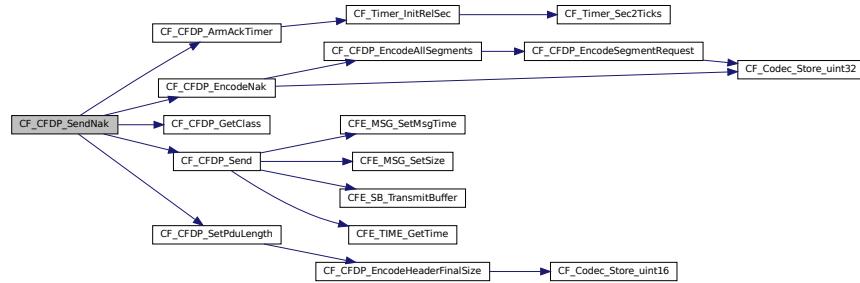
Unlike other "send" routines, the NAK PDU must be acquired and filled by the caller prior to invoking this routine. This routine only encodes and sends the previously-assembled PDU buffer. As such, the typical failure possibilities do not apply to this call.

Definition at line 612 of file cf\_cfdp.c.

References CF\_ASSERT, CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_CLASS\_2, CF\_CFDP\_EncodeNak(), CF\_CFDP\_GetClass(), CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_TRACE, CF\_Transaction::chan\_num, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_IntHeader::nak, CF\_Logical\_SegmentList::num\_segments, CF\_Logical\_PduBuffer::penc, and CF\_Logical\_PduNak::segment\_list.

Referenced by CF\_CFDP\_R\_SendNak().

Here is the call graph for this function:



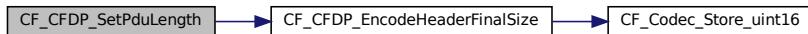
**12.34.2.51 CF\_CFDP\_SetPduLength()** static void CF\_CFDP\_SetPduLength ( CF\_Logical\_PduBuffer\_t \* ph ) [static]

Definition at line 274 of file cf\_cfdp.c.

References CF\_CFDP\_EncodeHeaderFinalSize(), CF\_CODEC\_GET\_POSITION, CF\_Logical\_PduHeader::data\_encoded\_length, CF\_Logical\_PduHeader::header\_encoded\_length, CF\_Logical\_PduBuffer::pdu\_header, and CF\_Logical\_PduBuffer::penc.

Referenced by CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFd(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), and CF\_CFDP\_SendNak().

Here is the call graph for this function:



**12.34.2.52 CF\_CFDP\_SetTxnStatus()** void CF\_CFDP\_SetTxnStatus ( CF\_Transaction\_t \* txn, CF\_TxnStatus\_t txn\_stat )

Helper function to store transaction status code only.

This records the status in the history block but does not set FIN flag or take any other protocol/state machine actions.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

txn	Pointer to the transaction object
txn_stat	Status Code value to set within transaction

Definition at line 2014 of file cf\_cfdp.c.

References CF\_CFDP\_TxnIsOK(), CF\_Transaction::history, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_AllocChunkList(), CF\_CFDP\_CancelTransaction(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_S\_SubstateSendFileData(), and CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



#### 12.34.2.53 CF\_CFDP\_SetupRxTransaction() void CF\_CFDP\_SetupRxTransaction (

```

    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph
)
```

Sets up a new RX transaction based on PDU.

Sets up a new RX transaction.

All RX transactions are initiated based on the reception of a PDU with this node as the dest entity and a sequence number that does not match any existing transaction.

When the given PDU requires a new transaction to be created, this initializes the new transaction object. It sets up the chunklist and all the other required supplemental data structs.

**Assumptions, External Events, and Notes:**

None

#### Parameters

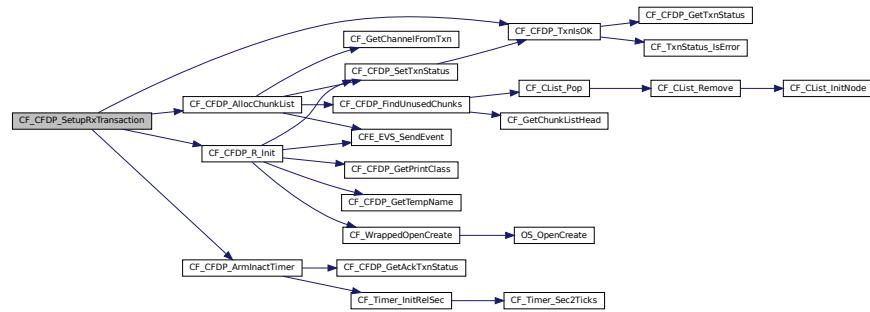
<i>txn</i>	Transaction pointer
<i>ph</i>	Received PDU buffer

Definition at line 1033 of file cf\_cfdp.c.

References CF\_CFDP\_AllocChunkList(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_R\_Init(), CF\_CFDP\_TxnIsOK(), CF\_TxnState\_HOLD, CF\_TxnState\_R1, CF\_TxnState\_R2, CF\_Transaction::chunks, CF\_Transaction::history, CF\_Logical\_PduBuffer::pdu\_header, CF\_History::peer\_eid, CF\_Transaction::reliable\_mode, CF\_History::seq\_num, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical\_PduHeader::source\_eid, CF\_History::src\_eid, CF\_Transaction::state, and CF\_Logical\_PduHeader::txm\_mode.

Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



#### 12.34.2.54 CF\_CFDP\_SetupTxTransaction()

```
void CF_CFDP_SetupTxTransaction (
    CF_Transaction_t * txn )
```

Sets up a new TX transaction.

TX transactions are initiated based on commands or configured polling dirs

It sets up the chunklist and all the other required supplemental data structs and puts the transaction onto the TX queue, so that it is now usable by the engine.

##### Assumptions, External Events, and Notes:

Transactions come from the PEND queue

##### Parameters

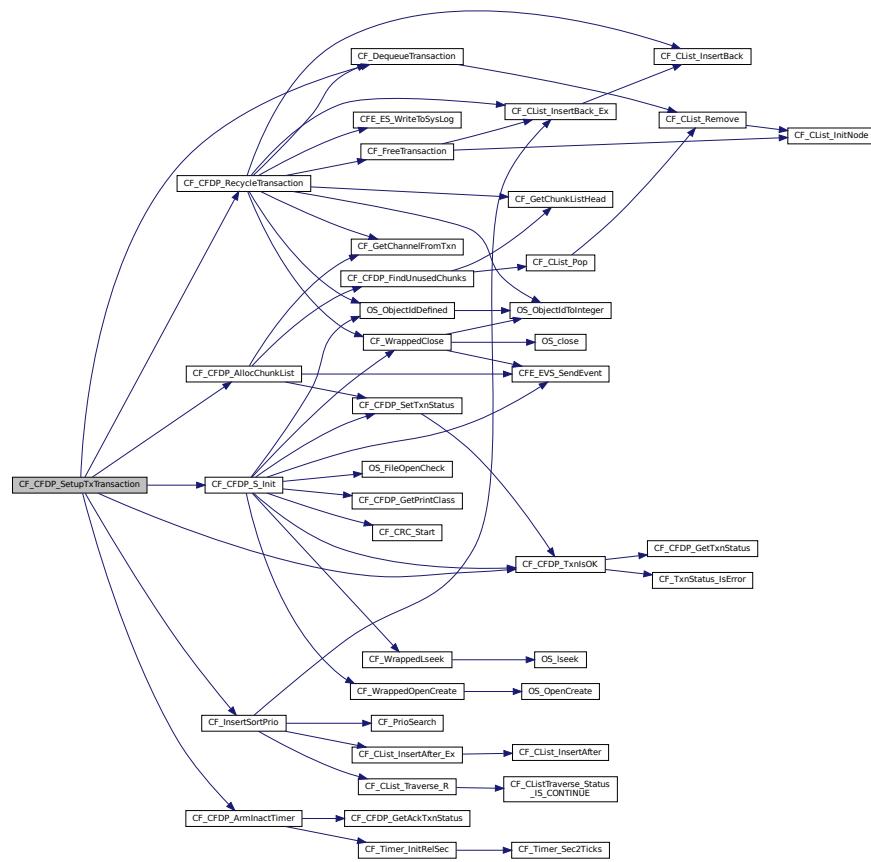
<i>txn</i>	Transaction pointer
------------	---------------------

Definition at line 998 of file cf\_cfdp.c.

References CF\_CFDP\_AllocChunkList(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S\_Init(), CF\_CFDP\_TxnIsOK(), CF\_DequeueTransaction(), CF\_InsertSortPrio(), CF\_QueueIdx\_TX, and CF\_Transaction::chunks.

Referenced by CF\_CFDP\_StartFirstPending().

Here is the call graph for this function:



**12.34.2.55 CF\_CFDP\_StartFirstPending()** `bool CF_CFDP_StartFirstPending ( CF_Channel_t * chan )`

Pulls next TX transaction from the PEND queue.

If the PEND queue is not empty, pulls the first transaction from it, and prepares for it to be activated.

**Assumptions, External Events, and Notes:**

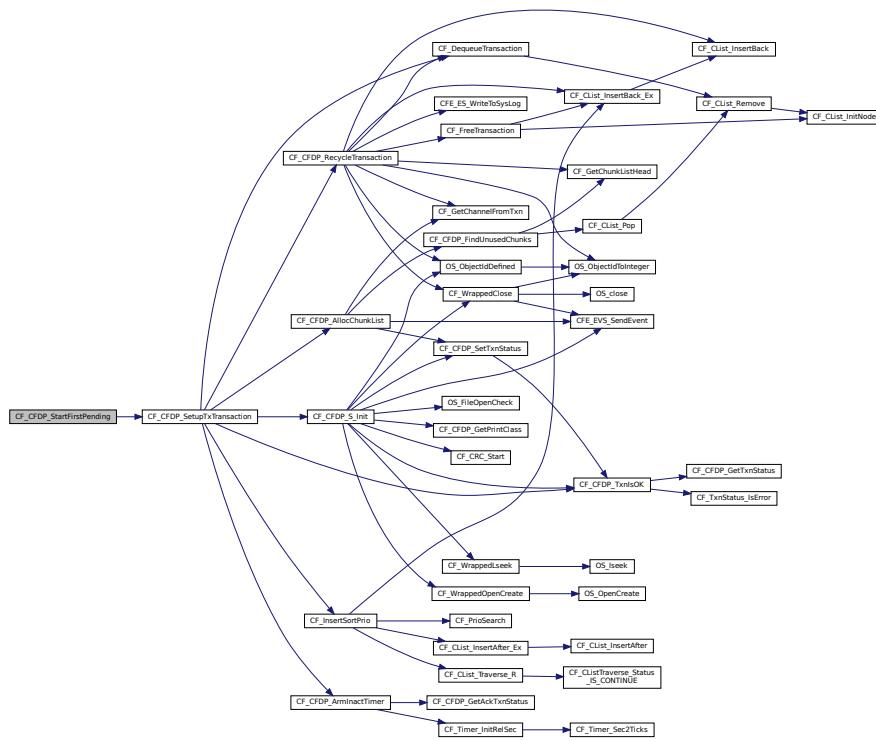
#### Parameters

<code>chan</code>	Channel pointer
-------------------	-----------------

Definition at line 1300 of file cf\_cfdp.c.

References CF\_CFDP\_SetupTxTransaction(), CF\_QueueIdx\_PEND, container\_of, and CF\_Channel::qs.  
Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



### 12.34.2.56 CF\_CFDP\_StartRxTransaction()

```
CF_Transaction_t* CF_CFDP_StartRxTransaction (
    uint8 chan_num )
```

Helper function to start a new RX transaction.

This sets various fields inside a newly-allocated transaction structure appropriately for receiving a file. Note that in the receive direction, most fields are unknown until the MD is received, and thus are left in their initial state here (generally 0).

If there is no capacity for another RX transaction, this returns NULL.

#### Parameters

<code>chan_num</code>	CF channel number
-----------------------	-------------------

#### Returns

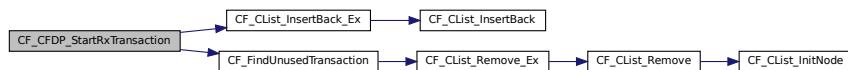
Pointer to new transaction

Definition at line 1576 of file cf\_cfdp.c.

References CF\_AppData, CF\_CList\_InsertBack\_Ex(), CF\_Direction\_RX, CF\_FindUnusedTransaction(), CF\_MAXSIMULTANEOUS\_RX, CF\_QueueIdx\_RX, CF\_TRACE, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Transaction::cl\_node, CF\_StateFlags::com, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Flags\_Common::q\_index, and CF\_HkChannel\_Data::q\_size.

Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



### 12.34.2.57 CF\_CFDP\_TickTransactions()

```
void CF_CFDP_TickTransactions (
```

```
    CF_Channel_t * chan )
```

Call R and then S tick functions for all active transactions.

#### Description

Traverses all transactions in the RX and TXW queues, and calls their tick functions. Note that the TXW queue is used twice: once for regular tick processing, and one for NAK response.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

#### Parameters

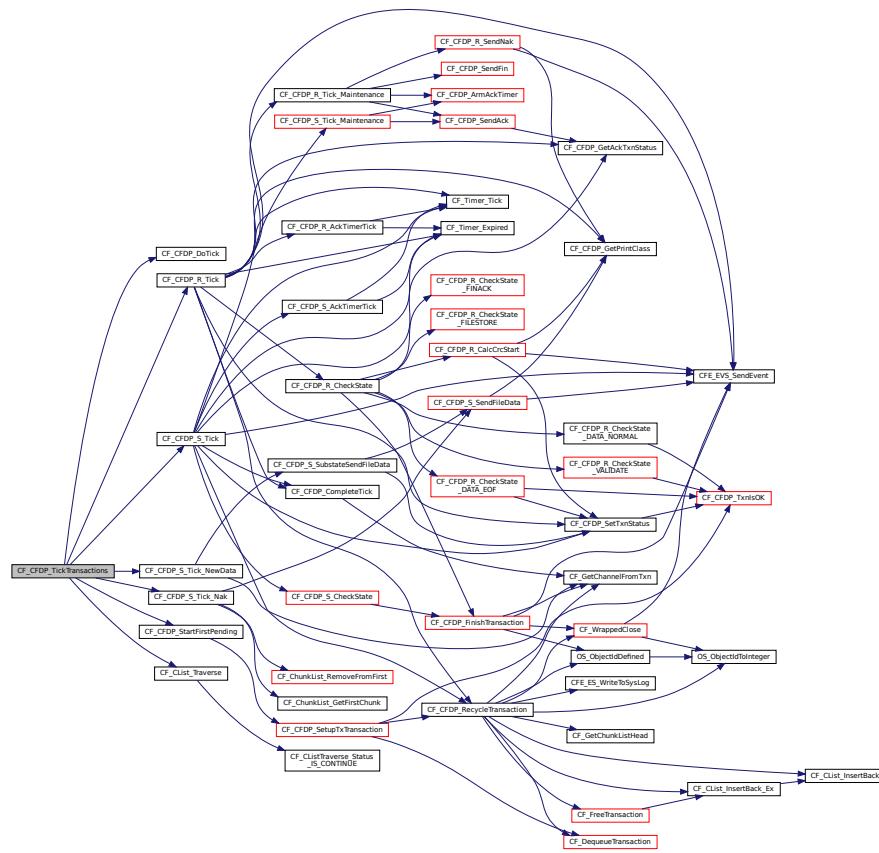
<code>chan</code>	Channel to tick
-------------------	-----------------

Definition at line 1375 of file cf\_cfdp.c.

References CF\_CFDP\_DoTick(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_S\_Tick\_Nak(), CF\_CFDP\_S\_Tick\_NewData(), CF\_CFDP\_StartFirstPending(), CF\_CList\_Traverse(), CF\_QueueIdx\_RX, CF\_QueueIdx\_TX, CF\_TickState\_COMPLETE, CF\_TickState\_INIT, CF\_TickState\_RX\_STATE, CF\_TickState\_TX\_FILEDATA, CF\_TickState\_TX\_NAK, CF\_TickState\_TX\_PEND, CF\_TickState\_TX\_STATE, CF\_CFDP\_Tick\_args::chan, CF\_CFDP\_Tick\_args::fn, CF\_Channel::outgoing\_counter, CF\_Channel::qs, CF\_CFDP\_Tick\_args::resume\_point, CF\_Channel::tick\_resume, and CF\_Channel::tx\_blocked.

Referenced by CF\_CFDP\_CycleEngine().

Here is the call graph for this function:



```
12.34.2.58 CF_CFDP_TxFile() CFE_Status_t CF_CFDP_TxFile (
    const char * src_filename,
    const char * dst_filename,
    CF_CFDP_Class_t cfcp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority,
    CF_EntityId_t dest_id )
```

Begin transmit of a file.

#### Description

This function sets up a transaction for and starts transmit of the given filename.

#### Assumptions, External Events, and Notes:

src\_filename must not be NULL. dst\_filename must not be NULL.

### Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename
<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

### Return values

<b>CFE_SUCCESS</b>	Successful execution. Operation was performed successfully
--------------------	--

### Returns

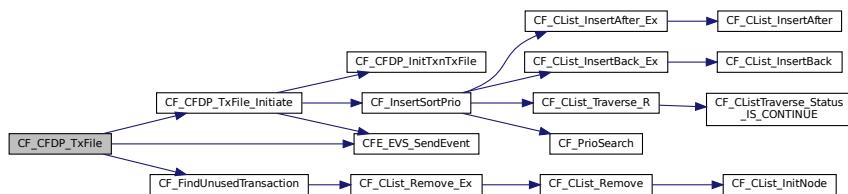
CFE\_SUCCESS on success. CF\_ERROR on error.

Definition at line 1528 of file cf\_cfdp.c.

References CF\_AppData, CF\_ASSERT, CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID, CF\_CFDP\_TxFile\_Initiate(), CF\_DIRECTION\_TX, CF\_ERROR, CF\_FindUnusedTransaction(), CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN, CF\_NUM\_CHANNELS, CF\_TRACE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Engine::channels, CF\_Flags\_Tx::cmd\_tx, CF\_TxnFilenames::dst\_filename, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::history, CF\_Channel::num\_cmd\_tx, CF\_TxnFilenames::src\_filename, and CF\_StateFlags::tx.

Referenced by CF\_TxFileCmd().

Here is the call graph for this function:



**12.34.2.59 CF\_CFDP\_TxFile\_Initiate()** static void CF\_CFDP\_TxFile\_Initiate (

```

CF_Transaction_t * txn,
CF_CFDP_Class_t cfdp_class,
uint8 keep,
uint8 chan,
uint8 priority,
CF_EntityId_t dest_id ) [static]
  
```

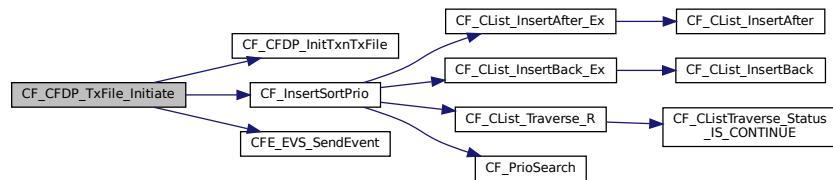
Definition at line 1500 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_InitTxnTxFile(), CF\_CFDP\_S\_START\_SEND\_INF\_EID, CF\_FILENAME\_MAX\_LEN, CF\_InsertSortPrio(), CF\_QueueIdx\_PEND, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(),

CF\_AppData\_t::config\_table, CF\_TxnFilenames::dst\_filename, CF\_AppData\_t::engine, CF\_History::fnames, CF\_Transaction::history, CF\_ConfigTable::local\_eid, CF\_History::peer\_eid, CF\_History::seq\_num, CF\_Engine::seq\_num, CF\_History::src\_eid, and CF\_TxnFilenames::src\_filename.

Referenced by CF\_CFDP\_ProcessPlaybackDirectory(), and CF\_CFDP\_TxFile().

Here is the call graph for this function:



#### 12.34.2.60 CF\_CFDP\_TxnIsOK() CF\_TxnStatus\_t CF\_CFDP\_TxnIsOK (

```
const CF_Transaction_t * txn )
```

Helper function to check if a transaction is errored.

Simplifies conditionals where state machines need to check if a transaction is OK to continue operating normally

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

**Return values**

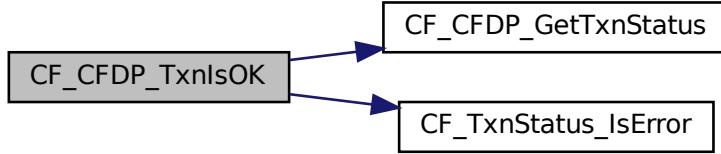
<i>true</i>	if transaction is in a good state, no errors
<i>false</i>	if transaction has an error

Definition at line 2044 of file cf\_cfdp.c.

References CF\_CFDP\_GetTxnStatus(), and CF\_TxnStatus\_IsError().

Referenced by CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_CheckState\_VALIDATE(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_SetTxnStatus(), CF\_CFDP\_SetupRxTransaction(), and CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



**12.34.2.61 CF\_CFDP\_UpdatePollPbCounted()** static void CF\_CFDP\_UpdatePollPbCounted ( CF\_Playback\_t \* pb, int up, uint8 \* counter ) [static]

Definition at line 1750 of file cf\_cfdp.c.

References CF\_ASSERT, and CF\_Playback::counted.

Referenced by CF\_CFDP\_ProcessPlaybackDirectories(), and CF\_CFDP\_ProcessPollingDirectories().

## 12.35 apps/cf/fsw/src/cf\_cfdp.h File Reference

```
#include "cf_cfdp_types.h"
```

### Data Structures

- struct [CF\\_CFDP\\_Tick\\_args](#)  
*Structure for use with the [CF\\_CFDP\\_DoTick\(\)](#) functions.*

### TypeDefs

- typedef struct [CF\\_CFDP\\_Tick\\_args](#) [CF\\_CFDP\\_Tick\\_args\\_t](#)  
*Structure for use with the [CF\\_CFDP\\_DoTick\(\)](#) functions.*

### Functions

- static int [CF\\_CFDP\\_GetPrintClass](#) (const [CF\\_Transaction\\_t](#) \*tx)
- Get printable CFDP class number.*
- void [CF\\_CFDP\\_EncodeStart](#) ([CF\\_EncoderState\\_t](#) \*penc, void \*msgbuf, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph, size\_t encaps\_hdr\_size, size\_t total\_size)
- Initiate the process of encoding a new PDU to send.*
- void [CF\\_CFDP\\_DecodeStart](#) ([CF\\_DecoderState\\_t](#) \*pdec, const void \*msgbuf, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph, size\_t encaps\_hdr\_size, size\_t total\_size)
- Initiate the process of decoding a received PDU.*
- void [CF\\_CFDP\\_FinishTransaction](#) ([CF\\_Transaction\\_t](#) \*tx, bool keep\_history)
- Finish a transaction.*

- void **CF\_CFDP\_RecycleTransaction** (**CF\_Transaction\_t** \*txn)  
*Recover resources associated with a transaction.*
- void **CF\_CFDP\_SetTxnStatus** (**CF\_Transaction\_t** \*txn, **CF\_TxnStatus\_t** txn\_stat)  
*Helper function to store transaction status code only.*
- **CF\_TxnStatus\_t CF\_CFDP\_GetTxnStatus** (**const CF\_Transaction\_t** \*txn)  
*Helper function to retrieve transaction status code only.*
- **CF\_TxnStatus\_t CF\_CFDP\_TxnIsOK** (**const CF\_Transaction\_t** \*txn)  
*Helper function to check if a transaction is errored.*
- void **CF\_CFDP\_SendEotPkt** (**CF\_Transaction\_t** \*txn)  
*Send an end of transaction packet.*
- bool **CF\_CFDP\_CheckAckNakCount** (**CF\_Transaction\_t** \*txn, **uint8** \*counter)  
*Increment ack/nak counter and check limit.*
- void **CF\_CFDP\_CompleteTick** (**CF\_Transaction\_t** \*txn)  
*Complete tick processing on a transaction.*
- **CFE\_Status\_t CF\_CFDP\_InitEngine** (**void**)  
*Initialization function for the CFDP engine.*
- void **CF\_CFDP\_CycleEngine** (**void**)  
*Cycle the engine. Called once per wakeup.*
- void **CF\_CFDP\_DisableEngine** (**void**)  
*Disables the CFDP engine and resets all state in it.*
- **CFE\_Status\_t CF\_CFDP\_TxFile** (**const char** \*src\_filename, **const char** \*dst\_filename, **CF\_CFDP\_Class\_t** cfdfp\_class, **uint8** keep, **uint8** chan, **uint8** priority, **CF\_EntityId\_t** dest\_id)  
*Begin transmit of a file.*
- **CFE\_Status\_t CF\_CFDP\_PlaybackDir** (**const char** \*src\_filename, **const char** \*dst\_filename, **CF\_CFDP\_Class\_t** cfdfp\_class, **uint8** keep, **uint8** chan, **uint8** priority, **uint16** dest\_id)  
*Begin transmit of a directory.*
- **CF\_Logical\_PduBuffer\_t \* CF\_CFDP\_ConstructPduHeader** (**const CF\_Transaction\_t** \*txn, **CF\_CFDP\_FileDirective\_t** directive\_code, **CF\_EntityId\_t** src\_eid, **CF\_EntityId\_t** dst\_eid, **bool** towards\_sender, **CF\_TransactionSeq\_t** tsn, **bool** silent)  
*Build the PDU header in the output buffer to prepare to send a packet.*
- **CFE\_Status\_t CF\_CFDP\_SendMd** (**CF\_Transaction\_t** \*txn)  
*Build a metadata PDU for transmit.*
- **CFE\_Status\_t CF\_CFDP\_SendFd** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*Send a previously-assembled filedata PDU for transmit.*
- **CFE\_Status\_t CF\_CFDP\_SendEof** (**CF\_Transaction\_t** \*txn)  
*Build an EOF PDU for transmit.*
- **CFE\_Status\_t CF\_CFDP\_SendAck** (**CF\_Transaction\_t** \*txn, **CF\_CFDP\_FileDirective\_t** dir\_code)  
*Build an ACK PDU for transmit.*
- **CFE\_Status\_t CF\_CFDP\_SendFin** (**CF\_Transaction\_t** \*txn)  
*Build a FIN PDU for transmit.*
- void **CF\_CFDP\_SendNak** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*Send a previously-assembled NAK PDU for transmit.*
- **void CF\_CFDP\_AppendTlv** (**CF\_Logical\_TlvList\_t** \*ptlv\_list, **CF\_CFDP\_TlvType\_t** tlv\_type)  
*Appends a single TLV value to the logical PDU data.*
- **CFE\_Status\_t CF\_CFDP\_RecvPh** (**uint8** chan\_num, **CF\_Logical\_PduBuffer\_t** \*ph)  
*Unpack a basic PDU header from a received message.*
- **CFE\_Status\_t CF\_CFDP\_RecvMd** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)

- **`CFE_Status_t CF_CFDP_RecvFd (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Unpack a metadata PDU from a received message.*
- **`CFE_Status_t CF_CFDP_RecvEof (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Unpack a file data PDU from a received message.*
- **`CFE_Status_t CF_CFDP_RecvAck (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Unpack an EOF PDU from a received message.*
- **`CFE_Status_t CF_CFDP_RecvFin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Unpack an ACK PDU from a received message.*
- **`CFE_Status_t CF_CFDP_RecvNak (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Unpack a NAK PDU from a received message.*
- **`void CF_CFDP_DispatchRecv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Dispatch received packet to its handler.*
- **`void CF_CFDP_CancelTransaction (CF_Transaction_t *txn)`**

*Cancels a transaction.*
- **`void CF_CFDP_InitTxnTxFile (CF_Transaction_t *txn, CF_CFDP_Class_t cfdp_class, uint8 keep, uint8 chan, uint8 priority)`**

*Helper function to set tx file state in a transaction.*
- **`CF_Transaction_t * CF_CFDP_StartRxTransaction (uint8 chan_num)`**

*Helper function to start a new RX transaction.*
- **`int CF_CFDP_CopyStringFromLV (char *buf, size_t buf_maxsz, const CF_Logical_Lv_t *src_lv)`**

*Copy string data from a lv (length, value) pair.*
- **`void CF_CFDP_ArmAckTimer (CF_Transaction_t *txn)`**

*Arm the ACK timer.*
- **`void CF_CFDP_RecvDrop (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Receive state function to ignore a packet.*
- **`void CF_CFDP_RecvHold (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Receive state function during holdover period.*
- **`void CF_CFDP_RecvInit (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`**

*Receive state function to process new rx transaction.*
- **`CF_CListTraverse_Status_t CF_CFDP_CloseFiles (CF_CListNode_t *node, void *context)`**

*List traversal function to close all files in all active transactions.*
- **`void CF_CFDP_S_Tick_NewData (CF_Transaction_t *txn)`**

*Tick processor to send new file data.*
- **`void CF_CFDP_TickTransactions (CF_Channel_t *chan)`**

*Call R and then S tick functions for all active transactions.*
- **`void CF_CFDP_ProcessPlaybackDirectory (CF_Channel_t *chan, CF_Playback_t *pb)`**

*Step each active playback directory.*
- **`void CF_CFDP_ProcessPollingDirectories (CF_Channel_t *chan)`**

*Kick the dir playback if timer elapsed.*
- **`CF_CListTraverse_Status_t CF_CFDP_DoTick (CF_CListNode_t *node, void *context)`**

*List traversal function that calls a r or s tick function.*
- **`const char * CF_CFDP_GetMoveTarget (const char *dest_dir, const char *subject_file, char *dest_buf, size_t dest_size)`**

*Get move target.*
- **`void CF_CFDP_GetTempName (const CF_History_t *hist, char *FileNameBuf, size_t FileNameSize)`**

*Get temporary file name.*

- void [CF\\_CFDP\\_ReceivePdu](#) ([CF\\_Channel\\_t](#) \*chan, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph)  
*Receive PDU processing entry point.*
- void [CF\\_CFDP\\_SetupRxTransaction](#) ([CF\\_Transaction\\_t](#) \*txn, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph)  
*Sets up a new RX transaction based on PDU.*
- void [CF\\_CFDP\\_SetupTxTransaction](#) ([CF\\_Transaction\\_t](#) \*txn)  
*Sets up a new TX transaction.*
- bool [CF\\_CFDP\\_StartFirstPending](#) ([CF\\_Channel\\_t](#) \*chan)  
*Pulls next TX transaction from the PEND queue.*
- void [CF\\_CFDP\\_AllocChunkList](#) ([CF\\_Transaction\\_t](#) \*txn)  
*Allocates a chunk list for the transaction.*
- void [CF\\_CFDP\\_ArmInactTimer](#) ([CF\\_Transaction\\_t](#) \*txn)  
*Arms the inactivity timer.*

### 12.35.1 Detailed Description

The CF Application CFDP engine and packet parsing header file

### 12.35.2 Typedef Documentation

**12.35.2.1 [CF\\_CFDP\\_Tick\\_args\\_t](#)** [typedef struct CF\\_CFDP\\_Tick\\_args CF\\_CFDP\\_Tick\\_args\\_t](#)  
Structure for use with the [CF\\_CFDP\\_DoTick\(\)](#) functions.

### 12.35.3 Function Documentation

**12.35.3.1 [CF\\_CFDP\\_AllocChunkList\(\)](#)** [void CF\\_CFDP\\_AllocChunkList \( CF\\_Transaction\\_t \\* txn \)](#)

Allocates a chunk list for the transaction.

All transactions use a chunklist to track gaps in the file as it is sent or received. This is used for NAK generation and processing in class 2 or just simply checking if the whole file is received in class 1.

The chunk lists come from a pool and are re-used

Only active transactions need a chunklist. Pending TX transactions will not get one assigned until they become active.

#### Parameters

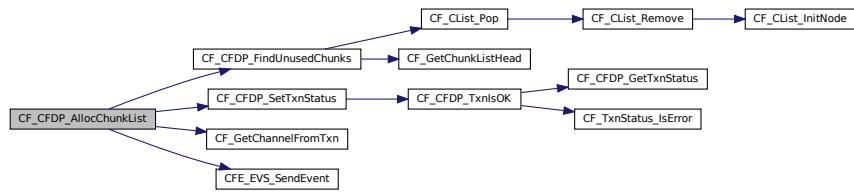
<a href="#">txn</a>	Transaction pointer
---------------------	---------------------

Definition at line 978 of file cf\_cfdp.c.

References [CF\\_CFDP\\_FindUnusedChunks\(\)](#), [CF\\_CFDP\\_NO\\_CHUNKLIST\\_AVAIL\\_EID](#), [CF\\_CFDP\\_SetTxnStatus\(\)](#), [CF\\_GetChannelFromTxn\(\)](#), [CF\\_TxnStatus\\_NO\\_RESOURCE](#), [CFE\\_EVS\\_EventType\\_ERROR](#), [CFE\\_EVS\\_SendEvent\(\)](#), [CF\\_Transaction::chunks](#), [CF\\_History::dir](#), [CF\\_Transaction::history](#), and [CF\\_History::seq\\_num](#).

Referenced by [CF\\_CFDP\\_SetupRxTransaction\(\)](#), and [CF\\_CFDP\\_SetupTxTransaction\(\)](#).

Here is the call graph for this function:



**12.35.3.2 CF\_CFDP\_AppendTlv()** `void CF_CFDP_AppendTlv ( CF_Logical_TlvList_t * ptlv_list, CF_CFDP_TlvType_t tlv_type )`

Appends a single TLV value to the logical PDU data.

This function implements common functionality between SendEof and SendFin which append a TLV value specifying the faulting entity ID.

#### Assumptions, External Events, and Notes:

`ptlv_list` must not be NULL. Only `CF_CFDP_TLV_TYPE_ENTITY_ID` type is currently implemented

#### Parameters

<code>ptlv_list</code>	TLV list from current PDU buffer.
<code>tlv_type</code>	Type of TLV to append. Currently must be <code>CF_CFDP_TLV_TYPE_ENTITY_ID</code> .

Definition at line 433 of file cf\_cfdp.c.

References `CF_AppData`, `CF_CFDP_GetValueEncodedSize()`, `CF_CFDP_TLV_TYPE_ENTITY_ID`, `CF_PDU_MAX_TLV`, `CF_AppData_t::config_table`, `CF_Logical_Tlv::data`, `CF_Logical_TlvData::data_ptr`, `CF_Logical_TlvData::eid`, `CF_Logical_Tlv::length`, `CF_ConfigTable::local_eid`, `CF_Logical_TlvList::num_tlv`, `CF_Logical_TlvList::tlv`, and `CF_Logical_Tlv::type`.

Referenced by `CF_CFDP_SendEof()`, and `CF_CFDP_SendFin()`.

Here is the call graph for this function:



**12.35.3.3 CF\_CFDP\_ArmAckTimer()** `void CF_CFDP_ArmAckTimer ( CF_Transaction_t * txn )`

Arm the ACK timer.

**Description**

Helper function to arm the ACK timer and set the flag.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

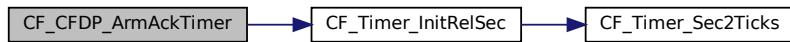
<i>txn</i>	Pointer to the transaction state
------------	----------------------------------

Definition at line 123 of file cf\_cfdp.c.

References CF\_Transaction::ack\_timer, CF\_Flags\_Common::ack\_timer\_armed, CF\_ChannelConfig::ack\_timer\_s, CF\_AppData, CF\_Timer\_InitRelSec(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_StateFlags::com, CF\_AppData\_t::config\_table, and CF\_Transaction::flags.

Referenced by CF\_CFDP\_R2\_Recv(), CF\_CFDP\_R\_Tick\_Maintenance(), CF\_CFDP\_S\_Tick\_Maintenance(), and CF\_CFDP\_SendNak().

Here is the call graph for this function:

**12.35.3.4 CF\_CFDP\_ArmInactTimer()** `void CF_CFDP_ArmInactTimer (`  
`CF_Transaction_t * txn )`

Arms the inactivity timer.

This is invoked whenever activity occurs on a transaction. The inactivity timer will be reset to the value in the configuration.

**Parameters**

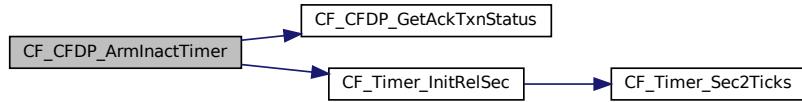
<i>txn</i>	Transaction pointer
------------	---------------------

Definition at line 157 of file cf\_cfdp.c.

References CF\_ChannelConfig::ack\_timer\_s, CF\_AppData, CF\_CFDP\_AckTxnStatus\_ACTIVE, CF\_CFDP\_GetAckTxnStatus(), CF\_Timer\_InitRelSec(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_AppData\_t::config\_table, CF\_Transaction::inactivity\_timer, and CF\_ChannelConfig::inactivity\_timer\_s.

Referenced by CF\_CFDP\_DispatchRecv(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_SetupRxTransaction(), and CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



**12.35.3.5 CF\_CFDP\_CancelTransaction()** void CF\_CFDP\_CancelTransaction ( CF\_Transaction\_t \* *txn* )

Cancels a transaction.

#### Assumptions, External Events, and Notes:

*txn* must not be NULL.

#### Parameters

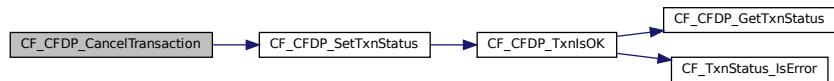
<i>txn</i>	Pointer to the transaction state
------------	----------------------------------

Definition at line 2116 of file cf\_cfdp.c.

References CF\_Flags\_Common::canceled, CF\_CFDP\_SetTxnStatus(), CF\_TxnStatus\_CANCEL\_REQUEST\_← RECEIVED, CF\_StateFlags::com, and CF\_Transaction::flags.

Referenced by CF\_Cancel\_TxnCmd().

Here is the call graph for this function:



**12.35.3.6 CF\_CFDP\_CheckAckNakCount()** bool CF\_CFDP\_CheckAckNakCount ( CF\_Transaction\_t \* *txn*, uint8 \* *counter* )

Increment ack/nak counter and check limit.

#### Description

Checks the counter against the configured limit. If still under limit, increment counter and return true. If reached limit, generate event and return false.

#### Assumptions, External Events, and Notes:

**Parameters**

<i>txn</i>	the transaction object
<i>counter</i>	pointer to the respective ack/nak counter

**Return values**

<i>true</i>	Within limit, another ACK/NAK is allowed
<i>false</i>	Reached limit, another ACK/NAK is not allowed

Definition at line 188 of file cf\_cfdp.c.

References CF\_HkFault::ack\_limit, CF\_ChannelConfig::ack\_limit, CF\_AppData, CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_← EID, CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID, CF\_Direction\_TX, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_AppData\_t::config\_table, CF\_HkChannel\_Data::counters, CF\_History::dir, CF\_HkCounters::fault, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_History::peer\_eid, and CF\_History::seq\_num.

Referenced by CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_← SubstateRecvEof(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), and CF\_CFDP\_S\_SubstateRecvFin().

Here is the call graph for this function:



**12.35.3.7 CF\_CFDP\_CloseFiles()** *CF\_CListTraverse\_Status\_t* CF\_CFDP\_CloseFiles (   
   *CF\_CListNode\_t* \* *node*,  
   *void* \* *context* )

List traversal function to close all files in all active transactions.

This helper is used in conjunction with [CF\\_CList\\_Traverse\(\)](#).

**Assumptions, External Events, and Notes:**

*node* must not be NULL.

**Parameters**

<i>node</i>	List node pointer
<i>context</i>	Opaque pointer, not used in this function

**Returns**

integer traversal code

### Return values

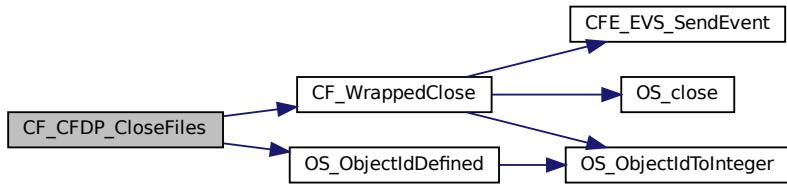
Always	CF_LIST_CONT indicate list traversal should not exit early.
--------	---

Definition at line 2132 of file cf\_cfdp.c.

References CF\_CLIST\_CONT, CF\_WrappedClose(), container\_of, CF\_Transaction::fd, and OS\_ObjectIdDefined().

Referenced by CF\_CFDP\_DisableEngine().

Here is the call graph for this function:



### 12.35.3.8 CF\_CFDP\_CompleteTick()

```
void CF_CFDP_CompleteTick(
```

```
    CF_Transaction_t * txn )
```

Complete tick processing on a transaction.

#### Description

Checks if transmit operations have been blocked due to TX limits If so, snapshot this TXN as a resume point for next cycle. If not, does nothing.

#### Assumptions, External Events, and Notes:

#### Parameters

<code>txn</code>	the transaction object
------------------	------------------------

Definition at line 1357 of file cf\_cfdp.c.

References CF\_GetChannelFromTxn(), CF\_Channel::tick\_resume, and CF\_Channel::tx\_blocked.

Referenced by CF\_CFDP\_R\_Tick(), and CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



```
12.35.3.9 CF_CFDP_ConstructPduHeader() CF_Logical_PduBuffer_t* CF_CFDP_ConstructPduHeader (
    const CF_Transaction_t * txn,
    CF_CFDP_FileDirective_t directive_code,
    CF_EntityId_t src_eid,
    CF_EntityId_t dst_eid,
    bool towards_sender,
    CF_TransactionSeq_t tsn,
    bool silent )
```

Build the PDU header in the output buffer to prepare to send a packet.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction object
<i>directive_code</i>	Code to use for file directive headers (set to 0 for data)
<i>src_eid</i>	Value to set in source entity ID field
<i>dst_eid</i>	Value to set in destination entity ID field
<i>towards_sender</i>	Whether this is transmitting toward the sender entity
<i>tsn</i>	Transaction sequence number to put into PDU
<i>silent</i>	If true, suppress error event if no message buffer available

#### Returns

Pointer to PDU buffer which may be filled with additional data

#### Return values

<code>NULL</code>	if no message buffer available
-------------------	--------------------------------

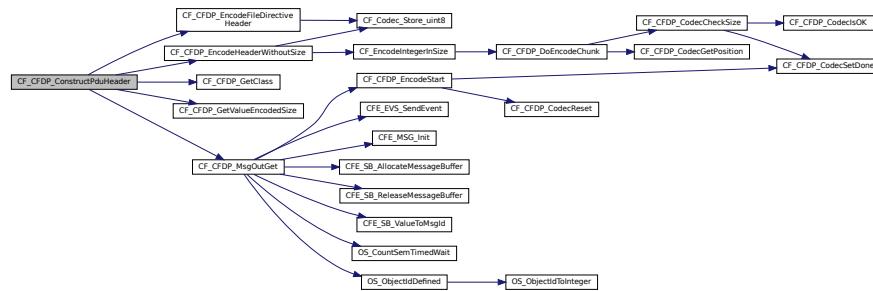
Definition at line 296 of file cf\_cfdp.c.

References CF\_CFDP\_CLASS\_1, CF\_CFDP\_EncodeFileDirectiveHeader(), CF\_CFDP\_EncodeHeaderWithoutSize(), CF\_CFDP\_GetClass(), CF\_CFDP\_GetValueEncodedSize(), CF\_CFDP\_MsgOutGet(), CF\_Logical\_PduHeader::destination\_eid, CF\_Logical\_PduHeader::direction, CF\_Logical\_PduFileDirectiveHeader::directive\_code, CF\_Logical\_PduHeader::eid\_length, CF\_Logical\_PduBuffer::fdirective, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::pdu\_type, CF\_Logical\_PduBuffer::penc, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical

\_PduHeader::source\_eid, CF\_Logical\_PduHeader::txm\_mode, CF\_Logical\_PduHeader::txn\_seq\_length, and CF\_Logical\_PduHeader::version.

Referenced by CF\_CFDP\_R\_SendNak(), CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFin(), and CF\_CFDP\_SendMd().

Here is the call graph for this function:



### 12.35.3.10 CF\_CFDP\_CopyStringFromLV()

```
int CF_CFDP_CopyStringFromLV (
    char * buf,
    size_t buf_maxsz,
    const CF_Logical_Lv_t * src_lv )
```

Copy string data from a lv (length, value) pair.

This copies a string value from an LV pair inside a PDU buffer. In CF this is used for file names embedded within PDUs.

#### Note

This function assures that the output string is terminated appropriately, such that it can be used as a normal C string. As such, the buffer size must be at least 1 byte larger than the maximum string length.

#### Assumptions, External Events, and Notes:

src\_lv must not be NULL. buf must not be NULL.

#### Parameters

<i>buf</i>	Pointer to buffer to store string
<i>buf_maxsz</i>	Total size of buffer pointer to by buf (usable size is 1 byte less, for termination)
<i>src_lv</i>	Pointer to LV pair from logical PDU buffer

#### Returns

The resulting string length, NOT including termination character

#### Return values

<i>CF_ERROR</i>	on error
-----------------	----------

Definition at line 2096 of file cf\_cfdp.c.

References CF\_ERROR, CF\_Logical\_Lv::data\_ptr, and CF\_Logical\_Lv::length.  
Referenced by CF\_CFDP\_RecvMd().

#### 12.35.3.11 CF\_CFDP\_CycleEngine()

```
void CF_CFDP_CycleEngine (
    void )
```

Cycle the engine. Called once per wakeup.

**Assumptions, External Events, and Notes:**

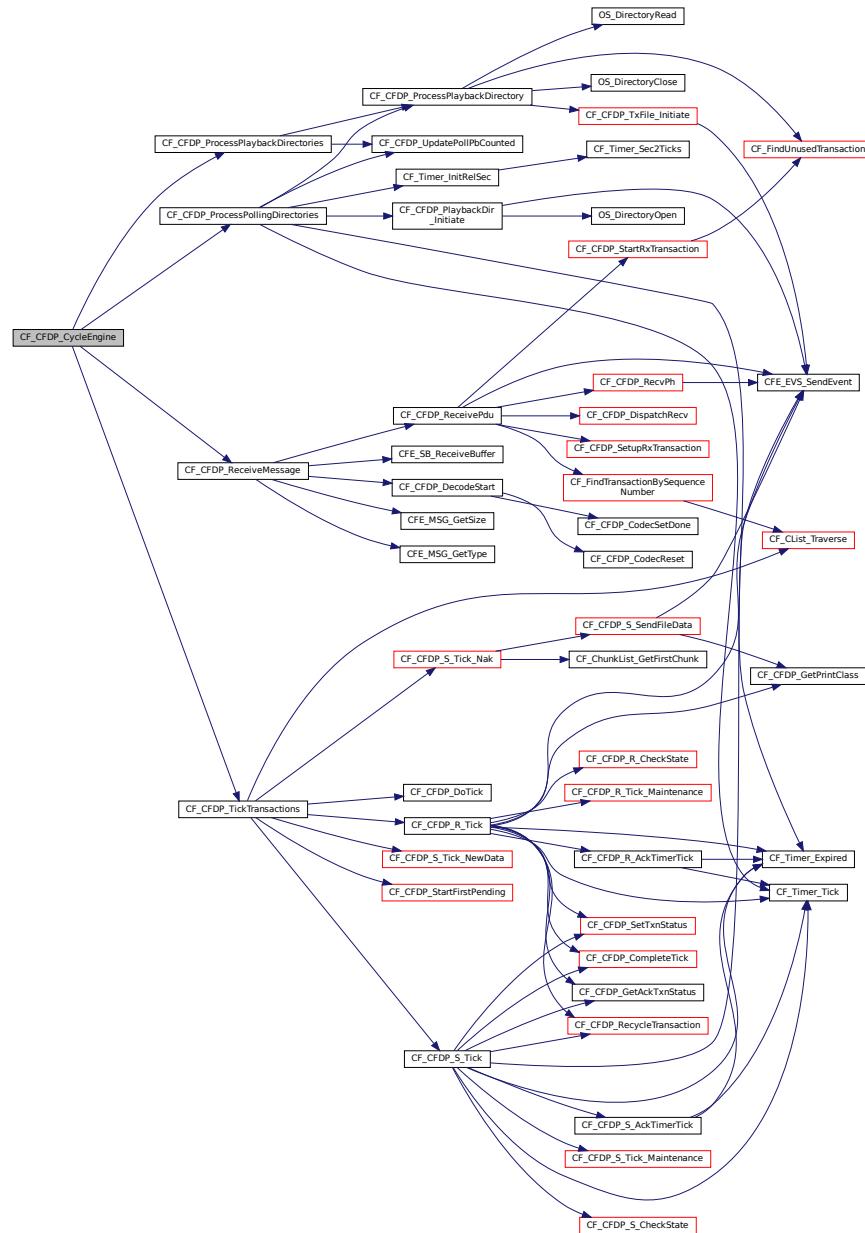
None

Definition at line 1862 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_ProcessPlaybackDirectories(), CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_ReceiveMessage(), CF\_CFDP\_TickTransactions(), CF\_NUM\_CHANNELS, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_HkChannel\_Data::frozen, CF\_AppData\_t::hk, CF\_Channel::outgoing\_counter, CF\_HkPacket::Payload, and CF\_Channel::tx\_blocked.

Referenced by CF\_WakeupCmd().

Here is the call graph for this function:



**12.35.3.12 CF\_CFDP\_DecodeStart()** void CF\_CFDP\_DecodeStart (

```

CF_DecoderState_t * pdec,
const void * msgbuf,
CF_Logical_PduBuffer_t * ph,
size_t encaps_hdr_size,
size_t total_size )
  
```

Initiate the process of decoding a received PDU.

This resets the decoder and PDU buffer to initial values, and prepares for decoding a new PDU that was received from a remote entity.

#### Parameters

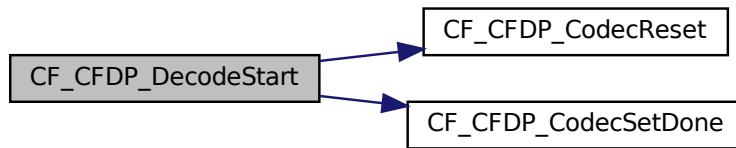
<i>pdec</i>	Decoder state structure, will be reset-initialized by this call to point to <i>msgbuf</i> .
<i>msgbuf</i>	Pointer to encapsulation message, in this case a CFE software bus message
<i>ph</i>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<i>encap_hdr_size</i>	Offset of first CFDP PDU octet within buffer
<i>total_size</i>	Total size of <i>msgbuf</i> encapsulation structure (decoding cannot exceed this)

Definition at line 89 of file cf\_cfdp.c.

References `CF_DecoderState::base`, `CF_CFDP_CodecReset()`, `CF_CFDP_CodecSetDone()`, `CF_DecoderState::codec_state`, `CF_CodecState::max_size`, and `CF_Logical_PduBuffer::pdec`.

Referenced by `CF_CFDP_ReceiveMessage()`.

Here is the call graph for this function:



**12.35.3.13 CF\_CFDP\_DisableEngine()** `void CF_CFDP_DisableEngine (`  
 `void )`

Disables the CFDP engine and resets all state in it.

## Assumptions, External Events, and Notes:

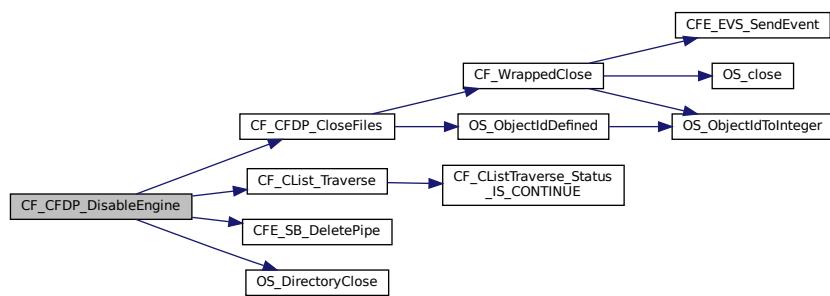
None

Definition at line 2148 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_CloseFiles(), CF\_CList\_Traverse(), CF\_MAX\_COMMANDED←\_PLAYBACK\_DIRECTORIES\_PER\_CHAN, CF\_MAX\_POLLING\_DIR\_PER\_CHAN, CF\_NUM\_CHANNELS, CF←\_Queueldx\_RX, CF\_Queueldx\_TX, CFE\_SB\_DeletePipe(), CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Playback::dir\_id, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_AppData\_t::hk, OS\_DirectoryClose(), CF\_Hk←Packet::Payload, CF\_Poll::pb, CF\_Channel::pipe, CF\_Channel::playback, CF\_Channel::poll, CF\_HkChannel\_Data::q←\_size, and CF\_Channel::qs.

Referenced by CF\_DisableEngineCmd().

Here is the call graph for this function:



**12.35.3.14 CF\_CFDP\_DispatchRecv()** void CF\_CFDP\_DispatchRecv ( CF\_Transaction\_t \* *txn*, CF\_Logical\_PduBuffer\_t \* *ph* )

Dispatch received packet to its handler.

This dispatches the PDU to the appropriate handler based on the transaction state

## Assumptions, External Events, and Notes:

*txn* must not be null. It must be an initialized transaction.

## Parameters

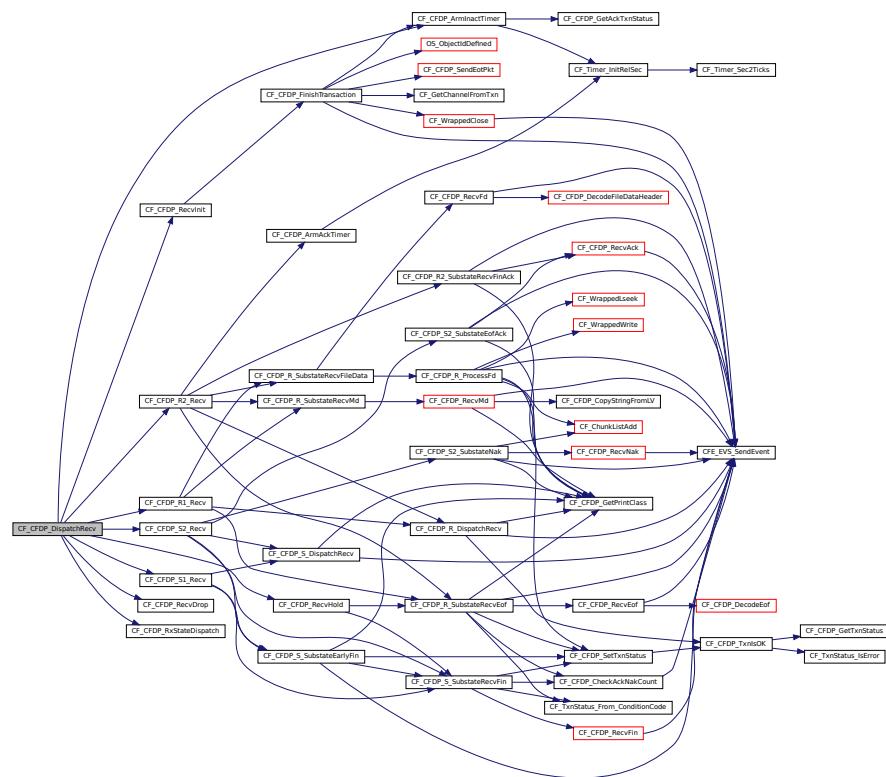
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 228 of file cf\_cfdp.c.

References CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_R1\_Recv(), CF\_CFDP\_R2\_Recv(), CF\_CFDP\_RecvDrop(), CF←\_CFDP\_RecvHold(), CF\_CFDP\_RecvInit(), CF\_CFDP\_RxStateDispatch(), CF\_CFDP\_S1\_Recv(), CF\_CFDP\_S2←Recv(), CF\_TxnState\_DROP, CF\_TxnState\_HOLD, CF\_TxnState\_INIT, CF\_TxnState\_R1, CF\_TxnState\_R2, CF\_Txn←State\_S1, CF\_TxnState\_S2, and CF\_CFDP\_TxnRecvDispatchTable\_t::rx.

Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



**12.35.3.15 CF\_CFDP\_DoTick()** CF\_CListTraverse\_Status\_t CF\_CFDP\_DoTick (

```
CF_CListNode_t * node,  
void * context )
```

List traversal function that calls a r or s tick function.

This helper is used in conjunction with [CF\\_CList\\_Traverse\(\)](#).

## **Assumptions, External Events, and Notes:**

node must not be NULL, context must not be NULL.

## Parameters

<i>node</i>	Pointer to list node
<i>context</i>	Pointer to CF_CFDP_Tick_args_t object (passed through)

## Returns

## integer traversal code

## Return values

*CF CLIST EXIT* when it's found, which terminates list traversal

## Return values

<code>CF_CLIST_CONT</code>	when it's isn't found, which causes list traversal to continue
----------------------------	--

Definition at line 1324 of file cf\_cfdp.c.

References `CF_CLIST_CONT`, `CF_CLIST_EXIT`, `CF_CFDP_Tick_args::chan`, `CF_StateFlags::com`, `container_of`, `CF_Transaction::flags`, `CF_CFDP_Tick_args::fn`, `CF_CFDP_Tick_args::resume_point`, `CF_Flags_Common::suspended`, and `CF_Channel::tx_blocked`.

Referenced by `CF_CFDP_TickTransactions()`.

#### 12.35.3.16 CF\_CFDP\_EncodeStart() void CF\_CFDP\_EncodeStart (

```
    CF_EncoderState_t * penc,
    void * msgbuf,
    CF_Logical_PduBuffer_t * ph,
    size_t encaps_hdr_size,
    size_t total_size )
```

Initiate the process of encoding a new PDU to send.

This resets the encoder and PDU buffer to initial values, and prepares for encoding a new PDU for sending to a remote entity.

## Parameters

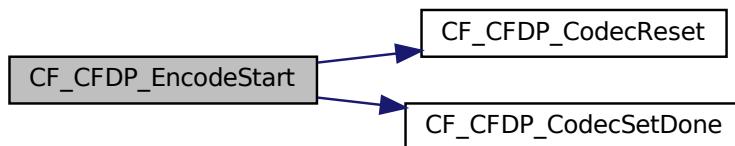
<code>penc</code>	Encoder state structure, will be reset-initialized by this call to point to <code>msgbuf</code> .
<code>msgbuf</code>	Pointer to encapsulation message, in this case a CFE software bus message
<code>ph</code>	Pointer to logical PDU buffer content, will be cleared to all zero by this call
<code>encap_hdr_size</code>	Offset of first CFDP PDU octet within buffer
<code>total_size</code>	Allocated size of <code>msgbuf</code> encapsulation structure (encoding cannot exceed this)

Definition at line 55 of file cf\_cfdp.c.

References `CF_EncoderState::base`, `CF_CFDP_CodecReset()`, `CF_CFDP_CodecSetDone()`, `CF_EncoderState::codec_state`, `CF_CodecState::max_size`, and `CF_Logical_PduBuffer::penc`.

Referenced by `CF_CFDP_MsgOutGet()`.

Here is the call graph for this function:



#### 12.35.3.17 CF\_CFDP\_FinishTransaction() void CF\_CFDP\_FinishTransaction (

```
    CF_Transaction_t * txn,
    bool keep_history )
```

Finish a transaction.

This marks the transaction as completed and puts it into a holdover state. After the inactivity timer expires, the resources will be recycled and become available for re-use.

Holdover is necessary because even though locally we consider the transaction to be complete, there may be undelivered PDUs still in network queues that get delivered to us late. By holding this transaction for a bit longer, we can still associate those PDUs with this transaction/seq\_num and appropriately handle them as dupes/spurious deliveries.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

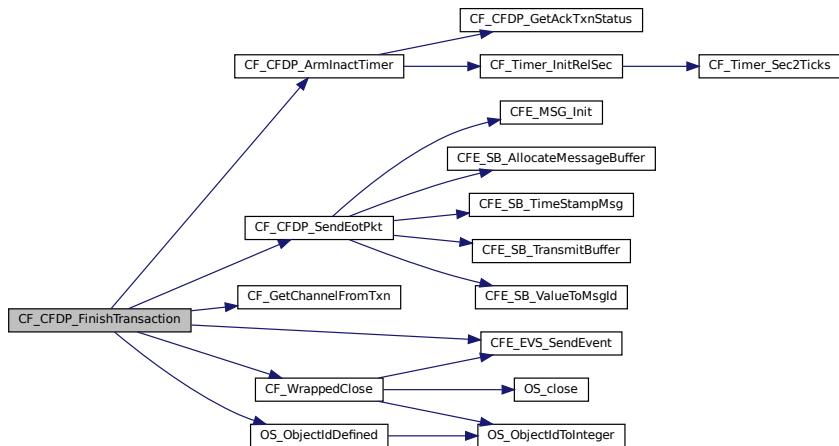
<i>txn</i>	Pointer to the transaction object
<i>keep_history</i>	Whether the transaction info should be preserved in history

Definition at line 1897 of file cf\_cfdp.c.

References CF\_ASSERT, CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_SendEotPkt(), CF\_Direction\_TX, CF\_GetChannelFromTxn(), CF\_QueueIdx\_FREE, CF\_RESET\_FREED\_XACT\_DBG\_EID, CF\_TRACE, CF\_TxnState\_HOLD, CF\_WrappedClose(), CFE\_EVS\_EventType\_DEBUG, CFE\_EVS\_SendEvent(), CF\_Flags\_Tx::cmd\_tx, CF\_StateFlags::com, CF\_History::dir, CF\_Transaction::fd, CF\_Transaction::flags, CF\_Transaction::history, CF\_Flags\_Common::keep\_history, CF\_Channel::num\_cmd\_tx, CF\_Playback::num\_ts, OS\_OBJECT\_ID\_UNDEFINED, OS\_ObjectIdDefined(), CF\_Transaction::pb, CF\_Flags\_Common::q\_index, CF\_History::seq\_num, CF\_Transaction::state, and CF\_StateFlags::tx.

Referenced by CF\_Abandon\_TxnCmd(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_RecvInit(), CF\_CFDP\_S\_CheckState(), and CF\_PurgeTransaction().

Here is the call graph for this function:



```

12.35.3.18 CF_CFDP_GetMoveTarget() const char* CF_CFDP_GetMoveTarget (
    const char * dest_dir,
    const char * subject_file,
    char * dest_buf,
    size_t dest_size )
  
```

Get move target.

Gets the full path of the destination file after move to dest\_dir

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>dest_dir</i>	Directory to move file into
<i>subject_file</i>	Full path of file to move
<i>dest_buf</i>	Buffer to store result
<i>dest_size</i>	Size of result buffer

**Return values**

<i>NULL</i>	if the result is not valid (i.e. dest_dir not set)
<i>dest_buf</i>	if result is valid

Definition at line 2209 of file cf\_cfdp.c.

References CF\_EID\_INF\_CFDP\_BUF\_EXCEED, CF\_FILENAME\_TRUNCATED, CFE\_EVS\_EventType\_INFORMATION, and CFE\_EVS\_SendEvent().

Referenced by CF\_CFDP\_S\_HandleFileRetention().

Here is the call graph for this function:



**12.35.3.19 CF\_CFDP\_GetPrintClass()** static int CF\_CFDP\_GetPrintClass ( const CF\_Transaction\_t \* txn ) [inline], [static]

Get printable CFDP class number.

This is intended for use in log messages where a printf style spec string is used with a d modifier.

**Returns**

integer 1 for class 1

integer 2 for class 2

Definition at line 51 of file cf\_cfdp.h.

References CF\_Transaction::reliable\_mode.

Referenced by CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SendNak(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF↔\_CFDP\_RecvMd(), CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(),

CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SendFileData(), and CF\_CFDP\_S\_SubstateEarlyFin().

**12.35.3.20 CF\_CFDP\_GetTempName()** void CF\_CFDP\_GetTempName (

```
    const CF_History_t * hist,
    char * FileNameBuf,
    size_t FileNameSize )
```

Get temporary file name.

Gets the full path of the temporary file for a transaction

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>hist</i>	History pointer
<i>FileNameBuf</i>	Buffer to store result
<i>FileNameSize</i>	Size of result buffer

Definition at line 2197 of file cf\_cfdp.c.

References CF\_AppData, CF\_FILENAME\_MAX\_PATH, CF\_AppData\_t::config\_table, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_ConfigTable::tmp\_dir.

Referenced by CF\_CFDP\_R\_HandleFileRetention(), and CF\_CFDP\_R\_Init().

**12.35.3.21 CF\_CFDP\_GetTxnStatus()** CF\_TxnStatus\_t CF\_CFDP\_GetTxnStatus (

```
    const CF_Transaction_t * txn )
```

Helper function to retrieve transaction status code only.

This retrieves the status from the history block.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

**Returns**

Status Code value set within transaction

Definition at line 2028 of file cf\_cfdp.c.

References CF\_TxnStatus\_NO\_RESOURCE, CF\_Transaction::history, and CF\_History::txn\_stat.  
Referenced by CF\_CFDP\_TxnIsOK().

**12.35.3.22 CF\_CFDP\_InitEngine()** CFE\_Status\_t CF\_CFDP\_InitEngine (

```
    void )
```

Initialization function for the CFDP engine.

## Description

Performs all initialization of the CFDP engine

### Assumptions, External Events, and Notes:

Only called once.

## Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
--------------------------	--

## Returns

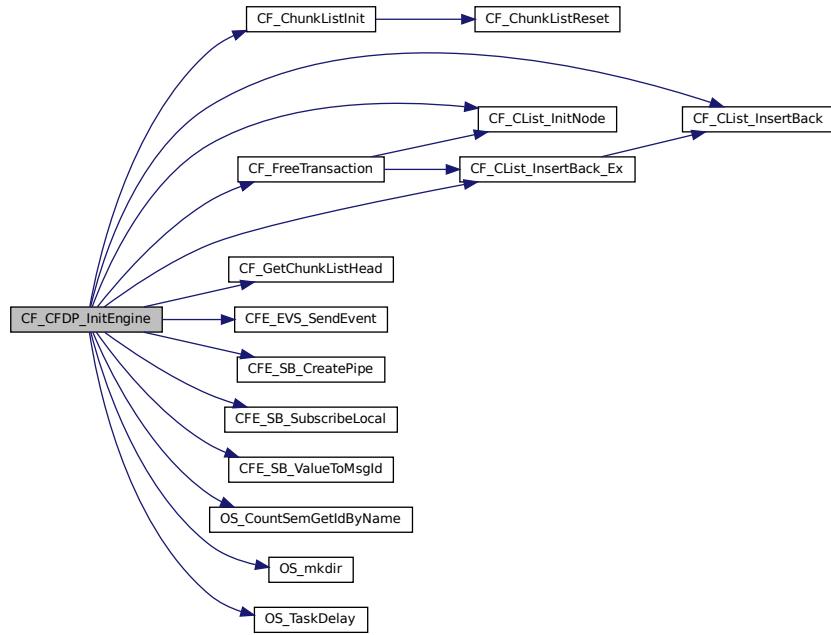
anything else on error.

Definition at line 1135 of file cf\_cfdp.c.

References CF\_AppData, CF\_Assert, CF\_CHANNEL\_NUM\_RX\_CHUNKS\_PER\_TRANSACTION, CF\_CHANNEL\_NUM\_TX\_CHUNKS\_PER\_TRANSACTION, CF\_CHANNEL\_PIPE\_PREFIX, CF\_ChunkListInit(), CF\_CList\_InitNode(), CF\_CList\_InsertBack(), CF\_CList\_InsertBack\_Ex(), CF\_CR\_CHANNEL\_PIPE\_ERR\_EID, CF\_Direction\_NUM, CF\_FreeTransaction(), CF\_GetChunkListHead(), CF\_INIT\_SEM\_ERR\_EID, CF\_INIT\_SUB\_ERR\_EID, CF\_NUM\_CHANNELS, CF\_NUM\_CHUNKS\_ALL\_CHANNELS, CF\_NUM\_HISTORIES\_PER\_CHANNEL, CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL, CF\_QueueIdx\_HIST\_FREE, CF\_STARTUP\_SEM\_MAX\_RETRIES, CF\_STARTUP\_SEM\_TASK\_DELAY, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SB\_CreatePipe(), CFE\_SB\_SubscribeLocal(), CFE\_SB\_ValueToMsgId(), CFE\_SUCCESS, CF\_ConfigTable::chan, CF\_Engine::channels, CF\_Engine::chunk\_mem, CF\_ChunkWrapper::chunks, CF\_Engine::chunks, CF\_History::cl\_node, CF\_ChunkWrapper::cl\_node, CF\_AppData\_t::config\_table, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_Engine::histories, CF\_ChannelConfig::mid\_input, OS\_CountSemGetIdByName(), OS\_ERR\_NAME\_NOT\_FOUND, OS\_mkdir(), OS\_SUCCESS, OS\_TaskDelay(), CF\_Channel::pipe, CF\_ChannelConfig::pipe\_depth\_input, CF\_Channel::sem\_id, CF\_ChannelConfig::sem\_name, CF\_ConfigTable::tmp\_dir, and CF\_Engine::transactions.

Referenced by CF\_AppInit(), and CF\_EnableEngineCmd().

Here is the call graph for this function:



**12.35.3.23 CF\_CFDP\_InitTxnTxFile()** void CF\_CFDP\_InitTxnTxFile (

```

    CF_Transaction_t * txn,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority )
  
```

Helper function to set tx file state in a transaction.

This sets various fields inside a newly-allocated transaction structure appropriately for sending a file.

#### Assumptions, External Events, and Notes:

`txn` must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction state
<code>cfdp_class</code>	Set to class 1 or class 2
<code>keep</code>	Whether to keep the local file
<code>chan</code>	CF channel number
<code>priority</code>	Priority of transfer

Definition at line 1486 of file cf\_cfdp.c.

References `CF_TxnState_S1`, `CF_TxnState_S2`, `CF_Transaction::chan_num`, `CF_Transaction::keep`, `CF_Transaction::priority`, `CF_Transaction::reliable_mode`, and `CF_Transaction::state`.

Referenced by CF\_CFDP\_TxFile\_Initiate().

```
12.35.3.24 CF_CFDP_PlaybackDir() CFE_Status_t CF_CFDP_PlaybackDir (
    const char * src_filename,
    const char * dst_filename,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority,
    uint16 dest_id )
```

Begin transmit of a directory.

#### Description

This function sets up CF\_Playback\_t structure with state so it can become part of the directory polling done at each engine cycle.

#### Assumptions, External Events, and Notes:

src\_filename must not be NULL. dst\_filename must not be NULL.

#### Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename
<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

#### Return values

<i>CFE_SUCCESS</i>	Successful execution. Operation was performed successfully
--------------------	--

**Returns**

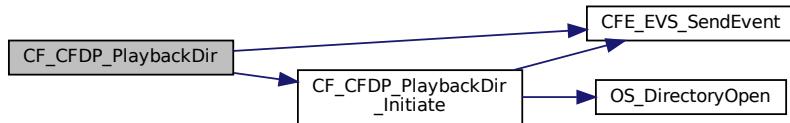
CFE\_SUCCESS on success. CF\_ERROR on error.

Definition at line 1647 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_DIR\_SLOT\_ERR\_EID, CF\_CFDP\_PlaybackDir\_Initiate(), CF\_ERROR, CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN, CFE\_ES\_PerfLogEntry, CFE\_ES\_EventType\_ERROR, CFE\_ES\_SendEvent(), CF\_Engine::channels, CF\_AppData\_t::engine, and CF\_Channel::playback.

Referenced by CF\_PlaybackDirCmd().

Here is the call graph for this function:



**12.35.3.25 CF\_CFDP\_ProcessPlaybackDirectory()** void CF\_CFDP\_ProcessPlaybackDirectory (   
`CF_Channel_t * chan,`  
`CF_Playback_t * pb )`

Step each active playback directory.

**Description**

Check if a playback directory needs iterated, and if so does, and if a valid file is found initiates playback on it.

**Assumptions, External Events, and Notes:**

chan must not be NULL, pb must not be NULL.

**Parameters**

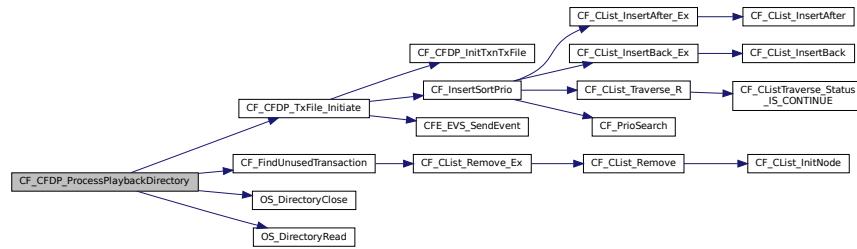
<i>chan</i>	The channel associated with the playback
<i>pb</i>	The playback state

Definition at line 1677 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_TxFile\_Initiate(), CF\_Direction\_TX, CF\_FILENAME\_MAX\_NAME, CF\_FILENAME\_MAX\_PATH, CF\_FindUnusedTransaction(), CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK, CF\_PERF\_ID\_DIRREAD, CF\_Playback::cfdp\_class, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CF\_Engine::channels, CF\_Playback::dest\_id, CF\_Playback::dir\_id, CF\_Playback::diropen, CF\_TxnFilenames::dst\_filename, CF\_AppData\_t::engine, os\_dirent\_t::FileName, CF\_History::fnames, CF\_Playback::fnames, CF\_Transaction::history, CF\_Playback::keep, CF\_Playback::num\_ts, OS\_DirectoryClose(), OS\_DirectoryRead(), OS\_DIRENTRY\_NAME, OS\_SUCCESS, CF\_Transaction::pb, CF\_Playback::pending\_file, CF\_Playback::priority, and CF\_TxnFilenames::src\_filename.

Referenced by CF\_CFDP\_ProcessPlaybackDirectories(), and CF\_CFDP\_ProcessPollingDirectories().

Here is the call graph for this function:



**12.35.3.26 CF\_CFDP\_ProcessPollingDirectories()** void CF\_CFDP\_ProcessPollingDirectories ( CF\_Channel\_t \* chan )

Kick the dir playback if timer elapsed.

#### Description

This function waits for the polling directory interval timer, and if it has expired, starts a playback in the polling directory.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

#### Parameters

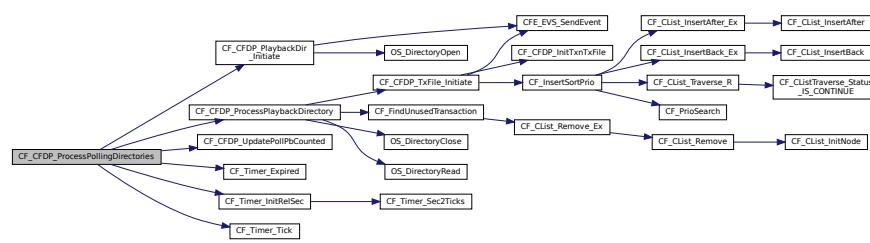
chan	The channel associated with the playback
------	--

Definition at line 1793 of file cf\_cfdp.c.

References CF\_Playback::busy, CF\_AppData, CF\_CFDP\_PlaybackDir\_Initiate(), CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_UpdatePollPbCounted(), CF\_MAX\_POLLING\_DIR\_PER\_CHAN, CF\_Timer\_Expired(), CF\_Timer\_InitRelSec(), CF\_Timer\_Tick(), CF\_PollDir::cfdp\_class, CF\_ConfigTable::chan, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_AppData\_t::config\_table, CF\_PollDir::dest\_eid, CF\_PollDir::dst\_dir, CF\_PollDir::enabled, CF\_AppData\_t::engine, CF\_AppData\_t::hk, CF\_PollDir::interval\_sec, CF\_Poll::interval\_timer, CF\_Playback::num\_ts, CF\_HkPacket::Payload, CF\_Poll::pb, CF\_Channel::poll, CF\_HkChannel\_Data::poll\_counter, CF\_ChannelConfig::polldir, CF\_PollDir::priority, CF\_PollDir::src\_dir, and CF\_Poll::timer\_set.

Referenced by CF\_CFDP\_CycleEngine().

Here is the call graph for this function:



```
12.35.3.27 CF_CFDP_ReceivePdu() void CF_CFDP_ReceivePdu (
    CF_Channel_t * chan,
    CF_Logical_PduBuffer_t * ph )
```

Receive PDU processing entry point.

Invoked from the transport interface (e.g. software bus or equivalent) after reception of a PDU from the network.

**Assumptions, External Events, and Notes:**

None

**Parameters**

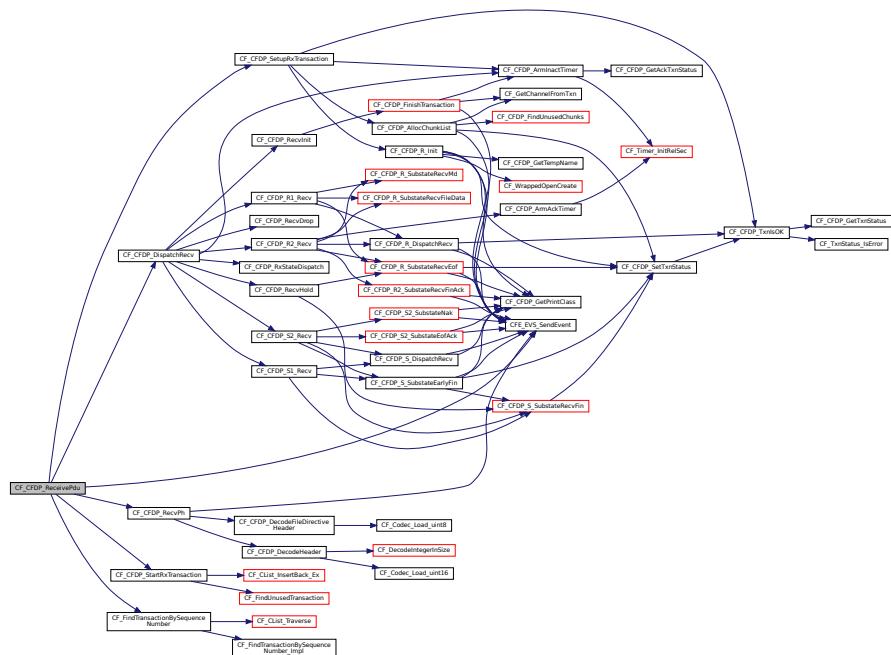
<i>chan</i>	Channel pointer
<i>ph</i>	Received PDU buffer

Definition at line 1076 of file cf\_cfdp.c.

References CF\_AppData, CF\_ASSERT, CF\_CFDP\_DispatchRecv(), CF\_CFDP\_INVALID\_DST\_ERR\_EID, CF\_CFDP\_RecvPh(), CF\_CFDP\_RX\_DROPPED\_ERR\_EID, CF\_CFDP\_SetupRxTransaction(), CF\_CFDP\_StartRxTransaction(), CF\_FindTransactionBySequenceNumber(), CF\_TxnState\_UNDEF, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Engine::channels, CF\_AppData\_t::config\_table, CF\_Logical\_PduHeader::destination\_eid, CF\_AppData\_t::engine, CF\_ConfigTable::local\_eid, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical\_PduHeader::source\_eid, and CF\_Transaction::state.

Referenced by CF\_CFDP\_ReceiveMessage().

Here is the call graph for this function:



**12.35.3.28 CF\_CFDP\_RecvAck()** CFE\_Status\_t CF\_CFDP\_RecvAck (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Unpack an ACK PDU from a received message.

This should only be invoked for buffers that have been identified as an acknowledgment PDU.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

#### Returns

integer status code

#### Return values

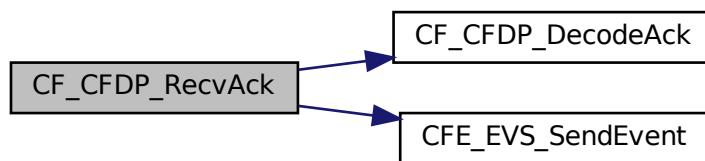
<i>CFE_SUCCESS</i>	on success
<i>CF_SHORT_PDU_ERROR</i>	on error

Definition at line 840 of file cf\_cfdp.c.

References CF\_Logical\_IntHeader::ack, CF\_CFDP\_DecodeAck(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_ACK\_SHORT\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Logical\_PduBuffer::int\_header, and CF\_Logical\_PduBuffer::pdec.

Referenced by CF\_CFDP\_R2\_SubstateRecvFinAck(), and CF\_CFDP\_S2\_SubstateEofAck().

Here is the call graph for this function:



**12.35.3.29 CF\_CFDP\_RecvDrop()** void CF\_CFDP\_RecvDrop (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Receive state function to ignore a packet.

**Description**

This function signature must match all receive state functions. The parameter *txn* is ignored here.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 909 of file cf\_cfdp.c.

References CF\_AppData, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_HkRecv::dropped, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_HkCounters::recv.  
Referenced by CF\_CFDP\_DispatchRecv().

```
12.35.3.30 CF_CFDP_RecvEof() CFE_Status_t CF_CFDP_RecvEof (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Unpack an EOF PDU from a received message.

This should only be invoked for buffers that have been identified as an end of file PDU.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

**Returns**

integer status code

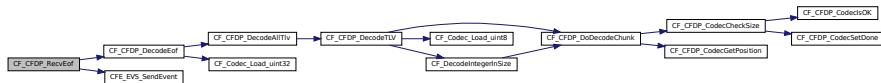
**Return values**

CFE_SUCCESS	on success
CF_SHORT_PDU_ERROR	on error

Definition at line 818 of file cf\_cfdp.c.

References CF\_CFDP\_DecodeEof(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_EOF\_SHORT\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Logical\_IntHeader::eof, CF\_Logical\_PduBuffer::int\_header, and CF\_Logical\_PduBuffer::pdec.  
Referenced by CF\_CFDP\_R\_SubstateRecvEof().

Here is the call graph for this function:



**12.35.3.31 CF\_CFDP\_RecvFd()** `CFE_Status_t CF_CFDP_RecvFd (`  
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph )`

Unpack a file data PDU from a received message.

This should only be invoked for buffers that have been identified as a file data PDU.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

#### Returns

integer status code

#### Return values

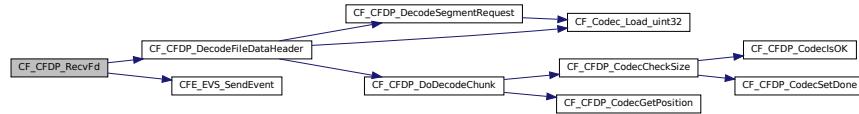
<code>CFE_SUCCESS</code>	on success
<code>CF_ERROR</code>	for general errors
<code>CF_SHORT_PDU_ERROR</code>	PDU too short

Definition at line 774 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_DecodeFileDataHeader(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_ERROR, CF\_PDU\_FD\_SHORT\_ERR\_EID, CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduHeader::crc\_flag, CF\_Logical\_PduFileDataHeader::data\_len, CF\_HkRecv::error, CF\_Logical\_IntHeader::fd, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdec, CF\_Logical\_PduBuffer::pdu\_header, CF\_HkCounters::recv, and CF\_Logical\_PduHeader::segment\_meta\_flag.

Referenced by CF\_CFDP\_R\_SubstateRecvFileData().

Here is the call graph for this function:



**12.35.3.32 CF\_CFDP\_RecvFin()** `CFE_Status_t CF_CFDP_RecvFin (`  
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph )`

Unpack an FIN PDU from a received message.

This should only be invoked for buffers that have been identified as a final PDU.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction state
<code>ph</code>	The logical PDU buffer being received

#### Returns

integer status code

#### Return values

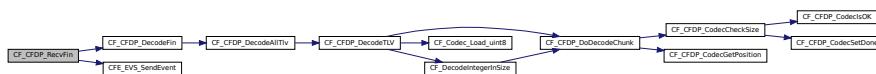
<code>CFE_SUCCESS</code>	on success
<code>CF_SHORT_PDU_ERROR</code>	on error

Definition at line 863 of file cf\_cfdp.c.

References CF\_CFDP\_DecodeFin(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_FIN\_SHORT\_ERR\_←  
 EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_←  
 Logical\_IntHeader::fin, CF\_Logical\_PduBuffer::int\_header, and CF\_Logical\_PduBuffer::pdec.

Referenced by CF\_CFDP\_S\_SubstateRecvFin().

Here is the call graph for this function:



**12.35.3.33 CF\_CFDP\_RecvHold()** `void CF_CFDP_RecvHold (`  
`CF_Transaction_t * txn,`

CF\_Logical\_PduBuffer\_t \* ph )

Receive state function during holdover period.

## Description

This function signature must match all receive state functions. Handles any possible spurious PDUs that might come in after the transaction is considered done. This can happen if ACKs were lost in transmission causing the sender to retransmit PDUs even though we already completed the transaction.

## **Assumptions, External Events, and Notes:**

**txn** must not be `NULL`.

## Parameters

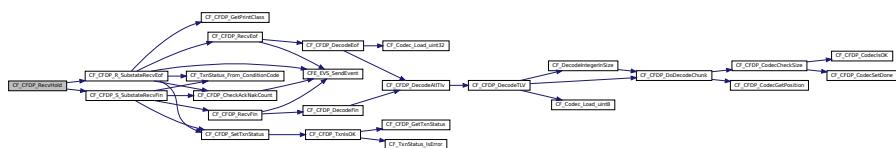
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 920 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_Direction\_TX, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_History::dir, CF\_Logical\_PduFileDirectiveHeader::directive\_code, CF\_Logical\_PduBuffer::fdirective, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_HkCounters::recv, and CF\_HkRecv::spurious.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.35.3.34 CF\_CFDP\_RecvInit()** void CF\_CFDP\_RecvInit (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Receive state function to process new rx transaction.

## Description

An idle transaction has never had message processing performed on it. Typically, the first packet received for a transaction would be the metadata PDU. There's a special case for R2 where the metadata PDU could be missed, and filedata comes in instead. In that case, an R2 transaction must still be started.

### **Assumptions, External Events, and Notes:**

txn must not be NULL. There must be a received message.

### Parameters

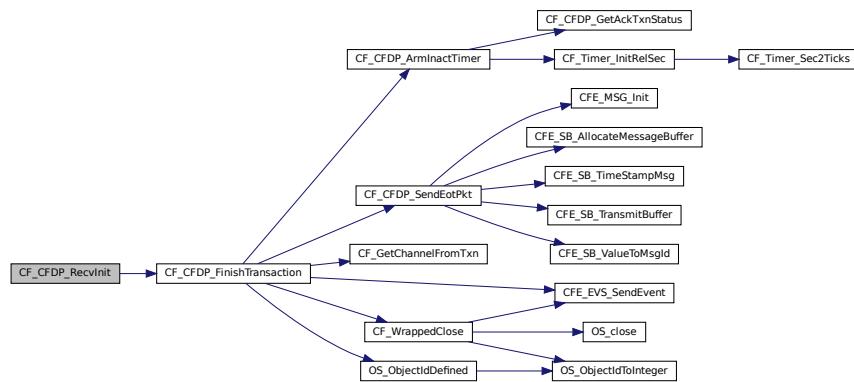
<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

Definition at line 965 of file cf\_cfdp.c.

References CF\_CFDP\_FinishTransaction().

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.35.3.35 CF\_CFDP\_RecvMd()** *CFE\_Status\_t* CF\_CFDP\_RecvMd ( *CF\_Transaction\_t* \* *txn*, *CF\_Logical\_PduBuffer\_t* \* *ph* )

Unpack a metadata PDU from a received message.

This should only be invoked for buffers that have been identified as a metadata PDU.

### Assumptions, External Events, and Notes:

*txn* must not be NULL.

### Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

### Returns

integer status code

### Return values

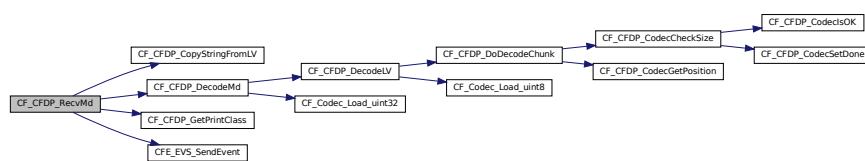
<i>CFE_SUCCESS</i>	on success
<i>CF_PDU_METADATA_ERROR</i>	on error

Definition at line 701 of file cf\_cfdp.c.

References CF\_AppData, CF\_CFDP\_CopyStringFromLV(), CF\_CFDP\_DecodeMd(), CF\_CFDP\_GetPrintClass(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_INVALID\_DST\_LEN\_ERR\_EID, CF\_PDU\_INVALID\_SRC\_LEN\_ERR\_EID, CF\_PDU\_MD\_RECVD\_INF\_EID, CF\_PDU\_MD\_SHORT\_ERR\_EID, CF\_PDU\_METADATA\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_Flags\_Common::close\_req, CF\_Logical\_PduMd::close\_req, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_Logical\_PduMd::dest\_filename, CF\_TxnFilenames::dst\_filename, CF\_HkRecv::error, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::fsiz, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_Lv::length, CF\_Logical\_IntHeader::md, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdec, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_Logical\_PduMd::size, CF\_Logical\_PduMd::source\_filename, CF\_History::src\_eid, and CF\_TxnFilenames::src\_filename.

Referenced by CF\_CFDP\_R\_SubstateRecvMd().

Here is the call graph for this function:



### 12.35.3.36 CF\_CFDP\_RecvNak() CFE\_Status\_t CF\_CFDP\_RecvNak (

```

    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph
)
```

Unpack a NAK PDU from a received message.

This should only be invoked for buffers that have been identified as a negative/non-acknowledgment PDU.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction state
<i>ph</i>	The logical PDU buffer being received

#### Returns

integer status code

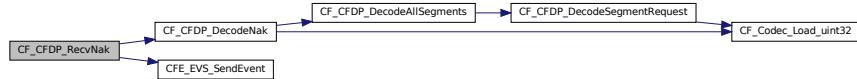
#### Return values

CFE_SUCCESS	on success
CF_SHORT_PDU_ERROR	on error

Definition at line 887 of file cf\_cfdp.c.

References CF\_CFDP\_DecodeNak(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_PDU\_NAK\_SHORT\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_

Logical\_PduBuffer::int\_header, CF\_Logical\_IntHeader::nak, and CF\_Logical\_PduBuffer::pdec.  
Referenced by CF\_CFDP\_S2\_SubstateNak().  
Here is the call graph for this function:



**12.35.3.37 CF\_CFDP\_RecvPh()** `CFE_Status_t CF_CFDP_RecvPh ( uint8 chan_num, CF_Logical_PduBuffer_t * ph )`

Unpack a basic PDU header from a received message.

#### Description

This interprets the common PDU header and the file directive header (if applicable) and populates the logical PDU buffer.

#### Assumptions, External Events, and Notes:

A new message has been received.

#### Parameters

<code>chan_num</code>	The channel number for statistics purposes
<code>ph</code>	The logical PDU buffer being received

#### Returns

integer status code

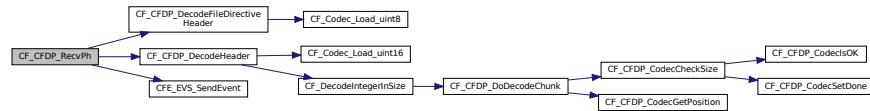
#### Return values

<code>CFE_SUCCESS</code>	on success
<code>CF_ERROR</code>	for general errors
<code>CF_SHORT_PDU_ERROR</code>	if PDU too short

Definition at line 641 of file cf\_cfdp.c.

References CF\_AppData, CF\_ASSERT, CF\_CFDP\_DecodeFileDirectiveHeader(), CF\_CFDP\_DecodeHeader(), CF\_CODEC\_GET\_SIZE, CF\_CODEC\_IS\_OK, CF\_ERROR, CF\_NUM\_CHANNELS, CF\_PDU\_LARGE\_FILE\_ERR\_EID, CF\_PDU\_SHORT\_HEADER\_ERR\_EID, CF\_PDU\_TRUNCATION\_ERR\_EID, CF\_SHORT\_PDU\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_HkRecv::error, CF\_Logical\_PduBuffer::fdirective, CF\_AppData\_t::hk, CF\_Logical\_PduHeader::large\_flag, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdec, CF\_HkRecv::pdu, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::pdu\_type, and CF\_HkCounters::recv.  
Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



### 12.35.3.38 CF\_CFDP\_RecycleTransaction()

```
void CF_CFDP_RecycleTransaction (
```

```
    CF_Transaction_t * txn )
```

Recover resources associated with a transaction.

Wipes all data in the transaction struct and returns everything to its relevant FREE list so it can be used again.

Notably, should any PDUs arrive after this that is related to this transaction, these PDUs will not be identifiable, and no longer associative to this transaction.

#### Assumptions, External Events, and Notes:

It is imperative that nothing uses the txn struct after this call, as it will now be invalid. This is effectively like free().

#### Parameters

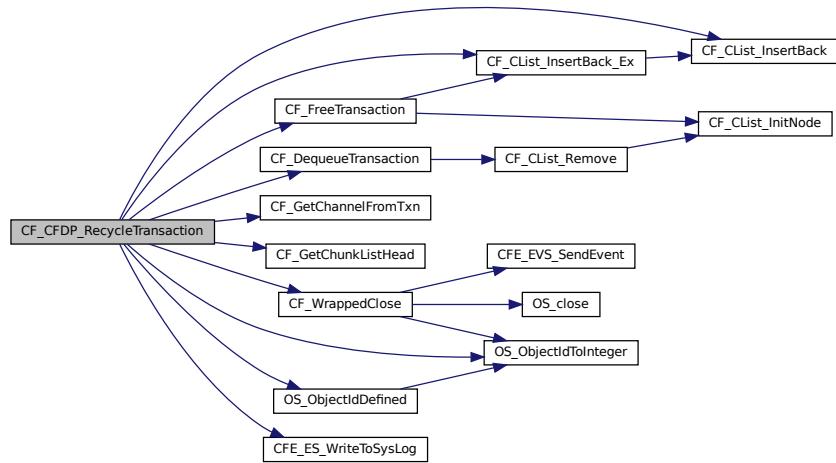
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 1954 of file cf\_cfdp.c.

References CF\_CList\_InsertBack(), CF\_CList\_InsertBack\_Ex(), CF\_DequeueTransaction(), CF\_FreeTransaction(), CF\_GetChannelFromTxn(), CF\_GetChunkListHead(), CF\_QueueIdx\_HIST, CF\_QueueIdx\_HIST\_FREE, CF\_TRACE, CF\_WrappedClose(), CFE\_ES\_WriteToSysLog(), CF\_Transaction::chan\_num, CF\_Transaction::chunks, CF\_History<:cl\_node, CF\_ChunkWrapper::cl\_node, CF\_StateFlags::com, CF\_History::dir, CF\_Transaction::fd, CF\_Transaction<:flags, CF\_Transaction::history, CF\_Flags\_Common::keep\_history, OS\_OBJECT\_ID\_UNDEFINED, OS\_ObjectIdDefined(), OS\_ObjectIdToInteger(), and CF\_History::seq\_num.

Referenced by CF\_CFDP\_R\_Tick(), CF\_CFDP\_S\_Tick(), and CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



**12.35.3.39 CF\_CFDP\_S\_Tick\_NewData()** `void CF_CFDP_S_Tick_NewData ( CF_Transaction_t * txn )`

Tick processor to send new file data.

This helper is used in conjunction with [CF\\_CList\\_Traverse\(\)](#) as part of Tick Processing.

#### Description

This sends new file data, which constitutes chunks of the file that have never been transmitted before.

This attempts to fill the output pipe as much as possible - it will create new PDUs until a transmission limit is reached. As such, it should run only after control traffic is handled. This will consume any unused bandwidth at the end of each wakeup.

#### Assumptions, External Events, and Notes:

`txn` must not be NULL.

#### Parameters

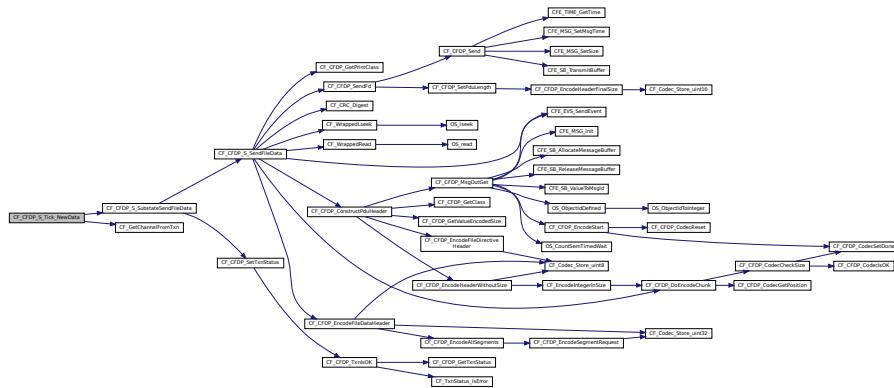
<code>txn</code>	Txn
------------------	-----

Definition at line 1251 of file cf\_cfdp.c.

References `CF_CFDP_S_SubstateSendFileData()`, `CF_GetChannelFromTxn()`, `CF_PERF_ID_PDUSENT`, `CF_TxSubState_DATA_NORMAL`, `CFE_ES_PerfLogEntry`, `CFE_ES_PerfLogExit`, `CF_Transaction::chan_num`, `CF_StateFlags::com`, `CF_Transaction::flags`, `CF_Channel::outgoing_counter`, `CF_Transaction::state_data`, `CF_StateData::sub_state`, and `CF_Flags_Common::suspended`.

Referenced by `CF_CFDP_TickTransactions()`.

Here is the call graph for this function:



**12.35.3.40 CF\_CFDP\_SendAck()** CFE\_Status\_t CF\_CFDP\_SendAck (

```
CF_Transaction_t * txn,  
CF_CFDP_FileDirective_t dir_code )
```

Build an ACK PDU for transmit.

## Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

## Note

`CF_CFDP_SendAck()` takes a `CF_TransactionSeq_t` instead of getting it from transaction history because of the special case where a FIN-ACK must be sent for an unknown transaction. It's better for long term maintenance to not build an incomplete `CF_History_t` for it.

## Parameters

<i>txn</i>	Pointer to the transaction object
<i>dir_code</i>	File directive code being ACK'ed

## Returns

CFE Status t status code

## Return values

<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

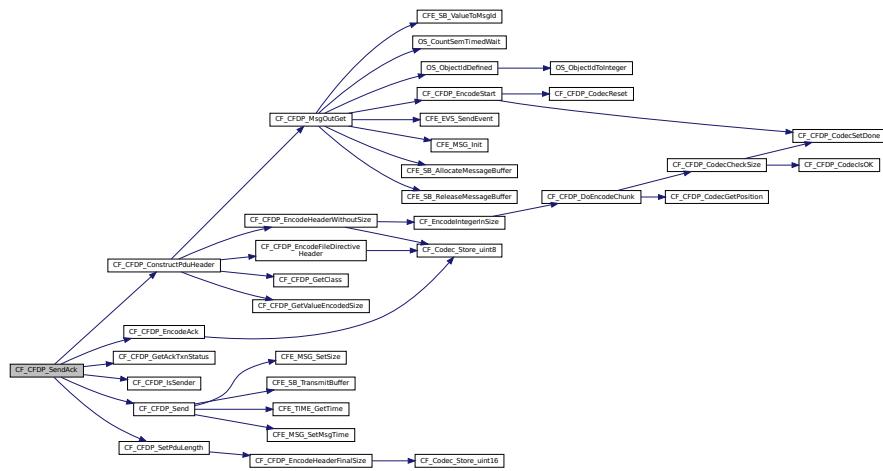
Definition at line 512 of file cf\_cfdp.c.

References CF\_Logical\_IntHeader::ack, CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::ack\_subtype\_code, CF\_Logical\_PduAck::cc, CF\_AppData, CF\_ASSERT, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeAck(), CF\_CFDP\_FileDirective ACK, CF\_CFDP\_FileDirective EOF, CF\_CFDP\_FileDirective FIN, CF\_CFDP\_FileDirective RST, CF\_CFDP\_FileDirective SYN, CF\_CFDP\_FileDirective URG, CF\_CFDP\_FileDirective WindowUpdate, CF\_CFDP\_SetPduHeader()

CFDP\_GetAckTxnStatus(), CF\_CFDP\_IsSender(), CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_SEND ↔ PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TRACE, CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_AppData\_t::config ↔ table, CF\_Transaction::history, CF\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_StateData::peer\_cc, CF\_History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_History::seq\_num, CF\_Transaction::state\_data, and CF\_Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_R\_Tick\_Maintenance(), and CF\_CFDP\_S\_Tick\_Maintenance().

Here is the call graph for this function:



#### 12.35.3.41 CF\_CFDP\_SendEof() CFE\_Status\_t CF\_CFDP\_SendEof ( CFE\_Transaction\_t \* txn )

Build an EOF PDU for transmit.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

##### Parameters

txn	Pointer to the transaction object
-----	-----------------------------------

##### Returns

CFE\_Status\_t status code

##### Return values

CFE_SUCCESS	on success.
CF_SEND_PDU_NO_BUF_AVAIL_ERROR	if message buffer cannot be obtained.

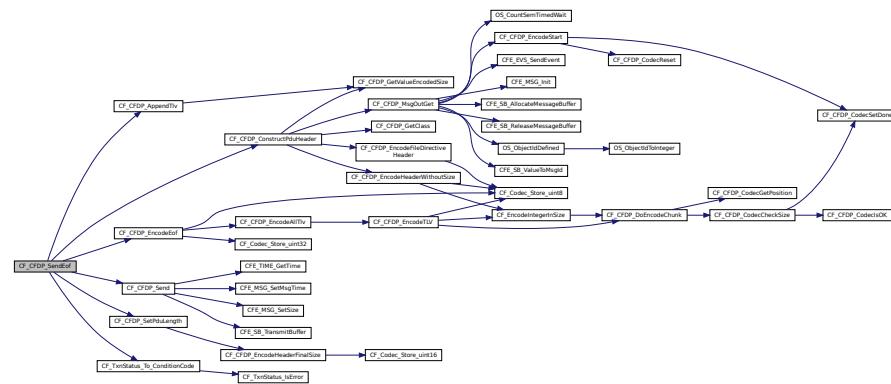
Definition at line 470 of file cf\_cfdp.c.

References CF\_Logical\_PduEof::cc, CF\_AppData, CF\_CFDP\_AppendTlv(), CF\_CFDP\_ConditionCode\_NO\_ERROR, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeEof(), CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_Send(), CF\_CFDL\_SetPduLength(), CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TRACE,

`CF_TxnStatus_To_ConditionCode(), CFE_SUCCESS, CF_Transaction::chan_num, CF_AppData_t::config_table, CF_Transaction::crc, CF_Logical_PduEof::crc, CF_Logical_ExtHeader::eof, CF_Transaction::fsize, CF_Transaction::history, CF_Logical_PduBuffer::int_header, CF_ConfigTable::local_eid, CF_History::peer_eid, CF_Logical_PduBuffer::penc, CF_Crc::result, CF_History::seq_num, CF_Logical_PduEof::size, CF_Logical_PduEof::tlv_list, and CF_History::txn_stat.`

Referenced by CF\_CFDP\_S\_Tick\_Maintenance().

Here is the call graph for this function:



**12.35.3.42 CF\_CFDP\_SendEotPkt()** void CF\_CFDP\_SendEotPkt ( CF\_Transaction\_t \* txn )

Send an end of transaction packet.

## Assumptions, External Events, and Notes:

txn must not be NULL.

## Parameters

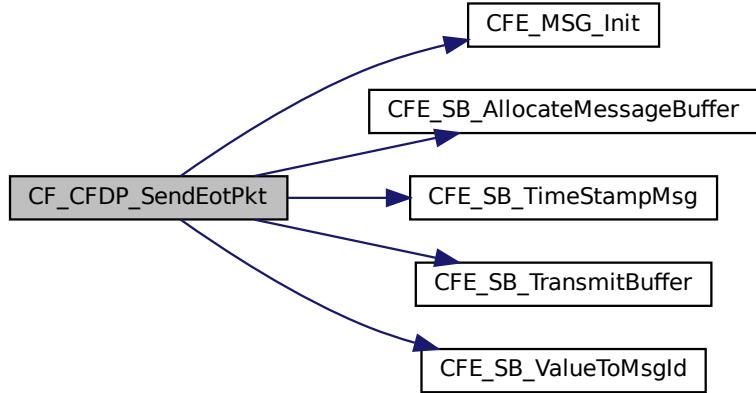
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 2055 of file cf\_cfdp.c.

References CF\_EOT\_TLM\_MID, CFE\_MSG\_Init(), CFE\_SB\_AllocateMessageBuffer(), CFE\_SB\_TimeStampMsg(), CFE\_SB\_TransmitBuffer(), CFE\_SB\_ValueToMsgId(), CF\_Transaction::chan\_num, CF\_EotPacket\_Payload::channel, CF\_Transaction::crc, CF\_EotPacket\_Payload::crc\_result, CF\_History::dir, CF\_EotPacket\_Payload::direction, CF\_EotPacket\_Payload::fnames, CF\_History::fnames, CF\_EotPacket\_Payload::fsize, CF\_Transaction::fsize, CF\_Transaction::history, CF\_EotPacket::Payload, CF\_EotPacket\_Payload::peer\_eid, CF\_History::peer\_eid, CF\_Crc::result, CF\_EotPacket\_Payload::seq\_num, CF\_History::seq\_num, CF\_EotPacket\_Payload::src\_eid, CF\_History::src\_eid, CF\_EotPacket\_Payload::state, CF\_Transaction::state, CF\_EotPacket::TelemetryHeader, CF\_EotPacket\_Payload::txn\_stat, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_FinishTransaction().

Here is the call graph for this function:



**12.35.3.43 CF\_CFDP\_SendFd()** `CFE_Status_t CF_CFDP_SendFd(`  
 `CF_Transaction_t * txn,`  
 `CF_Logical_PduBuffer_t * ph )`

Send a previously-assembled filedata PDU for transmit.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to logical PDU buffer content

#### Note

Unlike other "send" routines, the file data PDU must be acquired and filled by the caller prior to invoking this routine. This routine only sends the PDU that was previously allocated and assembled. As such, the typical failure possibilities do not apply to this call.

#### Returns

`CFE_Status_t` status code

#### Return values

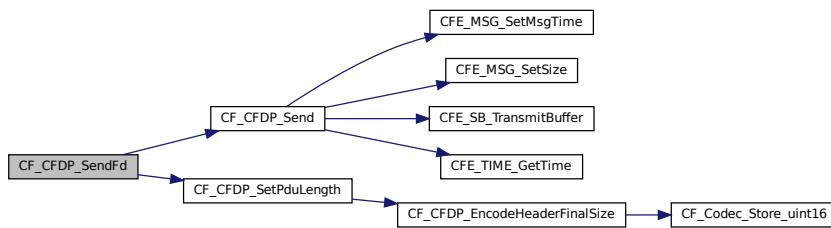
<code>CFE_SUCCESS</code>	on success. (error checks not yet implemented)
--------------------------	--

Definition at line 413 of file cf\_cfdp.c.

References CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CFE\_SUCCESS, and CF\_Transaction::chan\_num.

Referenced by CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:



#### 12.35.3.44 CF\_CFDP\_SendFin() [CFE\\_Status\\_t](#) CF\_CFDP\_SendFin ( [CF\\_Transaction\\_t](#) \* *txn* )

Build a FIN PDU for transmit.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

##### Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

##### Returns

[CFE\\_Status\\_t](#) status code

##### Return values

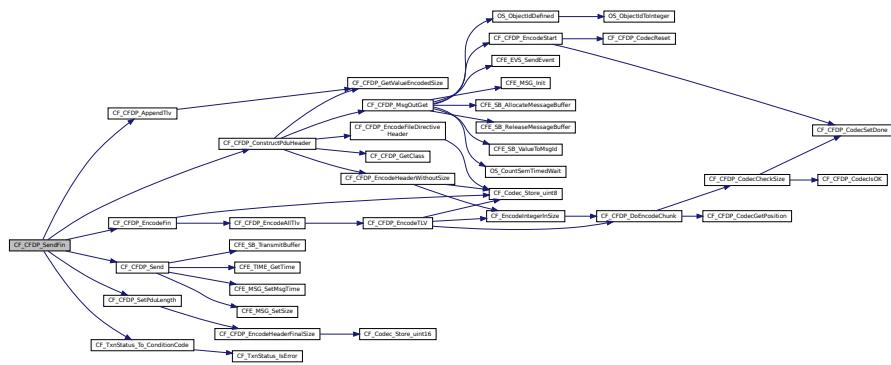
<a href="#">CFE_SUCCESS</a>	on success.
<a href="#">CF_SEND_PDU_NO_BUF_AVAIL_ERROR</a>	if message buffer cannot be obtained.

Definition at line 568 of file cf\_cfdp.c.

References CF\_Logical\_PduFin::cc, CF\_AppData, CF\_CFDP\_AppendTlv(), CF\_CFDP\_ConditionCode\_NO\_ERROR, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeFin(), CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TRACE, CF\_TxnStatus\_To\_ConditionCode(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_AppData\_t::config\_table, CF\_Logical\_PduFin::delivery\_code, CF\_Logical\_PduFin::file\_status, CF\_Logical\_IntHeader::fin, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::history, CF\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_History::seq\_num, CF\_Transaction::state\_data, CF\_Logical\_PduFin::tlv\_list, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_R\_Tick\_Maintenance().

Here is the call graph for this function:



**12.35.3.45 CF\_CFDP\_SendMd()** `CFE_Status_t` `CF_CFDP_SendMd (`  
`CF_Transaction_t * txn )`

Build a metadata PDU for transmit.

## Assumptions, External Events, and Notes

text must not be RELE.

## Parameters

*txn* Pointer to the transaction object

## Returns

CFE\_Status\_t status code

## Return values

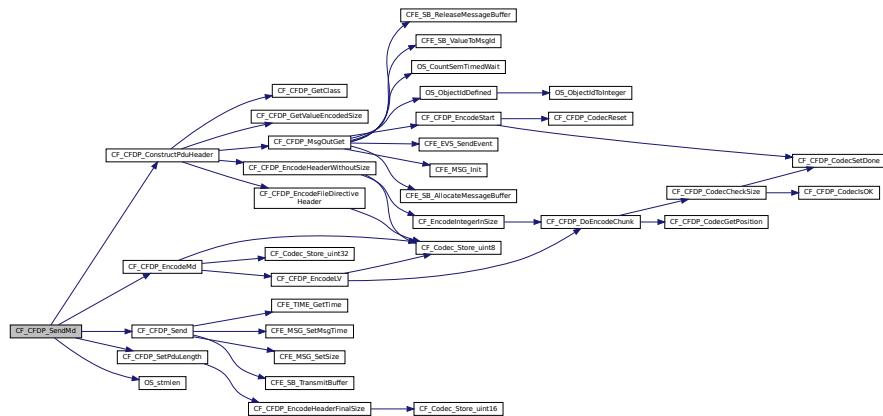
<i>CFE_SUCCESS</i>	on success.
<i>CF_SEND_PDU_NO_BUF_AVAIL_ERROR</i>	if message buffer cannot be obtained.

Definition at line 366 of file cf\_cfdp.c.

References CF\_AppData, CF\_Assert, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_EncodeMd(), CF\_CFDP\_File↔  
 Directive\_METADATA, CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR,  
 CF\_TRACE, CF\_TxnState\_S1, CF\_TxnState\_S2, CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_Logical\_Pdu↔  
 Md::checksum\_type, CF\_Flags\_Common::close\_req, CF\_Logical\_PduMd::close\_req, CF\_StateFlags::com, CF\_App↔  
 Data\_t::config\_table, CF\_Logical\_Lv::data\_ptr, CF\_Logical\_PduMd::dest\_filename, CF\_TxnFilenames::dst\_filename,  
 CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::fsize, CF\_Transaction::history, CF\_Logical\_PduBuffer↔  
 ::int\_header, CF\_Logical\_Lv::length, CF\_ConfigTable::local\_eid, CF\_Logical\_IntHeader::md, OS\_strlen(), CF↔  
 History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_History::seq\_num, CF\_Logical\_PduMd::size, CF\_Logical\_Pdu↔  
 Md::source\_filename, CF\_TxnFilenames::src\_filename, and CF\_Transaction::state.

Referenced by CF\_CFDP\_S\_Tick\_Maintenance().

Here is the call graph for this function:



```
12.35.3.46 CF_CFDP_SendNak() void CF_CFDP_SendNak (
```

CF_Transaction_t * txn,	CF_Logical_PduBuffer_t * ph )
-------------------------	-------------------------------

Send a previously-assembled NAK PDU for transmit.

## **Assumptions, External Events, and Notes:**

**txn** must not be `NULL`.

## Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to logical PDU buffer content

## Note

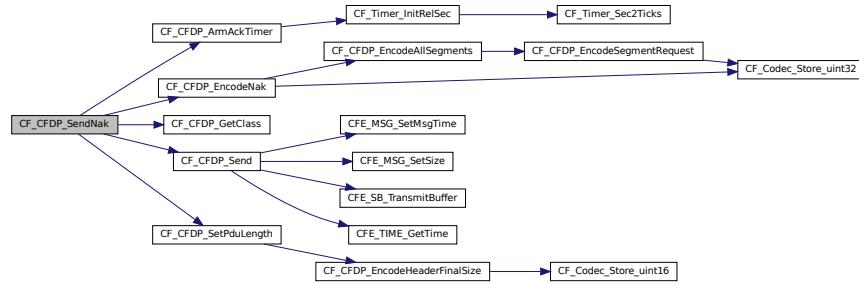
Unlike other "send" routines, the NAK PDU must be acquired and filled by the caller prior to invoking this routine. This routine only encodes and sends the previously-assembled PDU buffer. As such, the typical failure possibilities do not apply to this call.

Definition at line 612 of file cf\_cfdp.c.

References CF\_ASSERT, CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_CLASS\_2, CF\_CFDP\_EncodeNak(), CF\_CFDP\_GetClass(), CF\_CFDP\_Send(), CF\_CFDP\_SetPduLength(), CF\_TRACE, CF\_Transaction::chan\_num, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_IntHeader::nak, CF\_Logical\_SegmentList::num\_segments, CF\_Logical\_PduBuffer::penc, and CF\_Logical\_PduNak::segment\_list.

Referenced by CF CFDP R SendNak().

Here is the call graph for this function:



#### 12.35.3.47 CF\_CFDP\_SetTxnStatus()

```
void CF_CFDP_SetTxnStatus (
    CF_Transaction_t * txn,
    CF_TxnStatus_t txn_stat )
```

Helper function to store transaction status code only.

This records the status in the history block but does not set FIN flag or take any other protocol/state machine actions.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction object
<i>txn_stat</i>	Status Code value to set within transaction

Definition at line 2014 of file cf\_cfdp.c.

References CF\_CFDP\_TxnIsOK(), CF\_Transaction::history, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_AllocChunkList(), CF\_CFDP\_CancelTransaction(), CF\_CFDP\_R\_CalcCrcChunk(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_R\_Init(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_Init(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_S\_SubstateSendFileData(), and CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



#### 12.35.3.48 CF\_CFDP\_SetupRxTransaction()

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Sets up a new RX transaction based on PDU.

Sets up a new RX transaction.

All RX transactions are initiated based on the reception of a PDU with this node as the dest entity and a sequence number that does not match any existing transaction.

When the given PDU requires a new transaction to be created, this initializes the new transaction object. It sets up the chunklist and all the other required supplemental data structs.

## Assumptions, External Events, and Notes:

None

## Parameters

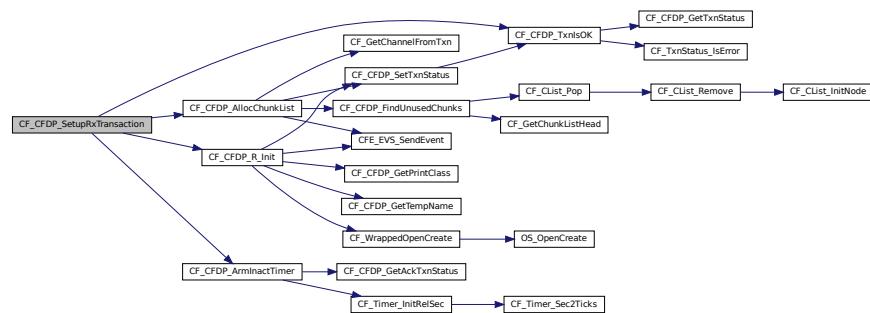
<i>txn</i>	Transaction pointer
<i>ph</i>	Received PDU buffer

Definition at line 1033 of file cf\_cfdp.c.

References CF\_CFDP\_AllocChunkList(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_R\_Init(), CF\_CFDP\_TxnIsOK(), CF\_TxnState\_HOLD, CF\_TxnState\_R1, CF\_TxnState\_R2, CF\_Transaction::chunks, CF\_Transaction::history, CF\_Logical\_PduBuffer::pdu\_header, CF\_History::peer\_eid, CF\_Transaction::reliable\_mode, CF\_History::seq\_num, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical\_PduHeader::source\_eid, CF\_History::src\_eid, CF\_Transaction::state, and CF\_Logical\_PduHeader::txm\_mode.

Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



**12.35.3.49 CF\_CFDP\_SetupTxTransaction()** void CF\_CFDP\_SetupTxTransaction ( CF\_Transaction\_t \* txn )

begin

TX transactions are initiated by

It sets up the chunklist and all the other required supplemental data structs and puts the transaction onto the TX queue, so that it is now usable by the engine.

## Assumptions, External Events, and Notes:

Transactions come from the PEND queue

## Parameters

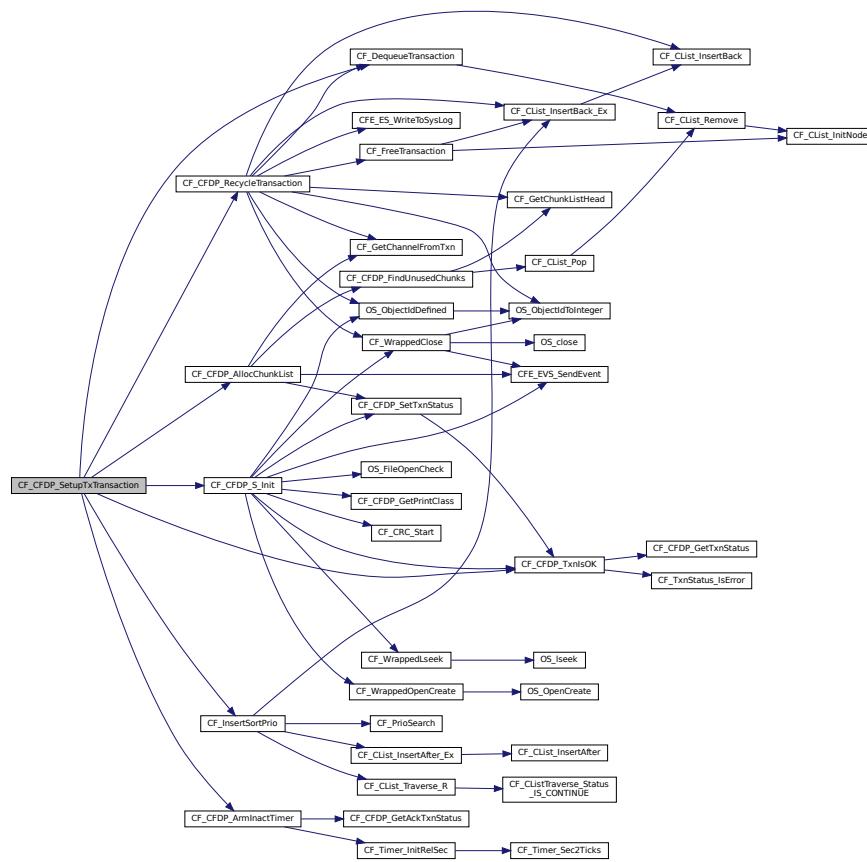
<code>txn</code>	Transaction pointer
------------------	---------------------

Definition at line 998 of file cf\_cfdp.c.

References CF\_CFDP\_AllocChunkList(), CF\_CFDP\_ArmInactTimer(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S\_Init(), CF\_CFDP\_TxnIsOK(), CF\_DequeueTransaction(), CF\_InsertSortPrio(), CF\_QueueIdx\_TX, and CF\_Transaction::chunks.

Referenced by CF\_CFDP\_StartFirstPending().

Here is the call graph for this function:



**12.35.3.50 CF\_CFDP\_StartFirstPending()** `bool CF_CFDP_StartFirstPending ( CF_Channel_t * chan )`

Pulls next TX transaction from the PEND queue.

If the PEND queue is not empty, pulls the first transaction from it, and prepares for it to be activated.

**Assumptions, External Events, and Notes:**

## Parameters

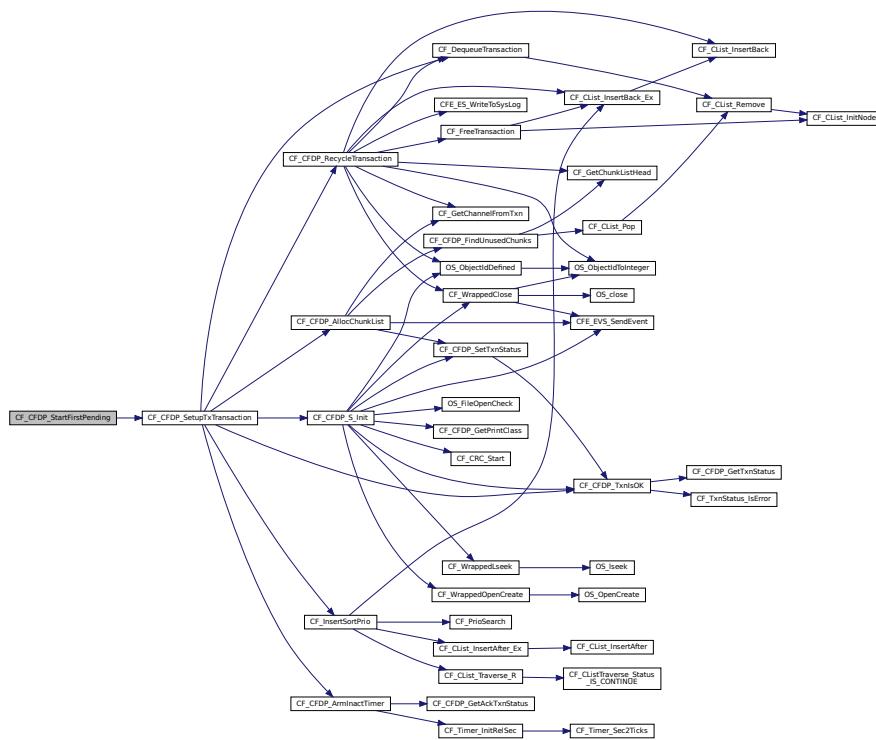
<code>chan</code>	Channel pointer
-------------------	-----------------

Definition at line 1300 of file cf\_cfdp.c.

References CF\_CFDP\_SetupTxTransaction(), CF\_QueueIdx\_PEND, container\_of, and CF\_Channel::qs.

Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



### 12.35.3.51 CF\_CFDP\_StartRxTransaction()

```
CF_Transaction_t* CF_CFDP_StartRxTransaction (
    uint8 chan_num )
```

Helper function to start a new RX transaction.

This sets various fields inside a newly-allocated transaction structure appropriately for receiving a file. Note that in the receive direction, most fields are unknown until the MD is received, and thus are left in their initial state here (generally 0).

If there is no capacity for another RX transaction, this returns NULL.

## Parameters

<code>chan_num</code>	CF channel number
-----------------------	-------------------

**Returns**

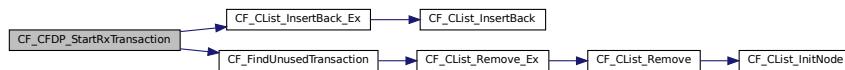
Pointer to new transaction

Definition at line 1576 of file cf\_cfdp.c.

References CF\_AppData, CF\_CList\_InsertBack\_Ex(), CF\_Direction\_RX, CF\_FindUnusedTransaction(), CF\_MAXSIMULTANEOUS\_RX, CF\_QueueIdx\_RX, CF\_TRACE, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Transaction::cl\_node, CF\_StateFlags::com, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Flags\_Common::q\_index, and CF\_HkChannel\_Data::q\_size.

Referenced by CF\_CFDP\_ReceivePdu().

Here is the call graph for this function:



### **12.35.3.52 CF\_CFDP\_TickTransactions()** void CF\_CFDP\_TickTransactions ( CF\_Channel\_t \* chan )

Call R and then S tick functions for all active transactions.

**Description**

Traverses all transactions in the RX and TXW queues, and calls their tick functions. Note that the TXW queue is used twice: once for regular tick processing, and one for NAK response.

**Assumptions, External Events, and Notes:**

chan must not be NULL.

**Parameters**

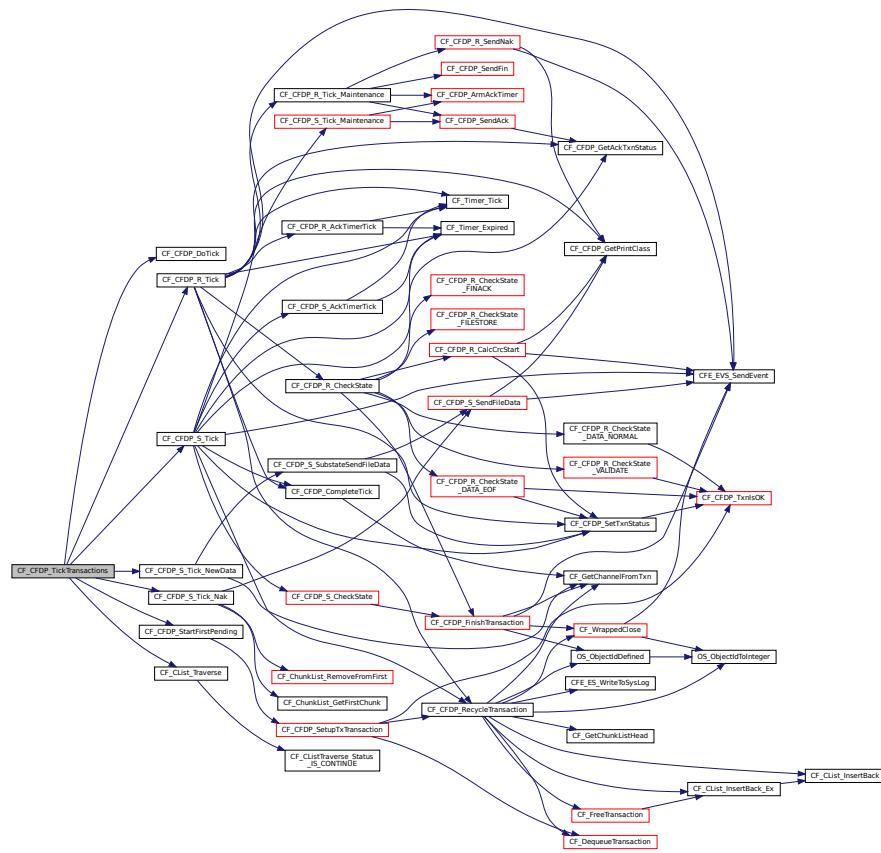
chan	Channel to tick
------	-----------------

Definition at line 1375 of file cf\_cfdp.c.

References CF\_CFDP\_DoTick(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S\_Tick(), CF\_CFDP\_S\_Tick\_Nak(), CF\_CFDP\_STick\_NewData(), CF\_CFDP\_StartFirstPending(), CF\_CList\_Traverse(), CF\_QueueIdx\_RX, CF\_QueueIdx\_TX, CF\_TickState\_COMPLETE, CF\_TickState\_INIT, CF\_TickState\_RX\_STATE, CF\_TickState\_TX\_FILEDATA, CF\_TickState\_TX\_NAK, CF\_TickState\_TX\_PEND, CF\_TickState\_TX\_STATE, CF\_CFDP\_Tick\_args::chan, CF\_CFDP\_Tick\_args::fn, CF\_Channel::outgoing\_counter, CF\_Channel::qs, CF\_CFDP\_Tick\_args::resume\_point, CF\_Channel::tick\_resume, and CF\_Channel::tx\_blocked.

Referenced by CF\_CFDP\_CycleEngine().

Here is the call graph for this function:



```
12.35.3.53 CF_CFDP_TxFile() CFE_Status_t CF_CFDP_TxFile (
    const char * src_filename,
    const char * dst_filename,
    CF_CFDP_Class_t cfdp_class,
    uint8 keep,
    uint8 chan,
    uint8 priority,
    CF_EntityId_t dest_id )
```

Begin transmit of a file.

#### Description

This function sets up a transaction for and starts transmit of the given filename.

#### Assumptions, External Events, and Notes:

src\_filename must not be NULL. dst\_filename must not be NULL.

### Parameters

<i>src_filename</i>	Local filename
<i>dst_filename</i>	Remote filename
<i>cfdp_class</i>	Whether to perform a class 1 or class 2 transfer
<i>keep</i>	Whether to keep or delete the local file after completion
<i>chan</i>	CF channel number to use
<i>priority</i>	CF priority level
<i>dest_id</i>	Entity ID of remote receiver

### Return values

<b>CFE_SUCCESS</b>	Successful execution. Operation was performed successfully
--------------------	--

### Returns

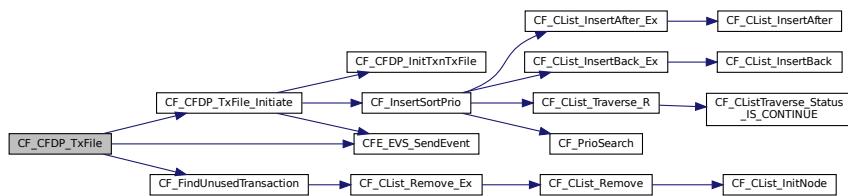
CFE\_SUCCESS on success. CF\_ERROR on error.

Definition at line 1528 of file cf\_cfdp.c.

References CF\_AppData, CF\_ASSERT, CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID, CF\_CFDP\_TxFile\_Initiate(), CF\_DIRECTION\_TX, CF\_ERROR, CF\_FindUnusedTransaction(), CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN, CF\_NUM\_CHANNELS, CF\_TRACE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Engine::channels, CF\_Flags\_Tx::cmd\_tx, CF\_TxnFilenames::dst\_filename, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::history, CF\_Channel::num\_cmd\_tx, CF\_TxnFilenames::src\_filename, and CF\_StateFlags::tx.

Referenced by CF\_TxFileCmd().

Here is the call graph for this function:



**12.35.3.54 CF\_CFDP\_TxnIsOK()** **CF\_TxnStatus\_t** CF\_CFDP\_TxnIsOK (   
 const **CF\_Transaction\_t** \* *txn* )

Helper function to check if a transaction is errored.

Simplifies conditionals where state machines need to check if a transaction is OK to continue operating normally

### Assumptions, External Events, and Notes:

*txn* must not be NULL.

### Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

### Return values

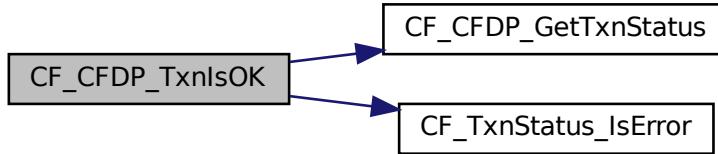
<i>true</i>	if transaction is in a good state, no errors
<i>false</i>	if transaction has an error

Definition at line 2044 of file cf\_cfdp.c.

References CF\_CFDP\_GetTxnStatus(), and CF\_TxnStatus\_IsError().

Referenced by CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_CheckState\_VALIDATE(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_HandleFileRetention(), CF\_CFDP\_S\_Init(), CF\_CFDP\_SetTxnStatus(), CF\_CFDP\_SetupRxTransaction(), and CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



## 12.36 apps/cf/fsw/src(cf\_cfdp\_dispatch.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_dispatch.h"
#include <stdio.h>
#include <string.h>
#include "cf_assert.h"
```

### Functions

- void **CF\_CFDP\_R\_DispatchRecv** (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph, const CF\_CFDP\_R\_SubstateDispatchTable\_t \*dispatch, CF\_CFDP\_StateRecvFunc\_t fd\_fn)  
*Dispatch function for received PDUs on receive-file transactions.*
- void **CF\_CFDP\_S\_DispatchRecv** (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph, const CF\_CFDP\_S\_SubstateRecvDispatchTable\_t \*dispatch)  
*Dispatch function for received PDUs on send-file transactions.*
- void **CF\_CFDP\_RxStateDispatch** (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph, const CF\_CFDP\_TxnRecvDispatchTable\_t \*dispatch)  
*Top-level Dispatch function receive a PDU based on current state of a transaction.*

### 12.36.1 Function Documentation

```
12.36.1.1 CF_CFDP_R_DispatchRecv() void CF_CFDP_R_DispatchRecv (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph,
    const CF_CFDP_R_SubstateDispatchTable_t * dispatch,
    CF_CFDP_StateRecvFunc_t fd_fn )
```

Dispatch function for received PDUs on receive-file transactions.

Receive file transactions primarily only react/respond to received PDUs

#### Parameters

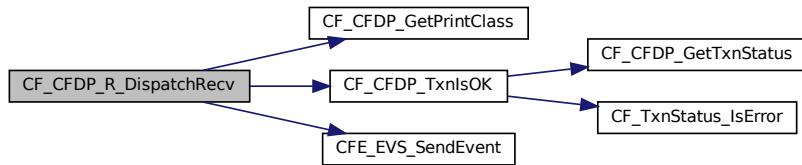
<i>txn</i>	Transaction
<i>ph</i>	PDU Buffer
<i>dispatch</i>	Dispatch table for file directive PDUs
<i>fd_fn</i>	Function to handle file data PDUs

Definition at line 40 of file cf\_cfdp\_dispatch.c.

References CF\_AppData, CF\_CFDP\_FileDirective\_INVALID\_MAX, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_DC::INV\_ERR\_EID, CF\_CFDP\_TxnIsOK(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduFileDirective::Header::directive\_code, CF\_HkRecv::dropped, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, CF\_Logical\_Pdu::Buffer::fdirective, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdu::header, CF\_Logical\_PduHeader::pdu\_type, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_HkRecv::spurious, CF\_History::src\_eid, CF\_CFDP\_R\_SubstateDispatchTable\_t::state, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_R1\_Recv(), and CF\_CFDP\_R2\_Recv().

Here is the call graph for this function:



```
12.36.1.2 CF_CFDP_RxStateDispatch() void CF_CFDP_RxStateDispatch (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph,
    const CF_CFDP_TxnRecvDispatchTable_t * dispatch )
```

Top-level Dispatch function receive a PDU based on current state of a transaction.

#### Parameters

<i>txn</i>	Transaction
------------	-------------

**Parameters**

<i>ph</i>	Received PDU Buffer
<i>dispatch</i>	Transaction State-based Dispatch table

Definition at line 151 of file cf\_cfdp\_dispatch.c.

References CF\_ASSERT, CF\_TxnState\_DROP, CF\_TxnState\_INVALID, CF\_Logical\_PduBuffer::pdu\_header, CF\_Transaction::reliable\_mode, CF\_CFDP\_TxnRecvDispatchTable\_t::rx, CF\_Transaction::state, and CF\_Logical\_PduHeader::txm\_mode.

Referenced by CF\_CFDP\_DispatchRecv().

```
12.36.1.3 CF_CFDP_S_DispatchRecv() void CF_CFDP_S_DispatchRecv (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph,
    const CF_CFDP_S_SubstateRecvDispatchTable_t * dispatch )
```

Dispatch function for received PDUs on send-file transactions.

Send file transactions also react/respond to received PDUs. Note that a file data PDU is not expected here.

**Parameters**

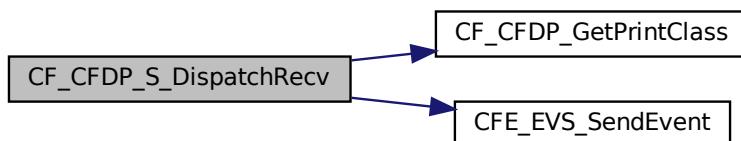
<i>txn</i>	Transaction
<i>ph</i>	PDU Buffer
<i>dispatch</i>	Dispatch table for file directive PDUs

Definition at line 96 of file cf\_cfdp\_dispatch.c.

References CF\_AppData, CF\_CFDP\_FileDirective\_INVALID\_MAX, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_DC\_INV\_ERR\_EID, CF\_CFDP\_S\_NON\_FD\_PDU\_ERR\_EID, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduFileDirectiveHeader::directive\_code, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, CF\_Logical\_PduBuffer::fdirective, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::pdu\_type, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_HkRecv::spurious, CF\_History::src\_eid, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_CFDP\_S\_SubstateRecvDispatchTable\_t::substate.

Referenced by CF\_CFDP\_S1\_Recv(), and CF\_CFDP\_S2\_Recv().

Here is the call graph for this function:



## Data Structures

- struct [CF\\_CFDP\\_TxnSendDispatchTable\\_t](#)  
*A table of transmit handler functions based on transaction state.*
- struct [CF\\_CFDP\\_TxnRecvDispatchTable\\_t](#)  
*A table of receive handler functions based on transaction state.*
- struct [CF\\_CFDP\\_FileDirectiveDispatchTable\\_t](#)  
*A table of receive handler functions based on file directive code.*
- struct [CF\\_CFDP\\_R\\_SubstateDispatchTable\\_t](#)  
*A dispatch table for receive file transactions, receive side.*
- struct [CF\\_CFDP\\_S\\_SubstateRecvDispatchTable\\_t](#)  
*A dispatch table for send file transactions, receive side.*
- struct [CF\\_CFDP\\_S\\_SubstateSendDispatchTable\\_t](#)  
*A dispatch table for send file transactions, transmit side.*

## Typedefs

- typedef void(\* [CF\\_CFDP\\_StateSendFunc\\_t](#)) ([CF\\_Transaction\\_t](#) \*txn)  
*A function for dispatching actions to a handler, without existing PDU data.*
- typedef void(\* [CF\\_CFDP\\_StateRecvFunc\\_t](#)) ([CF\\_Transaction\\_t](#) \*txn, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph)  
*A function for dispatching actions to a handler, with existing PDU data.*

## Functions

- void [CF\\_CFDP\\_R\\_DispatchRecv](#) ([CF\\_Transaction\\_t](#) \*txn, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph, const [CF\\_CFDP\\_R\\_SubstateDispatchTable\\_t](#) \*dispatch, [CF\\_CFDP\\_StateRecvFunc\\_t](#) fd\_fn)  
*Dispatch function for received PDUs on receive-file transactions.*
- void [CF\\_CFDP\\_S\\_DispatchRecv](#) ([CF\\_Transaction\\_t](#) \*txn, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph, const [CF\\_CFDP\\_S\\_SubstateRecvDispatchTable\\_t](#) \*dispatch)  
*Dispatch function for received PDUs on send-file transactions.*
- void [CF\\_CFDP\\_RxStateDispatch](#) ([CF\\_Transaction\\_t](#) \*txn, [CF\\_Logical\\_PduBuffer\\_t](#) \*ph, const [CF\\_CFDP\\_TxnRecvDispatchTable\\_t](#) \*dispatch)  
*Top-level Dispatch function receive a PDU based on current state of a transaction.*

### 12.37.1 Detailed Description

Common routines to dispatch operations based on a transaction state and/or received PDU type.

### 12.37.2 Typedef Documentation

#### 12.37.2.1 [CF\\_CFDP\\_StateRecvFunc\\_t](#) `typedef void(* CF_CFDP_StateRecvFunc_t) (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`

A function for dispatching actions to a handler, with existing PDU data.

This allows quick delegation of PDUs to handler functions using dispatch tables. This version is used on the receive side where a PDU buffer is associated with the activity, which is then interpreted by the handler being invoked.

#### Parameters

in, out	<i>txn</i>	The transaction object
in, out	<i>ph</i>	The PDU buffer currently being received/processed

Definition at line 53 of file cf\_cfdp\_dispatch.h.

**12.37.2.2 CF\_CFDP\_StateSendFunc\_t** `typedef void(* CF_CFDP_StateSendFunc_t) (CF_Transaction_t *txn)`

A function for dispatching actions to a handler, without existing PDU data.

This allows quick delegation to handler functions using dispatch tables. This version is used on the transmit side, where a PDU will likely be generated/sent by the handler being invoked.

#### Parameters

<code>in, out</code>	<code>txn</code>	The transaction object
----------------------	------------------	------------------------

Definition at line 41 of file cf\_cfdp\_dispatch.h.

### 12.37.3 Function Documentation

**12.37.3.1 CF\_CFDP\_R\_DispatchRecv()** `void CF_CFDP_R_DispatchRecv (`

<code>CF_Transaction_t * txn,</code>
<code>CF_Logical_PduBuffer_t * ph,</code>
<code>const CF_CFDP_R_SubstateDispatchTable_t * dispatch,</code>
<code>CF_CFDP_StateRecvFunc_t fd_fn )</code>

Dispatch function for received PDUs on receive-file transactions.

Receive file transactions primarily only react/respond to received PDUs

#### Parameters

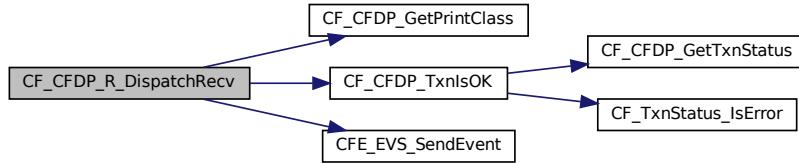
<code>txn</code>	Transaction
<code>ph</code>	PDU Buffer
<code>dispatch</code>	Dispatch table for file directive PDUs
<code>fd_fn</code>	Function to handle file data PDUs

Definition at line 40 of file cf\_cfdp\_dispatch.c.

References CF\_AppData, CF\_CFDP\_FileDirective\_INVALID\_MAX, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_DC\_Inv\_EID, CF\_CFDP\_TxnIsOK(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduFileDirective::Header::directive\_code, CF\_HkRecv::dropped, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, CF\_Logical\_PduBuffer::fdirective, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::pdu\_type, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_HkRecv::spurious, CF\_History::src\_eid, CF\_CFDP\_R\_SubstateDispatchTable\_t::state, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_R1\_Recv(), and CF\_CFDP\_R2\_Recv().

Here is the call graph for this function:



**12.37.3.2 CF\_CFDP\_RxStateDispatch()** `void CF_CFDP_RxStateDispatch (`  
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph,`  
`const CF_CFDP_TxnRecvDispatchTable_t * dispatch )`

Top-level Dispatch function receive a PDU based on current state of a transaction.

#### Parameters

<i>txn</i>	Transaction
<i>ph</i>	Received PDU Buffer
<i>dispatch</i>	Transaction State-based Dispatch table

Definition at line 151 of file cf\_cfdp\_dispatch.c.

References `CF_ASSERT`, `CF_TxnState_DROP`, `CF_TxnState_INVALID`, `CF_Logical_PduBuffer::pdu_header`, `CF_Transaction::reliable_mode`, `CF_CFDP_TxnRecvDispatchTable_t::rx`, `CF_Transaction::state`, and `CF_Logical_PduHeader::txm_mode`.

Referenced by `CF_CFDP_DispatchRecv()`.

**12.37.3.3 CF\_CFDP\_S\_DispatchRecv()** `void CF_CFDP_S_DispatchRecv (`  
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph,`  
`const CF_CFDP_S_SubstateRecvDispatchTable_t * dispatch )`

Dispatch function for received PDUs on send-file transactions.

Send file transactions also react/respond to received PDUs. Note that a file data PDU is not expected here.

#### Parameters

<i>txn</i>	Transaction
<i>ph</i>	PDU Buffer
<i>dispatch</i>	Dispatch table for file directive PDUs

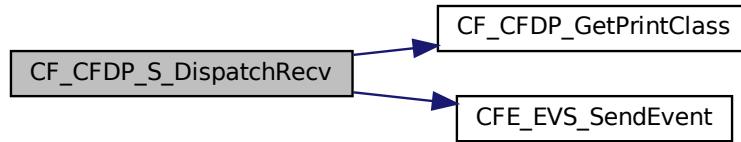
Definition at line 96 of file cf\_cfdp\_dispatch.c.

References `CF_AppData`, `CF_CFDP_FileDirective_INVALID_MAX`, `CF_CFDP_GetPrintClass()`, `CF_CFDP_S_DC_INV_ERR_EID`, `CF_CFDP_S_NON_FD_PDU_ERR_EID`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_HkChannel_Data::counters`, `CF_Logical_PduFileDirectiveHeader::directive_code`, `CF_CFDP_FileDirectiveDispatchTable_t::fdirective`, `CF_Logical_PduBuffer`,

::fdirective, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdu\_header, CF\_Logical\_PduHeader::pdu\_type, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_HkRecv::spurious, CF\_History::src\_eid, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_CFDP\_S\_SubstateRecvDispatchTable\_t::substate.

Referenced by CF\_CFDP\_S1\_Recv(), and CF\_CFDP\_S2\_Recv().

Here is the call graph for this function:



## 12.38 apps/cf/fsw/src/cf\_cfdp\_pdu.h File Reference

```
#include "common_types.h"
#include "cf_platform_cfg.h"
#include <stddef.h>
```

### Data Structures

- struct [CF\\_CFDP\\_uint8\\_t](#)  
*Encoded 8-bit value in the CFDP PDU.*
- struct [CF\\_CFDP\\_uint16\\_t](#)  
*Encoded 16-bit value in the CFDP PDU.*
- struct [CF\\_CFDP\\_uint32\\_t](#)  
*Encoded 32-bit value in the CFDP PDU.*
- struct [CF\\_CFDP\\_uint64\\_t](#)  
*Encoded 64-bit value in the CFDP PDU.*
- struct [CF\\_CFDP\\_PduHeader](#)  
*Structure representing base CFDP PDU header.*
- struct [CF\\_CFDP\\_PduFileDirectiveHeader](#)  
*Structure representing CFDP File Directive Header.*
- struct [CF\\_CFDP\\_lv](#)  
*Structure representing CFDP LV Object format.*
- struct [CF\\_CFDP\\_tlv](#)  
*Structure representing CFDP TLV Object format.*
- struct [CF\\_CFDP\\_PduEof](#)  
*Structure representing CFDP End of file PDU.*
- struct [CF\\_CFDP\\_PduFin](#)  
*Structure representing CFDP Finished PDU.*
- struct [CF\\_CFDP\\_PduAck](#)  
*Structure representing CFDP Acknowledge PDU.*
- struct [CF\\_CFDP\\_SegmentRequest](#)

*Structure representing CFDP Segment Request.*

- struct **CF\_CFDP\_PduNak**  
*Structure representing CFDP Non-Acknowledge PDU.*
- struct **CF\_CFDP\_PduMd**  
*Structure representing CFDP Metadata PDU.*
- struct **CF\_CFDP\_PduFileDataHeader**  
*PDU file data header.*
- struct **CF\_CFDP\_PduFileDialogContent**  
*PDU file data content typedef for limit checking outgoing\_file\_chunk\_size table value and set parameter command.*

## Macros

- #define **CF\_CFDP\_MAX\_HEADER\_SIZE** (sizeof(CF\_CFDP\_PduHeader\_t) + (3 \* sizeof(CF\_CFDP\_uint64\_t)))  
/\* 8 bytes for each variable item \*/  
*Maximum encoded size of a CFDP PDU header.*
- #define **CF\_CFDP\_MIN\_HEADER\_SIZE** (sizeof(CF\_CFDP\_PduHeader\_t) + (3 \* sizeof(CF\_CFDP\_uint8\_t))) /\* 1 byte for each variable item \*/  
*Minimum encoded size of a CFDP PDU header.*
- #define **CF\_APP\_MAX\_HEADER\_SIZE** (sizeof(CF\_CFDP\_PduHeader\_t) + sizeof(CF\_TransactionSeq\_t) + (3 \* sizeof(CF\_EntityId\_t)))  
*Maximum encoded size of a CFDP PDU that this implementation can accept.*

## Typedefs

- typedef struct **CF\_CFDP\_PduHeader** **CF\_CFDP\_PduHeader\_t**  
*Structure representing base CFDP PDU header.*
- typedef struct **CF\_CFDP\_PduFileDirectiveHeader** **CF\_CFDP\_PduFileDirectiveHeader\_t**  
*Structure representing CFDP File Directive Header.*
- typedef struct **CF\_CFDP\_lv** **CF\_CFDP\_lv\_t**  
*Structure representing CFDP LV Object format.*
- typedef struct **CF\_CFDP\_tlv** **CF\_CFDP\_tlv\_t**  
*Structure representing CFDP TLV Object format.*
- typedef struct **CF\_CFDP\_PduEof** **CF\_CFDP\_PduEof\_t**  
*Structure representing CFDP End of file PDU.*
- typedef struct **CF\_CFDP\_PduFin** **CF\_CFDP\_PduFin\_t**  
*Structure representing CFDP Finished PDU.*
- typedef struct **CF\_CFDP\_PduAck** **CF\_CFDP\_PduAck\_t**  
*Structure representing CFDP Acknowledge PDU.*
- typedef struct **CF\_CFDP\_SegmentRequest** **CF\_CFDP\_SegmentRequest\_t**  
*Structure representing CFDP Segment Request.*
- typedef struct **CF\_CFDP\_PduNak** **CF\_CFDP\_PduNak\_t**  
*Structure representing CFDP Non-Acknowledge PDU.*
- typedef struct **CF\_CFDP\_PduMd** **CF\_CFDP\_PduMd\_t**  
*Structure representing CFDP Metadata PDU.*
- typedef struct **CF\_CFDP\_PduFileDataHeader** **CF\_CFDP\_PduFileDataHeader\_t**  
*PDU file data header.*
- typedef struct **CF\_CFDP\_PduFileDialogContent** **CF\_CFDP\_PduFileDialogContent\_t**  
*PDU file data content typedef for limit checking outgoing\_file\_chunk\_size table value and set parameter command.*

## Enumerations

- enum `CF_CFDP_TlvType_t` {
   
`CF_CFDP_TLV_TYPE_FILESTORE_REQUEST` = 0 , `CF_CFDP_TLV_TYPE_FILESTORE_RESPONSE` = 1 ,
   
`CF_CFDP_TLV_TYPE_MESSAGE_TO_USER` = 2 , `CF_CFDP_TLV_TYPE_FAULT_HANDLER_OVERRIDE` = 4
   
 ,
   
`CF_CFDP_TLV_TYPE_FLOW_LABEL` = 5 , `CF_CFDP_TLV_TYPE_ENTITY_ID` = 6 , `CF_CFDP_TLV_TYPE_INVALID_MAX` = 7 }
   
*Values for "type" field of TLV structure.*
- enum `CF_CFDP_FileDirective_t` {
   
`CF_CFDP_FileDirective_INVALID_MIN` = 0 , `CF_CFDP_FileDirective_EOF` = 4 , `CF_CFDP_FileDirective_FIN` = 5 , `CF_CFDP_FileDirective_ACK` = 6 ,
   
`CF_CFDP_FileDirective_METADATA` = 7 , `CF_CFDP_FileDirective_NAK` = 8 , `CF_CFDP_FileDirective_PROMPT` = 9 , `CF_CFDP_FileDirective_KEEP_ALIVE` = 12 ,
   
`CF_CFDP_FileDirective_INVALID_MAX` = 13 }
   
*Values for "directive\_code" within CF\_CFDP\_PduFileDirectiveHeader\_t.*
- enum `CF_CFDP_AckTxnStatus_t` {
   
`CF_CFDP_AckTxnStatus_UNDEFINED` = 0 , `CF_CFDP_AckTxnStatus_ACTIVE` = 1 , `CF_CFDP_AckTxnStatus_TERMINATED` = 2 , `CF_CFDP_AckTxnStatus_UNRECOGNIZED` = 3 ,
   
`CF_CFDP_AckTxnStatus_INVALID` = 4 }
   
*Values for "acknowledgment transfer status".*
- enum `CF_CFDP_FinDeliveryCode_t` { `CF_CFDP_FinDeliveryCode_COMPLETE` = 0 , `CF_CFDP_FinDeliveryCode_INCOMPLETE` = 1 , `CF_CFDP_FinDeliveryCode_INVALID` = 2 }
   
*Values for "finished delivery code".*
- enum `CF_CFDP_FinFileStatus_t` {
   
`CF_CFDP_FinFileStatus_DISCARDED` = 0 , `CF_CFDP_FinFileStatus_DISCARDED_FILESTORE` = 1 ,
   
`CF_CFDP_FinFileStatus_RETAINED` = 2 , `CF_CFDP_FinFileStatus_UNREPORTED` = 3 ,
   
`CF_CFDP_FinFileStatus_INVALID` = 4 }
   
*Values for "finished file status".*
- enum `CF_CFDP_ConditionCode_t` {
   
`CF_CFDP_ConditionCode_NO_ERROR` = 0 , `CF_CFDP_ConditionCode_POS_ACK_LIMIT_REACHED` = 1 ,
   
`CF_CFDP_ConditionCode_KEEP_ALIVE_LIMIT_REACHED` = 2 , `CF_CFDP_ConditionCode_INVALID_TRANSMISSION_MODE` = 3 ,
   
`CF_CFDP_ConditionCode_FILESTORE_REJECTION` = 4 , `CF_CFDP_ConditionCode_FILE_CHECKSUM_FAILURE` = 5 ,
   
`CF_CFDP_ConditionCode_FILE_SIZE_ERROR` = 6 , `CF_CFDP_ConditionCode_NAK_LIMIT_REACHED` = 7 ,
   
`CF_CFDP_ConditionCode_INACTIVITY_DETECTED` = 8 , `CF_CFDP_ConditionCode_INVALID_FILE_STRUCTURE` = 9 ,
   
`CF_CFDP_ConditionCode_CHECK_LIMIT_REACHED` = 10 , `CF_CFDP_ConditionCode_UNSUPPORTED_CHECKSUM_TYPE` = 11 ,
   
`CF_CFDP_ConditionCode_SUSPEND_REQUEST_RECEIVED` = 14 , `CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED` = 15 }
   
*Values for "condition code".*

### 12.38.1 Detailed Description

Structures defining to CFDP PDUs

Note that structures and enumerations defined in this file with a CF\_CFDP prefix are defined according to the CCSDS CFDP specification (727.0-B-5). These values must match the specification for that structure/field, they are not locally changeable.

**Note**

Many of the structures defined in this file are variably-sized when encoded for network transmission. As a result, C structures used to map to these structures are of limited usefulness, generally only capable of describing the first element(s) where offsets are fixed. A marker member is utilized to indicate where the fixed data ends and variable length data begins. At some point, the structures in this file should change to encode/decode functions.

### 12.38.2 Macro Definition Documentation

**12.38.2.1 CF\_APP\_MAX\_HEADER\_SIZE** #define CF\_APP\_MAX\_HEADER\_SIZE (sizeof(CF\_CFDP\_PduHeader\_t) + sizeof(CF\_TransactionSeq\_t) + (3 \* sizeof(CF\_EntityId\_t)))

Maximum encoded size of a CFDP PDU that this implementation can accept.

This definition reflects the current configuration of the CF application. Note that this is based on the size of the native representation of Entity ID and sequence number. Although the bitwise representations of these items are different in the encoded packets vs. the native representation, the basic size still correlates (e.g. if it takes 4 bytes natively, it will be encoded into 4 bytes).

Definition at line 75 of file cf\_cfdp\_pdu.h.

**12.38.2.2 CF\_CFDP\_MAX\_HEADER\_SIZE** #define CF\_CFDP\_MAX\_HEADER\_SIZE (sizeof(CF\_CFDP\_PduHeader\_t) + (3 \* sizeof(CF\_CFDP\_uint64\_t))) /\* 8 bytes for each variable item \*/

Maximum encoded size of a CFDP PDU header.

Per the blue book, the size of the Entity ID and Sequence Number may be up to 8 bytes. CF is configurable in what it can accept and transmit, which may be smaller than what the blue book permits.

Definition at line 54 of file cf\_cfdp\_pdu.h.

**12.38.2.3 CF\_CFDP\_MIN\_HEADER\_SIZE** #define CF\_CFDP\_MIN\_HEADER\_SIZE (sizeof(CF\_CFDP\_PduHeader\_t) + (3 \* sizeof(CF\_CFDP\_uint8\_t))) /\* 1 byte for each variable item \*/

Minimum encoded size of a CFDP PDU header.

Per the blue book, the size of the Entity ID and Sequence Number must be at least 1 byte.

Definition at line 62 of file cf\_cfdp\_pdu.h.

### 12.38.3 Typedef Documentation

**12.38.3.1 CF\_CFDP\_lv\_t** typedef struct CF\_CFDP\_lv CF\_CFDP\_lv\_t

Structure representing CFDP LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

Defined per table 5-2 of CCSDS 727.0-B-5

**12.38.3.2 CF\_CFDP\_PduAck\_t** typedef struct CF\_CFDP\_PduAck CF\_CFDP\_PduAck\_t

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

**12.38.3.3 CF\_CFDP\_PduEof\_t** typedef struct CF\_CFDP\_PduEof CF\_CFDP\_PduEof\_t

Structure representing CFDP End of file PDU.

Defined per section 5.2.2 / table 5-6 of CCSDS 727.0-B-5

**12.38.3.4 CF\_CFDP\_PduFileDialogContent\_t** `typedef struct CF_CFDP_PduFileDialogContent CF_CFDP_PduFileDialogContent_t`  
 PDU file data content typedef for limit checking outgoing\_file\_chunk\_size table value and set parameter command.  
 This definition allows for the largest data block possible, as CF\_MAX\_PDU\_SIZE - the minimum possible header size.  
 In practice the outgoing file chunk size is limited by whichever is smaller; the remaining data, remaining space in the packet, and outgoing\_file\_chunk\_size.

**12.38.3.5 CF\_CFDP\_PduFileDataHeader\_t** `typedef struct CF_CFDP_PduFileDialogHeader CF_CFDP_PduFileDialogHeader_t`  
 PDU file data header.

**12.38.3.6 CF\_CFDP\_PduFileDirectiveHeader\_t** `typedef struct CF_CFDP_PduFileDirectiveHeader CF_CFDP_PduFileDirectiveHeader_t`  
 Structure representing CFDP File Directive Header.  
 Defined per section 5.2 of CCSDS 727.0-B-5

**12.38.3.7 CF\_CFDP\_PduFin\_t** `typedef struct CF_CFDP_PduFin CF_CFDP_PduFin_t`  
 Structure representing CFDP Finished PDU.  
 Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

**12.38.3.8 CF\_CFDP\_PduHeader\_t** `typedef struct CF_CFDP_PduHeader CF_CFDP_PduHeader_t`  
 Structure representing base CFDP PDU header.  
 This header appears at the beginning of all CFDP PDUs, of all types. Note that the header is variable length, it also contains source and destination entity IDs, and the transaction sequence number.  
 Defined per section 5.1 of CCSDS 727.0-B-5

#### Note

this contains variable length data for the EID+TSN, which is *not* included in this definition. As a result, the sizeof(`CF_CFDP_PduHeader_t`) reflects only the size of the fixed fields. Use CF\_HeaderSize() to get the actual size of this structure.

**12.38.3.9 CF\_CFDP\_PduMd\_t** `typedef struct CF_CFDP_PduMd CF_CFDP_PduMd_t`  
 Structure representing CFDP Metadata PDU.  
 Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5

**12.38.3.10 CF\_CFDP\_PduNak\_t** `typedef struct CF_CFDP_PduNak CF_CFDP_PduNak_t`  
 Structure representing CFDP Non-Acknowledge PDU.  
 Defined per section 5.2.6 / table 5-10 of CCSDS 727.0-B-5

**12.38.3.11 CF\_CFDP\_SegmentRequest\_t** `typedef struct CF_CFDP_SegmentRequest CF_CFDP_SegmentRequest_t`  
 Structure representing CFDP Segment Request.  
 Defined per section 5.2.6 / table 5-11 of CCSDS 727.0-B-5

**12.38.3.12 CF\_CFDP\_tlv\_t** `typedef struct CF_CFDP_tlv CF_CFDP_tlv_t`  
 Structure representing CFDP TLV Object format.  
 These Type + Length + Value pairs used in several CFDP PDU types, typically for file storage requests (section 5.4).  
 Defined per table 5-3 of CCSDS 727.0-B-5

## 12.38.4 Enumeration Type Documentation

**12.38.4.1 CF\_CFDP\_AckTxnStatus\_t enum CF\_CFDP\_AckTxnStatus\_t**

Values for "acknowledgment transfer status".

This enum is pertinent to the ACK PDU type, defines the values for the directive field.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_AckTxnStatus_UNDEFINED	
CF_CFDP_AckTxnStatus_ACTIVE	
CF_CFDP_AckTxnStatus_TERMINATED	
CF_CFDP_AckTxnStatus_UNRECOGNIZED	
CF_CFDP_AckTxnStatus_INVALID	

Definition at line 225 of file cf\_cfdp\_pdu.h.

**12.38.4.2 CF\_CFDP\_ConditionCode\_t enum CF\_CFDP\_ConditionCode\_t**

Values for "condition code".

This enum defines the values for the condition code field for the PDU types which have this field (EOF, FIN, ACK)

Defined per table 5-5 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_ConditionCode_NO_ERROR	
CF_CFDP_ConditionCode_POS_ACK_LIMIT_REACHED	
CF_CFDP_ConditionCode_KEEP_ALIVE_LIMIT_REACHED	
CF_CFDP_ConditionCode_INVALID_TRANSMISSION_MODE	
CF_CFDP_ConditionCode_FILESTORE_REJECTION	
CF_CFDP_ConditionCode_FILE_CHECKSUM_FAILURE	
CF_CFDP_ConditionCode_FILE_SIZE_ERROR	
CF_CFDP_ConditionCode_NAK_LIMIT_REACHED	
CF_CFDP_ConditionCode_INACTIVITY_DETECTED	
CF_CFDP_ConditionCode_INVALID_FILE_STRUCTURE	
CF_CFDP_ConditionCode_CHECK_LIMIT_REACHED	
CF_CFDP_ConditionCode_UNSUPPORTED_CHECKSUM_TYPE	
CF_CFDP_ConditionCode_SUSPEND_REQUEST_RECEIVED	
CF_CFDP_ConditionCode_CANCEL_REQUEST_RECEIVED	

Definition at line 274 of file cf\_cfdp\_pdu.h.

**12.38.4.3 CF\_CFDP\_FileDirective\_t enum CF\_CFDP\_FileDirective\_t**

Values for "directive\_code" within CF\_CFDP\_PduFileDirectiveHeader\_t.

Defined per table 5-4 of CCSDS 727.0-B-5

Enumerator

CF_CFDP_FileDirective_INVALID_MIN	Minimum used to limit range.
CF_CFDP_FileDirective_EOF	
CF_CFDP_FileDirective_FIN	
CF_CFDP_FileDirective_ACK	

## Enumerator

CF_CFDP_FileDirective_METADATA	
CF_CFDP_FileDirective_NAK	
CF_CFDP_FileDirective_PROMPT	
CF_CFDP_FileDirective_KEEP_ALIVE	
CF_CFDP_FileDirective_INVALID_MAX	Maximum used to limit range.

Definition at line 204 of file cf\_cfdp\_pdu.h.

**12.38.4.4 CF\_CFDP\_FinDeliveryCode\_t enum CF\_CFDP\_FinDeliveryCode\_t**

Values for "finished delivery code".

This enum is pertinent to the FIN PDU type, defines the values for the delivery code field.

Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

## Enumerator

CF_CFDP_FinDeliveryCode_COMPLETE	
CF_CFDP_FinDeliveryCode_INCOMPLETE	
CF_CFDP_FinDeliveryCode_INVALID	

Definition at line 242 of file cf\_cfdp\_pdu.h.

**12.38.4.5 CF\_CFDP\_FinFileStatus\_t enum CF\_CFDP\_FinFileStatus\_t**

Values for "finished file status".

This enum is pertinent to the FIN PDU type, defines the values for the file status field.

Defined per section 5.2.3 / table 5-7 of CCSDS 727.0-B-5

## Enumerator

CF_CFDP_FinFileStatus_DISCARDED	
CF_CFDP_FinFileStatus_DISCARDED_FILESTORE	
CF_CFDP_FinFileStatus_RETAINED	
CF_CFDP_FinFileStatus_UNREPORTED	
CF_CFDP_FinFileStatus_INVALID	

Definition at line 257 of file cf\_cfdp\_pdu.h.

**12.38.4.6 CF\_CFDP\_TlvType\_t enum CF\_CFDP\_TlvType\_t**

Values for "type" field of TLV structure.

Defined per section 5.4 of CCSDS 727.0-B-5

## Enumerator

CF_CFDP_TLV_TYPE_FILESTORE_REQUEST	
CF_CFDP_TLV_TYPE_FILESTORE_RESPONSE	
CF_CFDP_TLV_TYPE_MESSAGE_TO_USER	
CF_CFDP_TLV_TYPE_FAULT_HANDLER_OVERRIDE	

**Enumerator**

CF_CFDP_TLV_TYPE_FLOW_LABEL	
CF_CFDP_TLV_TYPE_ENTITY_ID	
CF_CFDP_TLV_TYPE_INVALID_MAX	

Definition at line 188 of file cf\_cfdp\_pdu.h.

## 12.39 apps/cf/fsw/src(cf\_cfdp\_r.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_r.h"
#include "cf_cfdp_dispatch.h"
#include <stdio.h>
#include <string.h>
#include "cf_assert.h"
```

### Functions

- **CFE\_Status\_t CF\_CFDP\_R\_CheckCrc (CF\_Transaction\_t \*txn)**  
*Checks that the transaction file's CRC matches expected.*
- **bool CF\_CFDP\_R\_CheckComplete (CF\_Transaction\_t \*txn)**  
*Checks R transaction state for transaction completion status.*
- **CFE\_Status\_t CF\_CFDP\_R\_ProcessFd (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph)**  
*Process a filedata PDU on a transaction.*
- **void CF\_CFDP\_R\_SubstateRecvEof (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph)**  
*Processing receive EOF common functionality for R1/R2.*
- **void CF\_CFDP\_R\_SubstateRecvFileData (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph)**  
*Process received file data for R.*
- **void CF\_CFDP\_R2\_GapCompute (const CF\_ChunkList\_t \*chunks, const CF\_Chunk\_t \*chunk, void \*opaque)**  
*Loads a single NAK segment request.*
- **CFE\_Status\_t CF\_CFDP\_R\_SendNak (CF\_Transaction\_t \*txn)**  
*Send a NAK PDU for R2.*
- **void CF\_CFDP\_R\_Init (CF\_Transaction\_t \*txn)**  
*Initialize a transaction structure for R.*
- **void CF\_CFDP\_R\_CalcCrcStart (CF\_Transaction\_t \*txn)**  
*Begin calculation of the file CRC.*
- **void CF\_CFDP\_R\_CalcCrcChunk (CF\_Transaction\_t \*txn)**  
*Calculate up to the configured amount of bytes of CRC.*
- **void CF\_CFDP\_R2\_SubstateRecvFinAck (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph)**  
*Process receive FIN-ACK PDU.*
- **void CF\_CFDP\_R\_SubstateRecvMd (CF\_Transaction\_t \*txn, CF\_Logical\_PduBuffer\_t \*ph)**  
*Substate function to receive an MD.*

- void `CF_CFDP_R1_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*R1 receive PDU processing.*
- void `CF_CFDP_R2_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*R2 receive PDU processing.*
- void `CF_CFDP_R_HandleFileRetention (CF_Transaction_t *txn)`  
*Remove/Move file after transaction.*
- void `CF_CFDP_R_AckTimerTick (CF_Transaction_t *txn)`  
*Perform acknowledgement timer tick (time-based) processing for R transactions.*
- `CF_RxSubState_t CF_CFDP_R_CheckState_DATA_NORMAL (CF_Transaction_t *txn)`
- `CF_RxSubState_t CF_CFDP_R_CheckState_DATA_EOF (CF_Transaction_t *txn)`
- `CF_RxSubState_t CF_CFDP_R_CheckState_VALIDATE (CF_Transaction_t *txn)`
- `CF_RxSubState_t CF_CFDP_R_CheckState_FILESTORE (CF_Transaction_t *txn)`
- `CF_RxSubState_t CF_CFDP_R_CheckState_FINACK (CF_Transaction_t *txn)`
- void `CF_CFDP_R_CheckState (CF_Transaction_t *txn)`  
*Run RX state machine for the transaction.*
- void `CF_CFDP_R_Tick_Maintenance (CF_Transaction_t *txn)`  
*Generate protocol state PDUs as needed.*
- void `CF_CFDP_R_Tick (CF_Transaction_t *txn)`  
*Perform tick (time-based) processing for R transactions.*

### 12.39.1 Detailed Description

The CF Application CFDP receive logic source file  
Handles all CFDP engine functionality specific to RX transactions.

### 12.39.2 Function Documentation

**12.39.2.1 CF\_CFDP\_R1\_Recv()** `void CF_CFDP_R1_Recv (`  
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph )`

R1 receive PDU processing.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

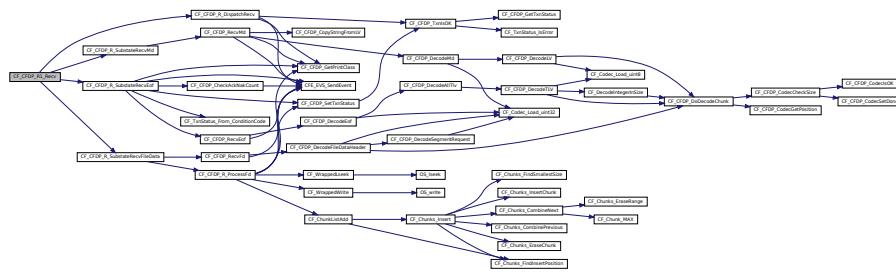
#### Parameters

<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to the PDU information

Definition at line 572 of file cf\_cfdp\_r.c.

References CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_FileDirective\_METADATA, CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_SubstateRecvFileData(), CF\_CFDP\_R\_SubstateRecvMd(), CF\_RxSubState\_DATA\_EOF, CF\_RxSubState\_DATA\_NORMAL, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, and CF\_CFDP\_R\_SubstateDispatchTable\_t::state.  
Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



```
12.39.2.2 CF_CFDP_R2_GapCompute() void CF_CFDP_R2_GapCompute (
```

const CF_ChunkList_t *	<i>chunks</i> ,
const CF_Chunk_t *	<i>chunk</i> ,
void *	<i>opaque</i> )

Loads a single NAK segment request.

## Description

This is a function callback from [CF\\_ChunkList\\_ComputeGaps\(\)](#).

## Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL, opaque must not be NULL.

### Parameters

<i>chunks</i>	Not used, required for compatibility with CF_ChunkList_ComputeGaps
<i>chunk</i>	Pointer to a single chunk information
<i>opaque</i>	Pointer to a <a href="#">CF_GapComputeArgs_t</a> object (passed via CF_ChunkList_ComputeGaps)

Definition at line 248 of file cf\_cfdp\_r.c.

References CF\_Assert, CF\_PDU\_MAX\_SEGMENTS, CF\_GapComputeArgs\_t::nak, CF\_Logical\_SegmentList::num\_segments, CF\_Chunk::offset, CF\_Logical\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_start, CF\_Logical\_PduNak::scope\_start, CF\_Logical\_PduNak::segment\_list, CF\_Logical\_SegmentList::segments, and CF\_Chunk::size.

Referenced by CF CFDP R SendNak().

```
12.39.2.3 CF_CFDP_R2_Recv() void CF_CFDP_R2_Recv (
```

R2 receive PDU processing.

## **Assumptions, External Events, and Notes:**

txn must not be NULL.

## Parameters

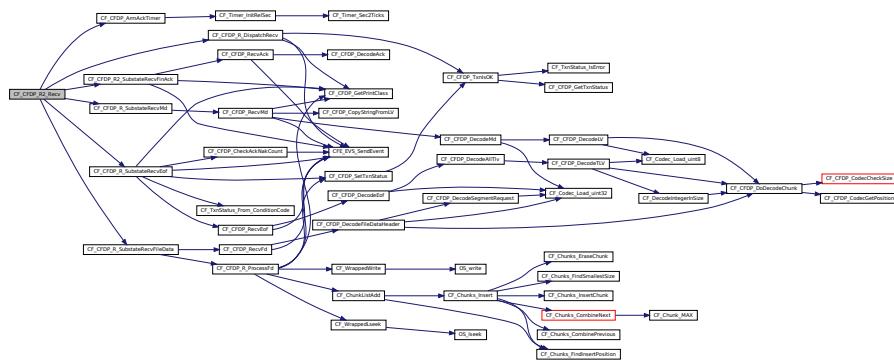
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 589 of file cf\_cfdp\_r.c.

References CF\_Flags\_Common::ack\_timer\_armed, CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_FileDirective\_ACK, CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_FileDirective\_METADATA, CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_SubstateRecvFileData(), CF\_CFDP\_R\_SubstateRecvMd(), CF\_RxSubState\_COMPLETE, CF\_RxSubState\_DATA\_EOF, CF\_RxSubState\_DATA\_NORMAL, CF\_RxSubState\_FILESTORE, CF\_RxSubState\_FINACK, CF\_RxSubState\_VALIDATE, CF\_StateFlags::com, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, CF\_Transaction::flags, and CF\_CFDP\_R\_SubstateDispatchTable\_t::state.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.39.2.4 CF\_CFDP\_R2\_SubstateRecvFinAck()** void CF\_CFDP\_R2\_SubstateRecvFinAck (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Process receive FIN-ACK PDU.

## Description

Receive and process a FIN-ACK PDU

## Assumptions, External Events, and Notes:

txnid must not be NULL. ph must not be NULL.

## Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

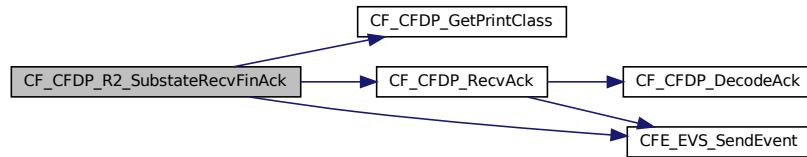
Definition at line 511 of file cf\_cfdp\_r.c.

References CF\_Logical\_IntHeader::ack, CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::cc, CF\_AppData, CF\_CFDP\_FileDirective FIN, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_PDU FINACK ERR EID, CF\_

\_CFDP\_RecvAck(), CF\_TRACE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF←\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_HkRecv::error, CF\_Flags\_Rx::finack\_recv, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_Pdu←Buffer::int\_header, CF\_HkPacket::Payload, CF\_HkCounters::recv, CF\_StateFlags::rx, CF\_History::seq\_num, CF←History::src\_eid, and CF\_Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_R2\_Recv().

Here is the call graph for this function:



**12.39.2.5 CF\_CFDP\_R\_AckTimerTick()** void CF\_CFDP\_R\_AckTimerTick ( CF\_Transaction\_t \* txn )

Perform acknowledgement timer tick (time-based) processing for R transactions.

#### Description

This is invoked as part of overall timer tick processing if the transaction has some sort of acknowledgement pending from the remote.

#### Assumptions, External Events, and Notes:

txn must not be NULL

#### Parameters

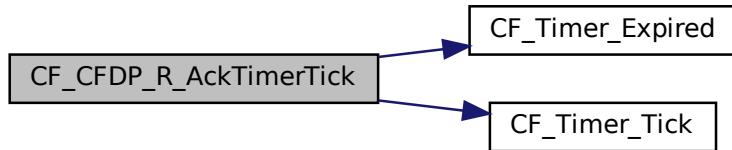
txn	Pointer to the transaction object
-----	-----------------------------------

Definition at line 756 of file cf\_cfdp\_r.c.

References CF\_Transaction::ack\_timer, CF\_Flags\_Common::ack\_timer\_armed, CF\_Timer\_Expired(), CF\_Timer\_Tick(), CF\_StateFlags::com, CF\_Transaction::flags, and CF\_Transaction::reliable\_mode.

Referenced by CF\_CFDP\_R\_Tick().

Here is the call graph for this function:



**12.39.2.6 CF\_CFDP\_R\_CalcCrcChunk()** void CF\_CFDP\_R\_CalcCrcChunk ( CF\_Transaction\_t \* txn )

Calculate up to the configured amount of bytes of CRC.

#### Description

The configuration table has a number of bytes to calculate per transaction per wakeup. At each wakeup, the file is read and this number of bytes are calculated. This function will set the checksum error condition code if the final CRC does not match.

#### PTFO

Increase throughput by consuming all CRC bytes per wakeup in transaction-order. This would require a change to the meaning of the value in the configuration table.

#### Assumptions, External Events, and Notes:

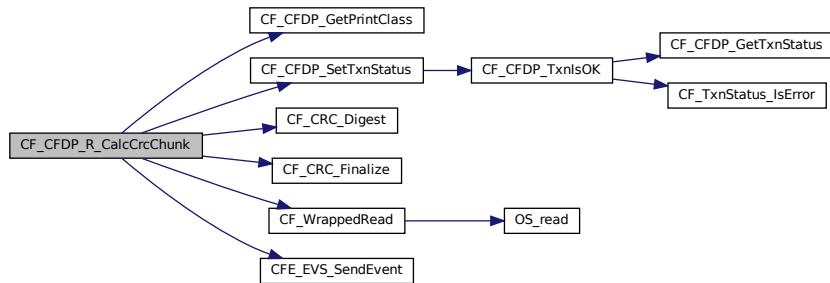
txn must not be NULL.

Definition at line 443 of file cf\_cfdp\_r.c.

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_READ\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_CRC\_Digest(), CF\_CRC\_Finalize(), CF\_R2\_CRC\_CHUNK\_SIZE, CF\_Txn\_Status\_FILE\_SIZE\_ERROR, CF\_WrappedRead(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_Flags\_Common::crc\_complete, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_read, CF\_Transaction::flags, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R\_CheckState\_VALIDATE().

Here is the call graph for this function:



#### 12.39.2.7 CF\_CFDP\_R\_CalcCrcStart() void CF\_CFDP\_R\_CalcCrcStart ( CF\_Transaction\_t \* txn )

Begin calculation of the file CRC.

##### Description

Seeks back to the beginning of the file and initializes the CRC

#### Assumptions, External Events, and Notes:

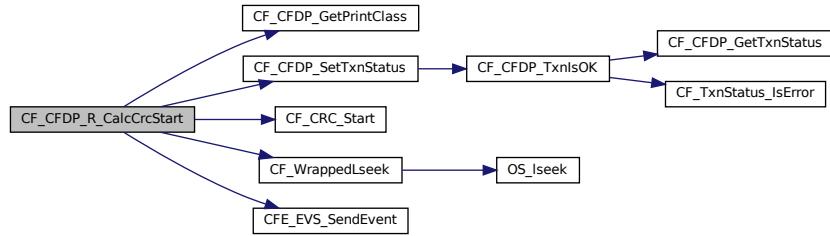
txn must not be NULL.

Definition at line 401 of file cf\_cfdp\_r.c.

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_SEEK\_CRC\_ERR\_EID, CF\_CFDP\_R\_SIZE\_MISMATCH\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_CRC\_Start(), CF\_Txn\_Status\_FILE\_SIZE\_ERROR, CF\_WrappedLseek(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_Flags\_Common::crc\_complete, CF\_StateData::eof\_size, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_seek, CF\_Transaction::flags, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, OS\_SEEK\_SET, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R\_CheckState().

Here is the call graph for this function:



### 12.39.2.8 CF\_CFDP\_R\_CheckComplete()

```
bool CF_CFDP_R_CheckComplete (
    CF_Transaction_t * txn )
```

Checks R transaction state for transaction completion status.

#### Description

This function is called anywhere there's a desire to know if the transaction has completed. It may trigger other actions by setting flags to be handled during tick processing. In order for a transaction to be complete, it must have had its meta-data PDU received, the EOF must have been received, and there must be no gaps in the file. CRC is not checked in this function, because it's only called from functions after EOF is received.

#### Assumptions, External Events, and Notes:

`txn` must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

#### Returns

boolean indicating if the file is complete or not

Definition at line 75 of file `cf_cfdp_r.c`.

References `CF_ChunkList_ComputeGaps()`, `CF_ChunkWrapper::chunks`, `CF_Transaction::chunks`, `CF_Flags_Rx::eof_count`, `CF_Transaction::flags`, `CF_Transaction::fsize`, `CF_Flags_Rx::md_recv`, `CF_StateFlags::rx`, and `CF_Flags_Rx::send_nak`.

Referenced by `CF_CFDP_R_CheckState_DATA_EOF()`.

Here is the call graph for this function:



**12.39.2.9 CF\_CFDP\_R\_CheckCrc()** `CFE_Status_t CF_CFDP_R_CheckCrc ( CF_Transaction_t * txn )`

Checks that the transaction file's CRC matches expected.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Return values**

<code>CFE_SUCCESS</code>	on CRC match, otherwise <code>CF_ERROR</code> .
--------------------------	---

**Parameters**

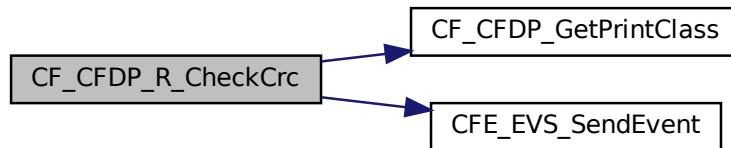
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 47 of file cf\_cfdp\_r.c.

References `CF_AppData`, `CF_CFDP_GetPrintClass()`, `CF_CFDP_R_CRC_ERR_EID`, `CF_ERROR`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_HkChannel_Data::counters`, `CF_Transaction::crc`, `CF_HkFault::crc_mismatch`, `CF_StateData::eof::crc`, `CF_HkCounters::fault`, `CF_Transaction::history`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, `CF_Crc::result`, `CF_History::seq_num`, `CF_History::src_eid`, and `CF_Transaction::state_data`.

Referenced by `CF_CFDP_R_HandleFileRetention()`.

Here is the call graph for this function:



**12.39.2.10 CF\_CFDP\_R\_CheckState()** `void CF_CFDP_R_CheckState ( CF_Transaction_t * txn )`

Run RX state machine for the transaction.

**Description**

Checks flags on the transaction and execute state transitions

**Assumptions, External Events, and Notes:**

txn must not be NULL.

## Parameters

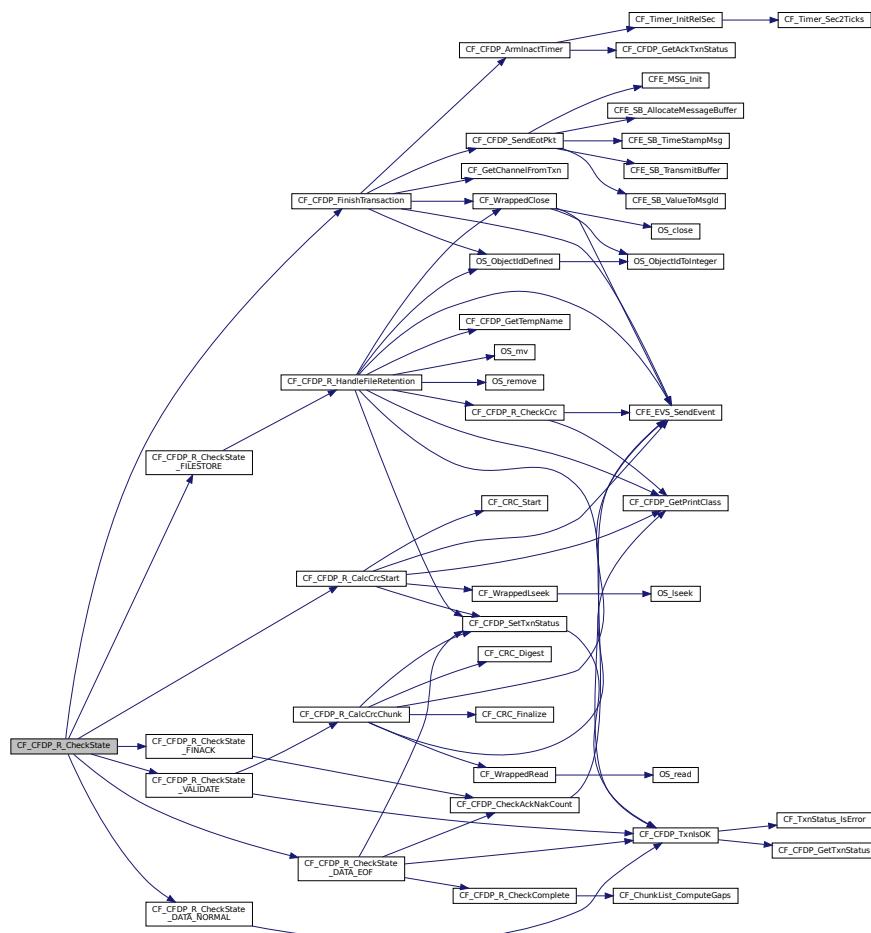
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 933 of file cf\_cfdp\_r.c.

References `CF_Flags_Common::ack_timer_armed`, `CF_StateData::acknak_count`, `CF_CFDP_FinishTransaction()`, `CF_CFDP_R_CalcCrcStart()`, `CF_CFDP_R_CheckState_DATA_EOF()`, `CF_CFDP_R_CheckState_NORMAL()`, `CF_CFDP_R_CheckState_FILESTORE()`, `CF_CFDP_R_CheckState_FINACK()`, `CF_CFDP_R_CheckState_VALIDATE()`, `CF_RxSubState_COMPLETE`, `CF_RxSubState_DATA_EOF`, `CF_RxSubState_DATA_NORMAL`, `CF_RxSubState_FILESTORE`, `CF_RxSubState_FINACK`, `CF_RxSubState_VALIDATE`, `CF_TRACE`, `CF_StateFlags::com`, `CF_Transaction::flags`, `CF_Transaction::reliable_mode`, `CF_StateFlags::rx`, `CF_Flags_Rx::send_fin`, `CF_Flags_Rx::send_nak`, `CF_Transaction::state_data`, and `CF_StateData::sub_state`.

Referenced by `CF_CFDP_R_Tick()`.

Here is the call graph for this function:



### 12.39.2.11 CF\_CFDP\_R\_CheckState\_DATA\_EOF()

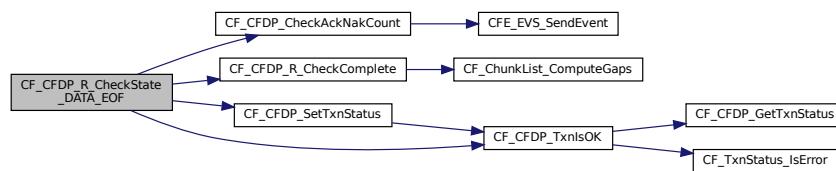
```
CF_RxSubState_t CF_CFDP_R_CheckState_DATA_EOF (
    CF_Transaction_t * txn )
```

Definition at line 806 of file cf\_cfdp\_r.c.

References CF\_Flags\_Common::ack\_timer\_armed, CF\_StateData::acknak\_count, CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_R\_CheckComplete(), CF\_CFDP\_SetTxnStatus(), CF\_CFDP\_TxnIsOK(), CF\_RxSubState\_FILESTORE, CF\_RxSubState\_VALIDATE, CF\_TxnStatus\_NAK\_LIMIT\_REACHED, CF\_StateFlags::com, CF\_Transaction::flags, CF\_Flags\_Common::is\_complete, CF\_Transaction::reliable\_mode, CF\_StateFlags::rx, CF\_Flags\_Rx::send\_nak, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_R\_CheckState().

Here is the call graph for this function:



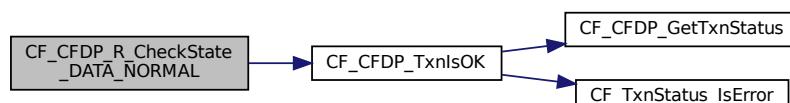
**12.39.2.12 CF\_CFDP\_R\_CheckState\_DATA\_NORMAL()** `CF_RxSubState_t CF_CFDP_R_CheckState_DATA_NORMAL ( CF_Transaction_t * txn )`

Definition at line 781 of file cf\_cfdp\_r.c.

References CF\_CFDP\_TxnIsOK(), CF\_RxSubState\_DATA\_EOF, CF\_RxSubState\_FILESTORE, CF\_Flags\_Rx::eof\_count, CF\_Transaction::flags, CF\_StateFlags::rx, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_R\_CheckState().

Here is the call graph for this function:



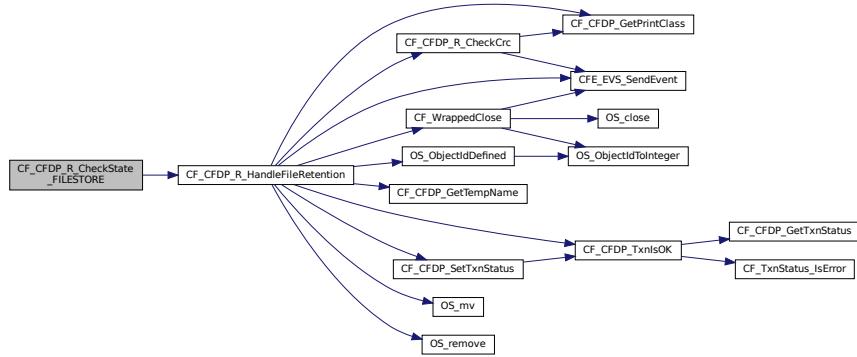
**12.39.2.13 CF\_CFDP\_R\_CheckState\_FILESTORE()** `CF_RxSubState_t CF_CFDP_R_CheckState_FILESTORE ( CF_Transaction_t * txn )`

Definition at line 870 of file cf\_cfdp\_r.c.

References CF\_CFDP\_R\_HandleFileRetention(), CF\_RxSubState\_COMPLETE, CF\_RxSubState\_FINACK, CF\_Flags\_Common::close\_req, CF\_StateFlags::com, CF\_Transaction::flags, CF\_Transaction::reliable\_mode, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_R\_CheckState().

Here is the call graph for this function:



#### 12.39.2.14 CF\_CFDP\_R\_CheckState\_FINACK() `CF_RxSubState_t CF_CFDP_R_CheckState_FINACK ( CF_Transaction_t * txn )`

Definition at line 896 of file cf\_cfdp\_r.c.

References `CF_Flags_Common::ack_timer_armed`, `CF_StateData::acknak_count`, `CF_CFDP_CheckAckNakCount()`, `CF_RxSubState_COMPLETE`, `CF_StateFlags::com`, `CF_Flags_Rx::finack_recv`, `CF_Transaction::flags`, `CF_Flags__Common::inactivity_fired`, `CF_Transaction::reliable_mode`, `CF_StateFlags::rx`, `CF_Flags_Rx::send_fin`, `CF_Transaction::state_data`, and `CF_StateData::sub_state`.

Referenced by `CF_CFDP_R_CheckState()`.

Here is the call graph for this function:



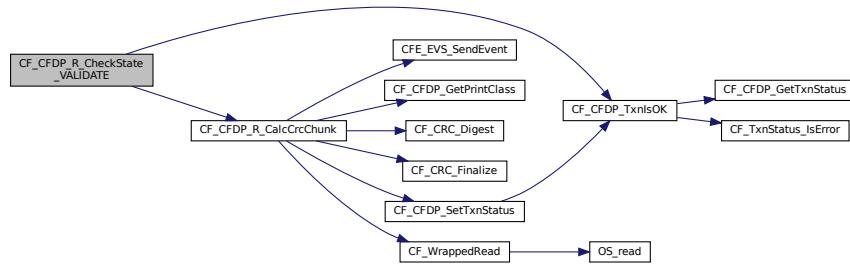
#### 12.39.2.15 CF\_CFDP\_R\_CheckState\_VALIDATE() `CF_RxSubState_t CF_CFDP_R_CheckState_VALIDATE ( CF_Transaction_t * txn )`

Definition at line 848 of file cf\_cfdp\_r.c.

References `CF_StateData::cached_pos`, `CF_CFDP_R_CalcCrcChunk()`, `CF_CFDP_TxnIsOK()`, `CF_RxSubState__FILESTORE`, `CF_Transaction::fsize`, `CF_Transaction::state_data`, and `CF_StateData::sub_state`.

Referenced by `CF_CFDP_R_CheckState()`.

Here is the call graph for this function:



### 12.39.2.16 CF\_CFDP\_R\_HandleFileRetention()

```
void CF_CFDP_R_HandleFileRetention (
    CF_Transaction_t * txn )
```

Remove/Move file after transaction.

Determines disposition of local file after file transfer completion.

For a receiver:

- If the transfer was successful then the temp file is moved into the final location under the indicated name from the MD PDU.
- If the file transfer is unsuccessful then the temp file is deleted.

**Assumptions, External Events, and Notes:**

**Parameters**

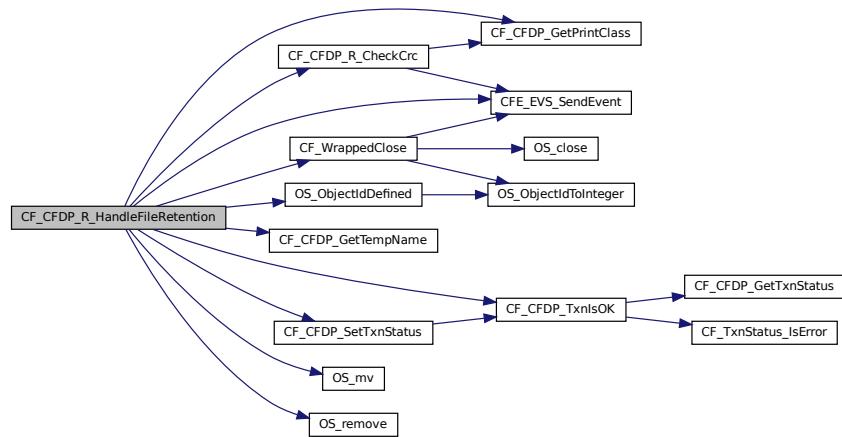
<code>txn</code>	Transaction object pointer
------------------	----------------------------

Definition at line 631 of file cf\_cfdp\_r.c.

References `CF_CFDP_FinDeliveryCode_COMPLETE`, `CF_CFDP_FinDeliveryCode_INCOMPLETE`, `CF_CFDP_FinDeliveryCode_INVALID`, `CF_CFDP_FinFileStatus_DISCARDED`, `CF_CFDP_FinFileStatus_DISCARDED_FILESTORE`, `CF_CFDP_FinFileStatus_INVALID`, `CF_CFDP_FinFileStatus_RETAINED`, `CF_CFDP_GetPrintClass()`, `CF_CFDP_GetTempName()`, `CF_CFDP_R_CheckCrc()`, `CF_CFDP_R_FILE_RETAINED_EID`, `CF_CFDP_R_NOT_RETAINED_EID`, `CF_CFDP_R_RENAME_ERR_EID`, `CF_CFDP_SetTxnStatus()`, `CF_CFDP_TxnIsOK()`, `CF_Txn_Status_FILE_CHECKSUM_FAILURE`, `CF_TxnStatus_FILESTORE_REJECTION`, `CF_TxnStatus_INVALID_FILE_STRUCTURE`, `CF_TxnStatus_NO_ERROR`, `CF_WrappedClose()`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_MISSION_MAX_PATH_LEN`, `CFE_SUCCESS`, `CF_StateFlags::com`, `CF_TxnFilenames::dst_filename`, `CF_Transaction::fd`, `CF_StateData::fin_dc`, `CF_StateData::fin_fs`, `CF_Transaction::flags`, `CF_History::fnames`, `CF_Transaction::history`, `CF_Flags_Common::is_complete`, `OS_mv()`, `OS_OBJECT_ID_UNDEFINED`, `OS_ObjectIdDefined()`, `OS_remove()`, `OS_SUCCESS`, `CF_StateFlags::rx`, `CF_History::seq_num`, `CF_History::src_eid`, `CF_Transaction::state_data`, `CF_Flags_Rx::tempfile_created`, and `CF_History::txn_stat`.

Referenced by `CF_CFDP_R_CheckState_FILESTORE()`.

Here is the call graph for this function:



### 12.39.2.17 CF\_CFDP\_R\_Init()

```
void CF_CFDP_R_Init (
    CF_Transaction_t * txn )
```

Initialize a transaction structure for R.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

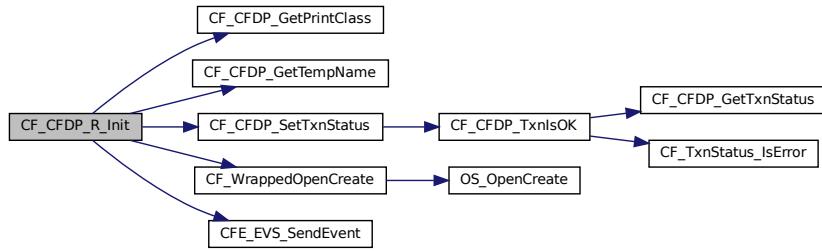
#### Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 360 of file cf\_cfdp\_r.c.

References CF\_AppData, CF\_CFDP\_FinDeliveryCode\_INVALID, CF\_CFDP\_FinFileStatus\_INVALID, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_GetTempName(), CF\_CFDP\_R\_CREAT\_ERR\_EID, CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID, CF\_CFDP\_SetTxnStatus(), CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_WrappedOpenCreate(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_MISSION\_MAX\_FILE\_LEN, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_open, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, OS\_FILE\_FLAG\_CREATE, OS\_FILE\_FLAG\_TRUNCATE, OS\_OBJECT\_ID\_UNDEFINED, OS\_READ\_WRITE, CF\_HkPacket::Payload, CF\_StateFlags::rx, CF\_History::seq\_num, CF\_History::src\_eid, CF\_Transaction::state\_data, and CF\_Flags\_Rx::tempfile\_created. Referenced by CF\_CFDP\_SetupRxTransaction().

Here is the call graph for this function:



### 12.39.2.18 CF\_CFDP\_R\_ProcessFd() [CFE\\_Status\\_t](#) CF\_CFDP\_R\_ProcessFd (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

Process a filedata PDU on a transaction.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Return values**

<a href="#">CFE_SUCCESS</a>	on success. CF_ERROR on error.
-----------------------------	--------------------------------

**Parameters**

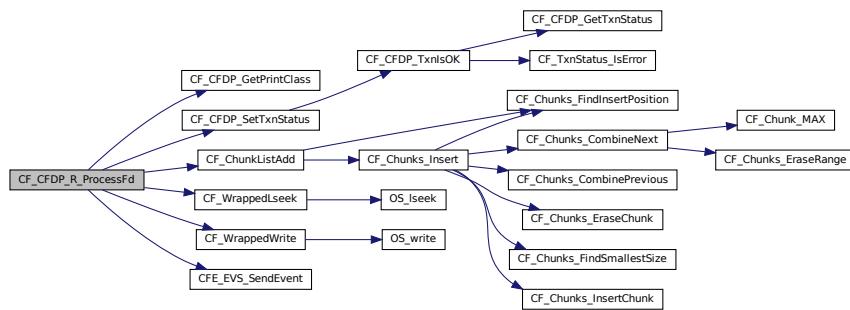
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 106 of file cf\_cfdp\_r.c.

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID, CF\_CFDP\_R\_WRITE\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_ChunkListAdd(), CF\_ERROR, CF\_Txn\_Status\_FILE\_SIZE\_ERROR, CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_WrappedLseek(), CF\_WrappedWrite(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_ChunkWrapper::chunks, CF\_Transaction::chunks, CF\_HkChannel\_Data::counters, CF\_Logical\_PduFileDataHeader::data\_len, CF\_Logical\_PduFileDataHeader::data\_ptr, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_Logical\_IntHeader::fd, CF\_HkRecv::file\_data\_bytes, CF\_HkFault::file\_seek, CF\_HkFault::file\_write, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_PduFileDataHeader::offset, OS\_SEEK\_SET, CF\_HkPacket::Payload, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R\_SubstateRecvFileData().

Here is the call graph for this function:



### 12.39.2.19 CF\_CFDP\_R\_SendNak() [CFE\\_Status\\_t](#) CF\_CFDP\_R\_SendNak ( [CF\\_Transaction\\_t](#) \* *txn* )

Send a NAK PDU for R2.

#### Description

NAK PDU is sent when there are gaps in the received data. The chunks class tracks this and generates the NAK PDU by calculating gaps internally and calling [CF\\_CFDP\\_R2\\_GapCompute\(\)](#). There is a special case where if a metadata PDU has not been received, then a NAK packet will be sent to request another.

#### Assumptions, External Events, and Notes:

*txn* must not be NULL.

#### Return values

<a href="#">CFE_SUCCESS</a>	on success. <a href="#">CF_ERROR</a> on error.
-----------------------------	--

#### Parameters

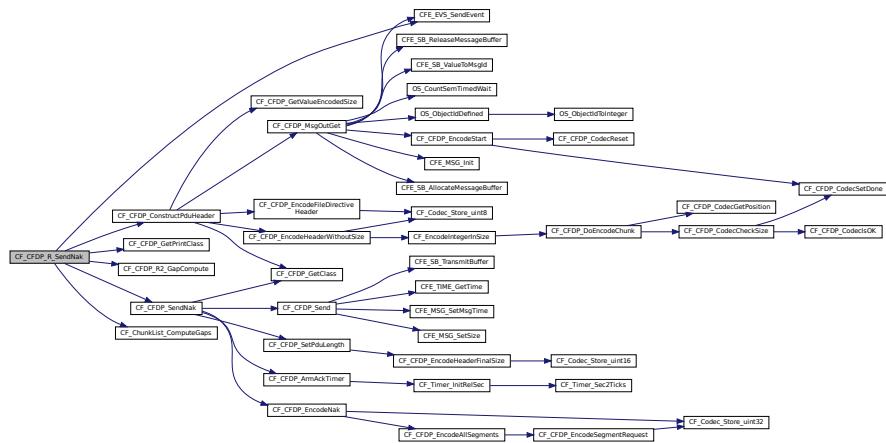
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 280 of file cf\_cfdp\_r.c.

References [CF\\_AppData](#), [CF\\_CFDP\\_ConstructPduHeader\(\)](#), [CF\\_CFDP\\_FileDirective\\_NAK](#), [CF\\_CFDP\\_GetPrintClass\(\)](#), [CF\\_CFDP\\_R2\\_GapCompute\(\)](#), [CF\\_CFDP\\_R\\_REQUEST\\_MD\\_INF\\_EID](#), [CF\\_CFDP\\_SendNak\(\)](#), [CF\\_ChunkList\\_ComputeGaps\(\)](#), [CF\\_SEND\\_PDU\\_NO\\_BUF\\_AVAIL\\_ERROR](#), [CFE\\_EVS\\_EventType\\_INFORMATION](#), [CFE\\_EVS\\_SendEvent\(\)](#), [CFE\\_SUCCESS](#), [CF\\_Transaction::chan\\_num](#), [CF\\_HkPacket\\_Payload::channel\\_hk](#), [CF\\_ChunkWrapper::chunks](#), [CF\\_Transaction::chunks](#), [CF\\_AppData\\_t::config\\_table](#), [CF\\_ChunkList::count](#), [CF\\_HkChannel\\_Data::counters](#), [CF\\_Transaction::flags](#), [CF\\_Transaction::fsize](#), [CF\\_Transaction::history](#), [CF\\_AppData\\_t::hk](#), [CF\\_Logical\\_PduBuffer::int\\_header](#), [CF\\_ConfigTable::local\\_eid](#), [CF\\_ChunkList::max\\_chunks](#), [CF\\_Flags\\_Rx::md\\_recv](#), [CF\\_Logical\\_IntHeader::nak](#), [CF\\_HkSent::nak\\_segment\\_requests](#), [CF\\_Logical\\_SegmentList::num\\_segments](#), [CF\\_Logical\\_SegmentRequest::offset\\_end](#), [CF\\_Logical\\_SegmentRequest::offset\\_start](#), [CF\\_HkPacket::Payload](#), [CF\\_History::peer\\_eid](#), [CF\\_StateFlags::rx](#), [CF\\_Logical\\_PduNak::scope\\_end](#), [CF\\_Logical\\_PduNak::scope\\_start](#), [CF\\_Logical\\_PduNak::segment\\_list](#), [CF\\_Logical\\_SegmentList::segments](#), [CF\\_HkCounters::sent](#), [CF\\_History::seq\\_num](#), and [CF\\_History::src\\_eid](#).

Referenced by CF\_CFDP\_R\_Tick\_Maintenance().

Here is the call graph for this function:



```
12.39.2.20 CF_CFDP_R_SubstateRecvEof() void CF_CFDP_R_SubstateRecvEof (
```

CF_Transaction_t * txn,	
CF_Logical_PduBuffer_t * ph )	

## Processing receive EOF common functionality for R1/R2.

## Description

This function is used for both R1 and R2 EOF receive. It calls the unmarshaling function and then checks known transaction data against the PDU.

### **Assumptions, External Events, and Notes:**

txnid must not be NULL. ph must not be NULL.

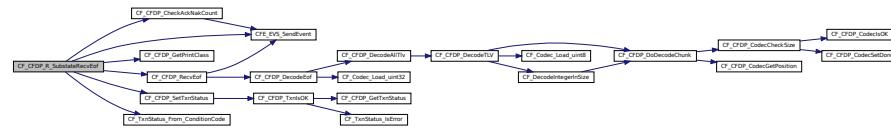
## Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 173 of file cf\_cfdp\_r.c.

References CF\_Logical\_PduEof::cc, CF\_AppData, CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_PDU\_EOF\_ERR\_EID, CF\_CFDP\_RecvEof(), CF\_CFDP\_SetTxnStatus(), CF\_TRACE, CF\_TxnStatus\_From\_ConditionCode(), CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduEof::crc, CF\_Logical\_IntHeader::eof, CF\_Flags\_Rx::eof\_count, CF\_StateData::eof\_crc, CF\_StateData::eof\_size, CF\_HkRecv::error, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_HkPacket::Payload, CF\_StateData::peer\_cc, CF\_HkCounters::recv, CF\_StateFlags::rx, CF\_History::seq\_num, CF\_Logical\_PduEof::size, CF\_History::src\_eid, and CF\_Transaction::state\_data. Referenced by CF\_CFDP\_R1\_Recv(), CF\_CFDP\_R2\_Recv(), and CF\_CFDP\_RecvHold().

Here is the call graph for this function:



**12.39.2.21 CF\_CFDP\_R\_SubstateRecvFileData()** void CF\_CFDP\_R\_SubstateRecvFileData (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

## Process received file data for R.

## Description

Writes data into temp storage file

## **Assumptions, External Events, and Notes:**

txnid must not be NULL. ph must not be NULL.

## Parameters

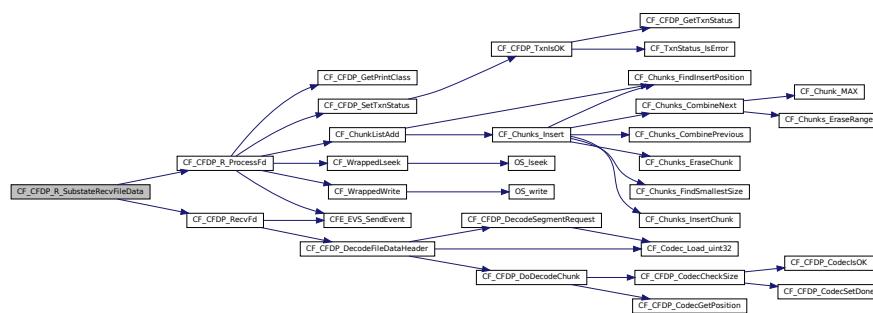
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 219 of file cf\_cfdp\_r.c.

References CF\_StateData::acknak\_count, CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_RecvFd(), CFE\_SUCCESS, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R1\_Recv(), and CF\_CFDP\_R2\_Recv().

Here is the call graph for this function:



**12.39.2.22 CF CFDP R SubstateRecvMd()** void CF CFDP R SubstateRecvMd (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Substate function to receive an MD.

#### Description

Receive and process an MD PDU

#### Assumptions, External Events, and Notes:

*txn* must not be NULL. *ph* must not be NULL.

#### Parameters

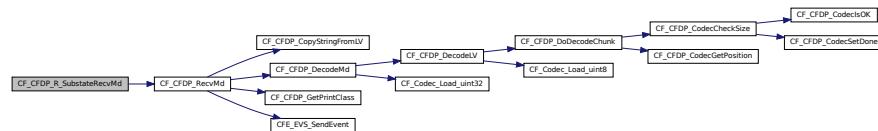
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 552 of file cf\_cfdp\_r.c.

References CF\_CFDP\_RecvMd(), CFE\_SUCCESS, CF\_Transaction::flags, CF\_Flags\_Rx::md\_recv, and CF\_State→Flags::rx.

Referenced by CF\_CFDP\_R1\_Recv(), and CF\_CFDP\_R2\_Recv().

Here is the call graph for this function:



### 12.39.2.23 CF\_CFDP\_R\_Tick() void CF\_CFDP\_R\_Tick ( CF\_Transaction\_t \* *txn* )

Perform tick (time-based) processing for R transactions.

#### Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send ACK, NAK, and FIN. It checks for inactivity timer and processes the ACK timer. The ACK timer is what triggers re-sends of PDUs that require acknowledgment.

#### Assumptions, External Events, and Notes:

*txn* must not be NULL. *cont* is unused, so may be NULL

#### Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

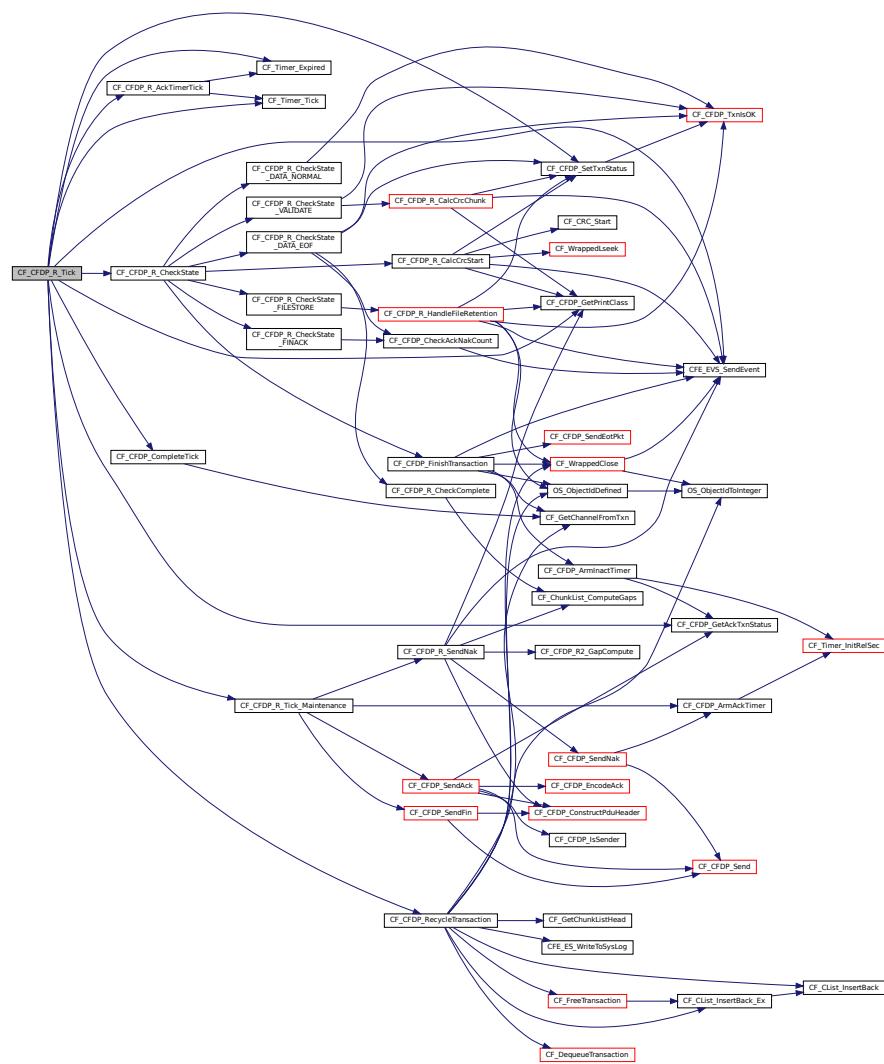
Definition at line 1060 of file cf\_cfdp\_r.c.

References CF\_AppData, CF\_CFDP\_AckTxnStatus\_ACTIVE, CF\_CFDP\_CompleteTick(), CF\_CFDP\_GetAck→TxnStatus(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_CheckState(), CF\_CFDP→\_R\_INACT\_TIMER\_ERR\_EID, CF\_CFDP\_R\_Tick\_Maintenance(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP→

\_SetTxnStatus(), CF\_Timer\_Expired(), CF\_Timer\_Tick(), CF\_TxnState\_HOLD, CF\_TxnStatus\_INACTIVITY ↔ DETECTED, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket ↔ Payload::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_HkCounters::fault, CF\_Transaction ↔ ::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Flags\_Common::inactivity\_fired, CF\_HkFault::inactivity\_timer, CF\_Transaction::inactivity\_timer, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state.

Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



#### 12.39.2.24 CF\_CFDP\_R\_Tick\_Maintenance()

```
void CF_CFDP_R_Tick_Maintenance (
    CF_Transaction_t * txn )
```

Generate protocol state PDUs as needed.

## Description

Generates the management PDUs such as FIN, NAK, and EOF-ACK

These PDUs are considered higher priority than file data

### Assumptions, External Events, and Notes:

`txn` must not be NULL.

## Parameters

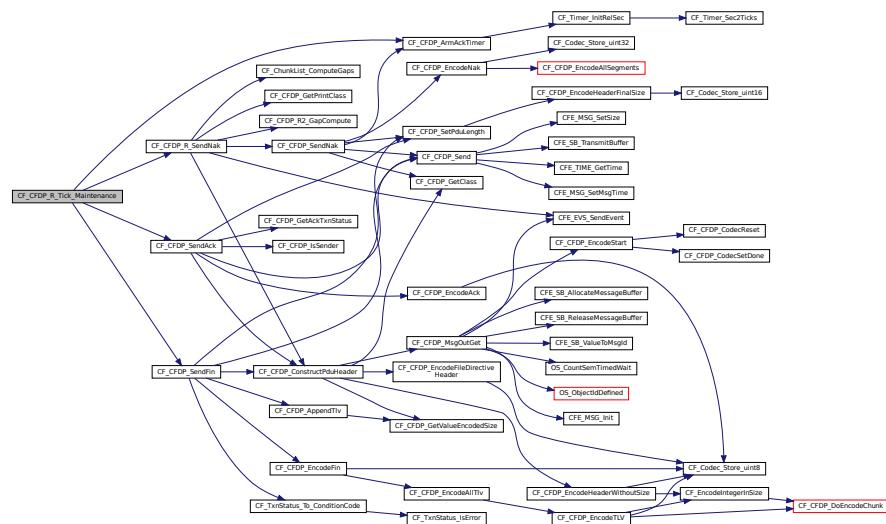
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 1014 of file `cf_cfdp_r.c`.

References `CF_CFDP_ArmAckTimer()`, `CF_CFDP_FileDirective_EOF`, `CF_CFDP_R_SendNak()`, `CF_CFDP_SendAck()`, `CF_CFDP_SendFin()`, `CFE_SUCCESS`, `CF_Flags_Rx::eof_ack_count`, `CF_Flags_Rx::eof_count`, `CF_Transaction::flags`, `CF_Transaction::reliable_mode`, `CF_StateFlags::rx`, `CF_Flags_Rx::send_fin`, and `CF_Flags_Rx::send_nak`.

Referenced by `CF_CFDP_R_Tick()`.

Here is the call graph for this function:



## 12.40 apps/cf/fsw/src/cf\_cfdp\_r.h File Reference

```
#include "cf_cfdp.h"
```

### Data Structures

- struct `CF_GapComputeArgs_t`

*Argument for Gap Compute function.*

### Functions

- void `CF_CFDP_R1_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`

- void `CF_CFDP_R2_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
  - R1 receive PDU processing.*
- void `CF_CFDP_R_AckTimerTick (CF_Transaction_t *txn)`
  - R2 receive PDU processing.*
  - Perform acknowledgement timer tick (time-based) processing for R transactions.*
- void `CF_CFDP_R_Tick (CF_Transaction_t *txn)`
  - Perform tick (time-based) processing for R transactions.*
- void `CF_CFDP_R_Init (CF_Transaction_t *txn)`
  - Initialize a transaction structure for R.*
- `CFE_Status_t CF_CFDP_R_CheckCrc (CF_Transaction_t *txn)`
  - Checks that the transaction file's CRC matches expected.*
- bool `CF_CFDP_R_CheckComplete (CF_Transaction_t *txn)`
  - Checks R transaction state for transaction completion status.*
- `CFE_Status_t CF_CFDP_R_ProcessFd (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
  - Process a filedata PDU on a transaction.*
- void `CF_CFDP_R_SubstateRecvEof (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
  - Processing receive EOF common functionality for R1/R2.*
- void `CF_CFDP_R_SubstateRecvFileData (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
  - Process received file data for R.*
- void `CF_CFDP_R2_GapCompute (const CF_ChunkList_t *chunks, const CF_Chunk_t *chunk, void *opaque)`
  - Loads a single NAK segment request.*
- `CFE_Status_t CF_CFDP_R_SendNak (CF_Transaction_t *txn)`
  - Send a NAK PDU for R2.*
- void `CF_CFDP_R_CalcCrcChunk (CF_Transaction_t *txn)`
  - Calculate up to the configured amount of bytes of CRC.*
- void `CF_CFDP_R_CalcCrcStart (CF_Transaction_t *txn)`
  - Begin calculation of the file CRC.*
- void `CF_CFDP_R2_SubstateRecvFinAck (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
  - Process receive FIN-ACK PDU.*
- void `CF_CFDP_R_SubstateRecvMd (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`
  - Substate function to receive an MD.*
- void `CF_CFDP_R_CheckState (CF_Transaction_t *txn)`
  - Run RX state machine for the transaction.*
- void `CF_CFDP_R_HandleFileRetention (CF_Transaction_t *txn)`
  - Remove/Move file after transaction.*
- void `CF_CFDP_R_Tick_Maintenance (CF_Transaction_t *txn)`
  - Generate protocol state PDUs as needed.*

### 12.40.1 Detailed Description

Implementation related to CFDP Receive File transactions

This file contains various state handling routines for transactions which are receiving a file.

### 12.40.2 Function Documentation

```
12.40.2.1 CF_CFDP_R1_Recv() void CF_CFDP_R1_Recv (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

R1 receive PDU processing.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

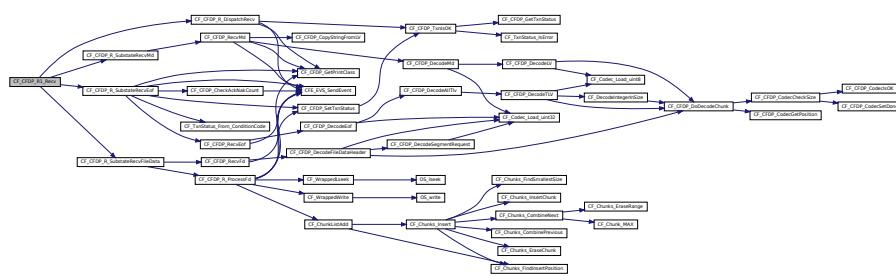
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 572 of file cf\_cfdp\_r.c.

References CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_FileDirective\_METADATA, CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_SubstateRecvFileData(), CF\_CFDP\_R\_SubstateRecvMd(), CF\_RxSubState\_DATA\_EOF, CF\_RxSubState\_DATA\_NORMAL, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, and CF\_CFDP\_R\_SubstateDispatchTable\_t::state.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



```
12.40.2.2 CF_CFDP_R2_GapCompute() void CF_CFDP_R2_GapCompute (
```

```
    const CF_ChunkList_t * chunks,
    const CF_Chunk_t * chunk,
    void * opaque )
```

Loads a single NAK segment request.

#### Description

This is a function callback from [CF\\_ChunkList\\_ComputeGaps\(\)](#).

#### Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL, opaque must not be NULL.

#### Parameters

<i>chunks</i>	Not used, required for compatibility with CF_ChunkList_ComputeGaps
<i>chunk</i>	Pointer to a single chunk information
<i>opaque</i>	Pointer to a <a href="#">CF_GapComputeArgs_t</a> object (passed via CF_ChunkList_ComputeGaps)

Definition at line 248 of file cf\_cfdp\_r.c.

References CF\_ASSERT, CF\_PDU\_MAX\_SEGMENTS, CF\_GapComputeArgs\_t::nak, CF\_Logical\_SegmentList::num\_segments, CF\_Chunk::offset, CF\_Logical\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_start, CF\_Logical\_PduNak::scope\_start, CF\_Logical\_PduNak::segment\_list, CF\_Logical\_SegmentList::segments, and CF\_Chunk::size.

Referenced by CF\_CFDP\_R\_SendNak().

```
12.40.2.3 CF_CFDP_R2_Recv() void CF_CFDP_R2_Recv (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

R2 receive PDU processing.

Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

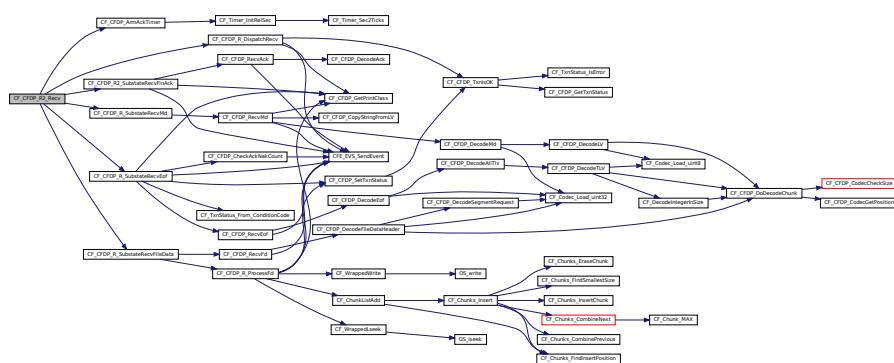
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 589 of file cf\_cfdp\_r.c.

References CF\_Flags\_Common::ack\_timer\_armed, CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_FileDirective\_ACK, CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_FileDirective\_METADATA, CF\_CFDP\_R2\_SubstateRecvFinAck(), CF\_CFDP\_R\_DispatchRecv(), CF\_CFDP\_R\_SubstateRecvEof(), CF\_CFDP\_R\_SubstateRecvFileData(), CF\_CFDP\_R\_SubstateRecvMd(), CF\_RxSubState\_COMPLETE, CF\_RxSubState\_DATA\_EOF, CF\_RxSubState\_DATA\_NORMAL, CF\_RxSubState\_FILESTORE, CF\_RxSubState\_FINACK, CF\_RxSubState\_VALIDATE, CF\_StateFlags::com, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, CF\_Transaction::flags, and CF\_CFDP\_R\_SubstateDispatchTable\_t::state.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



```
12.40.2.4 CF_CFDP_R2_SubstateRecvFinAck() void CF_CFDP_R2_SubstateRecvFinAck (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Process receive FIN-ACK PDU.

## Description

Receive and process a FIN-ACK PDU

### Assumptions, External Events, and Notes:

`txn` must not be NULL. `ph` must not be NULL.

## Parameters

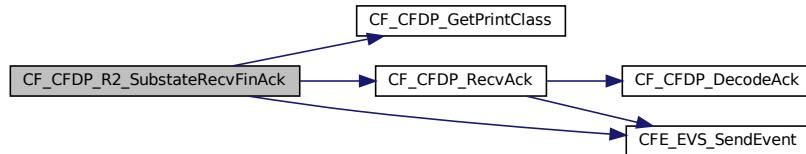
<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to the PDU information

Definition at line 511 of file `cf_cfdp_r.c`.

References `CF_Logical_IntHeader::ack`, `CF_Logical_PduAck::ack_directive_code`, `CF_Logical_PduAck::cc`, `CF_AppData`, `CF_CFDP_FileDirective_FIN`, `CF_CFDP_GetPrintClass()`, `CF_CFDP_R_PDU_FINACK_ERR_EID`, `CF_CFDP_RecvAck()`, `CF_TRACE`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_Transaction::chan_num`, `CF_HkPacket_Payload::channel_hk`, `CF_HkChannel_Data::counters`, `CF_HkRecv::error`, `CF_Flags_Rx::finack_recv`, `CF_Transaction::flags`, `CF_Transaction::history`, `CF_AppData_t::hk`, `CF_Logical_PduBuffer::int_header`, `CF_HkPacket::Payload`, `CF_HkCounters::recv`, `CF_StateFlags::rx`, `CF_History::seq_num`, `CF_History::src_eid`, and `CF_Logical_PduAck::txn_status`.

Referenced by `CF_CFDP_R2_Recv()`.

Here is the call graph for this function:



### 12.40.2.5 CF\_CFDP\_R\_AckTimerTick()

```
void CF_CFDP_R_AckTimerTick (
```

```
    CF_Transaction_t * txn )
```

Perform acknowledgement timer tick (time-based) processing for R transactions.

## Description

This is invoked as part of overall timer tick processing if the transaction has some sort of acknowledgement pending from the remote.

### Assumptions, External Events, and Notes:

`txn` must not be NULL

## Parameters

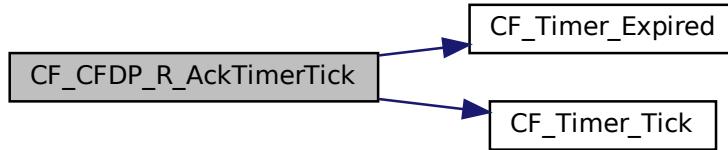
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 756 of file cf\_cfdp\_r.c.

References CF\_Transaction::ack\_timer, CF\_Flags\_Common::ack\_timer\_armed, CF\_Timer\_Expired(), CF\_Timer\_Tick(), CF\_StateFlags::com, CF\_Transaction::flags, and CF\_Transaction::reliable\_mode.

Referenced by CF\_CFDP\_R\_Tick().

Here is the call graph for this function:



#### 12.40.2.6 CF\_CFDP\_R\_CalcCrcChunk()

```
void CF_CFDP_R_CalcCrcChunk (
    CF_Transaction_t * txn )
```

Calculate up to the configured amount of bytes of CRC.

##### Description

The configuration table has a number of bytes to calculate per transaction per wakeup. At each wakeup, the file is read and this number of bytes are calculated. This function will set the checksum error condition code if the final CRC does not match.

##### PTFO

Increase throughput by consuming all CRC bytes per wakeup in transaction-order. This would require a change to the meaning of the value in the configuration table.

##### Assumptions, External Events, and Notes:

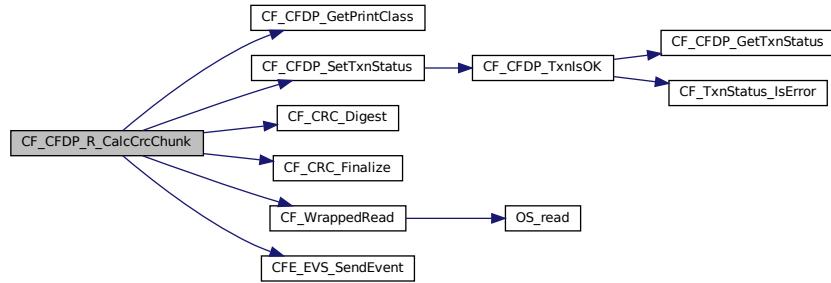
txn must not be NULL.

Definition at line 443 of file cf\_cfdp\_r.c.

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_READ\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_CRC\_Digest(), CF\_CRC\_Finalize(), CF\_R2\_CRC\_CHUNK\_SIZE, CF\_Txn\_Status\_FILE\_SIZE\_ERROR, CF\_WrappedRead(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_Flags\_Common::crc\_complete, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_read, CF\_Transaction::flags, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R\_CheckState\_VALIDATE().

Here is the call graph for this function:



**12.40.2.7 CF\_CFDP\_R\_CalcCrcStart()** void CF\_CFDP\_R\_CalcCrcStart ( CF\_Transaction\_t \* txn )

Begin calculation of the file CRC.

#### Description

Seeks back to the beginning of the file and initializes the CRC

#### Assumptions, External Events, and Notes:

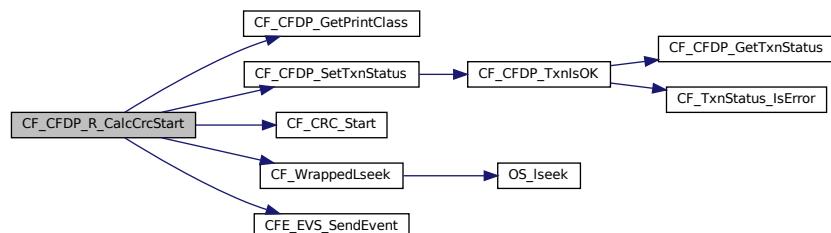
txn must not be NULL.

Definition at line 401 of file cf\_cfdp\_r.c.

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_SEEK\_CRC<=ERR\_EID, CF\_CFDP\_R\_SIZE\_MISMATCH\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_CRC\_Start(), CF\_TxnStatusFILE\_SIZE\_ERROR, CF\_WrappedLseek(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_Flags\_Common::crc\_complete, CF\_StateData::eof\_size, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_seek, CF\_Transaction::flags, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, OS\_SEEK\_SET, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R\_CheckState().

Here is the call graph for this function:



**12.40.2.8 CF\_CFDP\_R\_CheckComplete()** `bool CF_CFDP_R_CheckComplete ( CF_Transaction_t * txn )`

Checks R transaction state for transaction completion status.

#### Description

This function is called anywhere there's a desire to know if the transaction has completed. It may trigger other actions by setting flags to be handled during tick processing. In order for a transaction to be complete, it must have had its meta-data PDU received, the EOF must have been received, and there must be no gaps in the file. CRC is not checked in this function, because it's only called from functions after EOF is received.

#### Assumptions, External Events, and Notes:

`txn` must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

#### Returns

boolean indicating if the file is complete or not

Definition at line 75 of file cf\_cfdp\_r.c.

References `CF_ChunkList_ComputeGaps()`, `CF_ChunkWrapper::chunks`, `CF_Transaction::chunks`, `CF_Flags_Rx::eof_count`, `CF_Transaction::flags`, `CF_Transaction::fsize`, `CF_Flags_Rx::md_recv`, `CF_StateFlags::rx`, and `CF_Flags_Rx::send_nak`.

Referenced by `CF_CFDP_R_CheckState_DATA_EOF()`.

Here is the call graph for this function:



**12.40.2.9 CF\_CFDP\_R\_CheckCrc()** `CFE_Status_t CF_CFDP_R_CheckCrc ( CF_Transaction_t * txn )`

Checks that the transaction file's CRC matches expected.

#### Assumptions, External Events, and Notes:

`txn` must not be NULL.

#### Return values

<code>CFE_SUCCESS</code>	on CRC match, otherwise <code>CF_ERROR</code> .
--------------------------	---

### Parameters

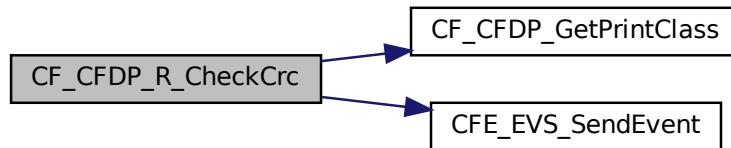
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 47 of file cf\_cfdp\_r.c.

References CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_CRC\_ERR\_EID, CF\_ERROR, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_HkFault::crc\_mismatch, CF\_StateData::eof\_crc, CF\_HkCounters::fault, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Crc::result, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R\_HandleFileRetention().

Here is the call graph for this function:



**12.40.2.10 CF\_CFDP\_R\_CheckState()** void CF\_CFDP\_R\_CheckState ( CF\_Transaction\_t \* *txn* )

Run RX state machine for the transaction.

### Description

Checks flags on the transaction and execute state transitions

### Assumptions, External Events, and Notes:

*txn* must not be NULL.

### Parameters

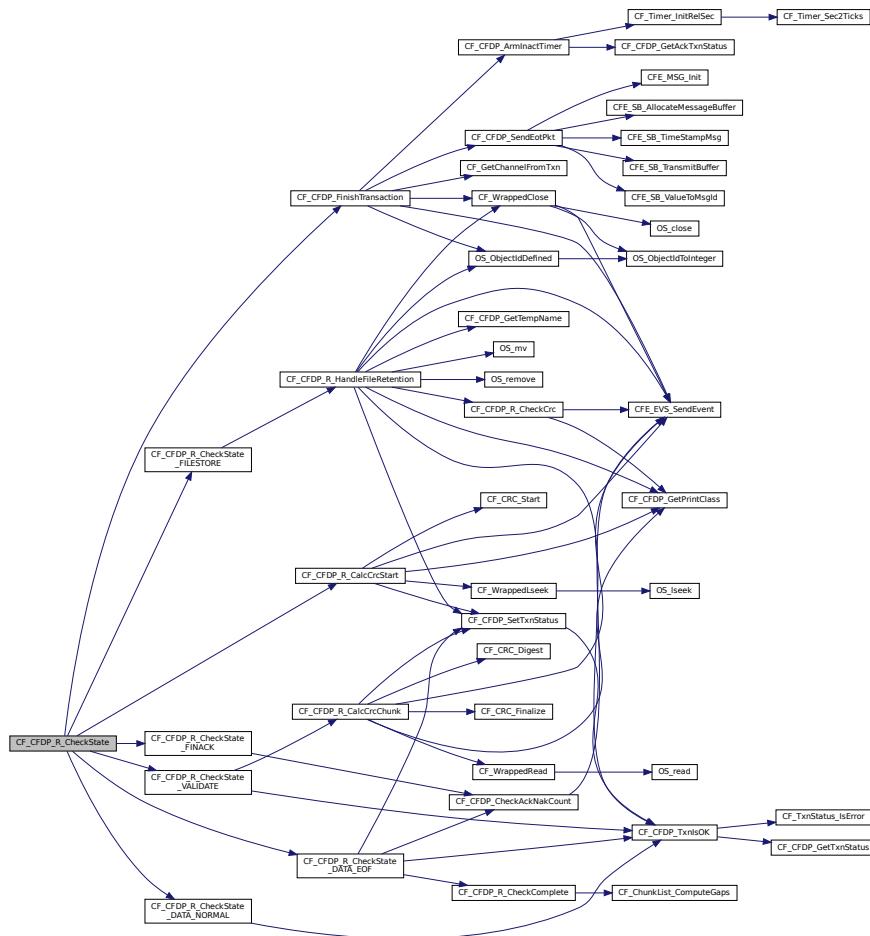
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 933 of file cf\_cfdp\_r.c.

References CF\_Flags\_Common::ack\_timer\_armed, CF\_StateData::acknak\_count, CF\_CFDP\_FinishTransaction(), CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_CheckState\_DATA\_EOF(), CF\_CFDP\_R\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_R\_CheckState\_FILESTORE(), CF\_CFDP\_R\_CheckState\_FINACK(), CF\_CFDP\_R\_CheckState\_VALIDATE(), CF\_RxSubState\_COMPLETE, CF\_RxSubState\_DATA\_EOF, CF\_RxSubState\_DATA\_NORMAL, CF\_RxSubState\_FILESTORE, CF\_RxSubState\_FINACK, CF\_RxSubState\_VALIDATE, CF\_TRACE, CF\_StateFlags::com, CF\_Transaction::flags, CF\_Transaction::reliable\_mode, CF\_StateFlags::rx, CF\_Flags\_Rx::send\_fin, CF\_Flags\_Rx::send\_nak, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_R\_Tick().

Here is the call graph for this function:



#### 12.40.2.11 `CF_CFDP_R_HandleFileRetention()`

```
void CF_CFDP_R_HandleFileRetention (
    CF_Transaction_t * txn )
```

Remove/Move file after transaction.

Determines disposition of local file after file transfer completion.

For a receiver:

- If the transfer was successful then the temp file is moved into the final location under the indicated name from the MD PDU.
- If the file transfer is unsuccessful then the temp file is deleted.

**Assumptions, External Events, and Notes:**

### Parameters

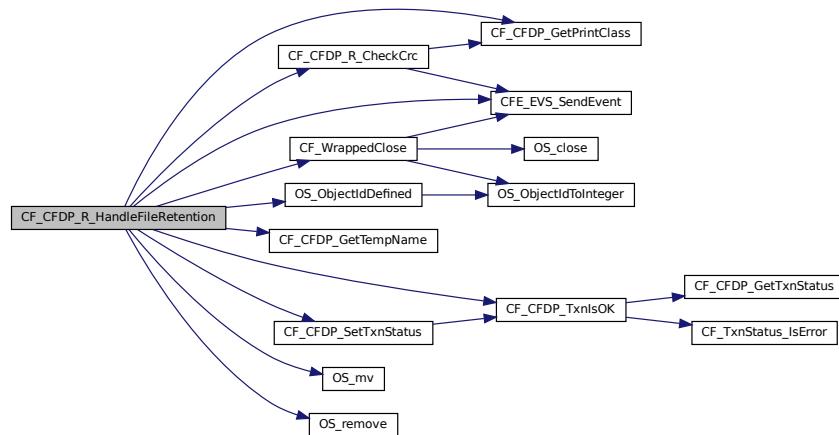
<i>txn</i>	Transaction object pointer
------------	----------------------------

Definition at line 631 of file cf\_cfdp\_r.c.

References CF\_CFDP\_FinDeliveryCode\_COMPLETE, CF\_CFDP\_FinDeliveryCode\_INCOMPLETE, CF\_CFDP\_FinDeliveryCode\_INVALID, CF\_CFDP\_FinFileStatus\_DISCARDED, CF\_CFDP\_FinFileStatus\_DISCARDED\_FILESTORE, CF\_CFDP\_FinFileStatus\_INVALID, CF\_CFDP\_FinFileStatus\_RETAINED, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_GetTempName(), CF\_CFDP\_R\_CheckCrc(), CF\_CFDP\_R\_FILE\_RETAINED\_EID, CF\_CFDP\_R\_NOT\_RETAINED\_EID, CF\_CFDP\_R\_RENAME\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_CFDP\_TxnIsOK(), CF\_Txn\_Status\_FILE\_CHECKSUM\_FAILURE, CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_TxnStatus\_INVALID\_FILE\_STRUCTURE, CF\_TxnStatus\_NO\_ERROR, CF\_WrappedClose(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_MISSION\_MAX\_PATH\_LEN, CFE\_SUCCESS, CF\_StateFlags::com, CF\_TxnFilenames::dst\_filename, CF\_Transaction::fd, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::history, CF\_Flags\_Common::is\_complete, OS\_mv(), OS\_OBJECT\_ID\_UNDEFINED, OS\_ObjectIdDefined(), OS\_remove(), OS\_SUCCESS, CF\_StateFlags::rx, CF\_History::seq\_num, CF\_History::src\_eid, CF\_Transaction::state\_data, CF\_Flags\_Rx::tempfile\_created, and CF\_History::txn\_stat.

Referenced by CF\_CFDP\_R\_CheckState\_FILESTORE().

Here is the call graph for this function:



**12.40.2.12 CF\_CFDP\_R\_Init()** void CF\_CFDP\_R\_Init ( CF\_Transaction\_t \* *txn* )

Initialize a transaction structure for R.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

### Parameters

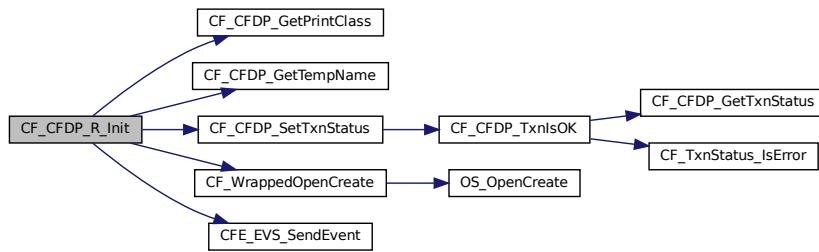
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 360 of file cf\_cfdp\_r.c.

References CF\_AppData, CF\_CFDP\_FinDeliveryCode\_INVALID, CF\_CFDP\_FinFileStatus\_INVALID, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_GetTempName(), CF\_CFDP\_R\_CREAT\_ERR\_EID, CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID, CF\_CFDP\_SetTxnStatus(), CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_WrappedOpenCreate(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_MISSION\_MAX\_FILE\_LEN, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_open, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, OS\_FILE\_FLAG\_CREATE, OS\_FILE\_FLAG\_TRUNCATE, OS\_OBJECT\_ID\_UNDEFINED, OS\_READ\_WRITE, CF\_HkPacket::Payload, CF\_StateFlags::rx, CF\_History::seq\_num, CF\_History::src\_eid, CF\_Transaction::state\_data, and CF\_Flags\_Rx::tempfile\_created.

Referenced by CF\_CFDP\_SetupRxTransaction().

Here is the call graph for this function:



**12.40.2.13 CF\_CFDP\_R\_ProcessFd()** CFE\_Status\_t CF\_CFDP\_R\_ProcessFd (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Process a filedata PDU on a transaction.

## Assumptions, External Events, and Notes:

txn must not be NULL.

## Return values

*CFE\_SUCCESS* on success. *CF\_ERROR* on error.

Parameters
<i>txn</i> Pointer to the transaction object

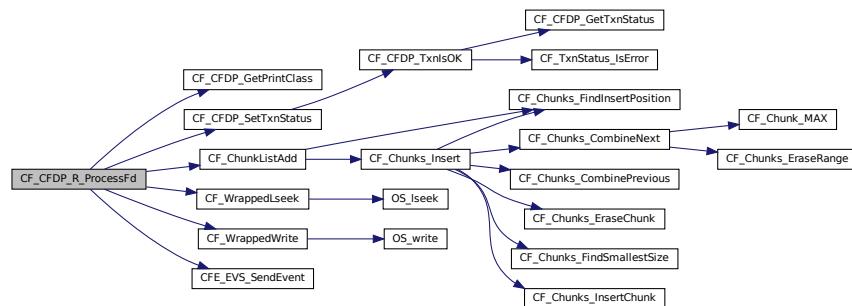
Definition at line 106 of file cf\_cfdp\_r.c

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID, CF\_CFDP\_R\_WRITE\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_ChunkListAdd(), CF\_ERROR, CF\_Txn\_Status\_FILE\_SIZE\_ERROR, CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_WrappedLseek(), CF\_WrappedWrite(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_Hk\_Packet\_Payload::channel\_hk, CF\_ChunkWrapper::chunks, CF\_Transaction::chunks, CF\_HkChannel\_Data::counters,

CF\_Logical\_PduFileDataHeader::data\_len, CF\_Logical\_PduFileDataHeader::data\_ptr, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_Logical\_IntHeader::fd, CF\_HkRecv::file\_data\_bytes, CF\_HkFault::file\_seek, CF\_HkFault::file\_write, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_PduFileDataHeader::offset, OS\_SEEK\_SET, CF\_HkPacket::Payload, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R\_SubstateRecvFileData().

Here is the call graph for this function:



**12.40.2.14 CF\_CFDP\_R\_SendNak()** `CFE_Status_t CF_CFDP_R_SendNak ( CF_Transaction_t * txn )`

Send a NAK PDU for R2.

#### Description

NAK PDU is sent when there are gaps in the received data. The chunks class tracks this and generates the NAK PDU by calculating gaps internally and calling [CF\\_CFDP\\_R2\\_GapCompute\(\)](#). There is a special case where if a metadata PDU has not been received, then a NAK packet will be sent to request another.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Return values

<code>CFE_SUCCESS</code>	on success. CF_ERROR on error.
--------------------------	--------------------------------

#### Parameters

<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

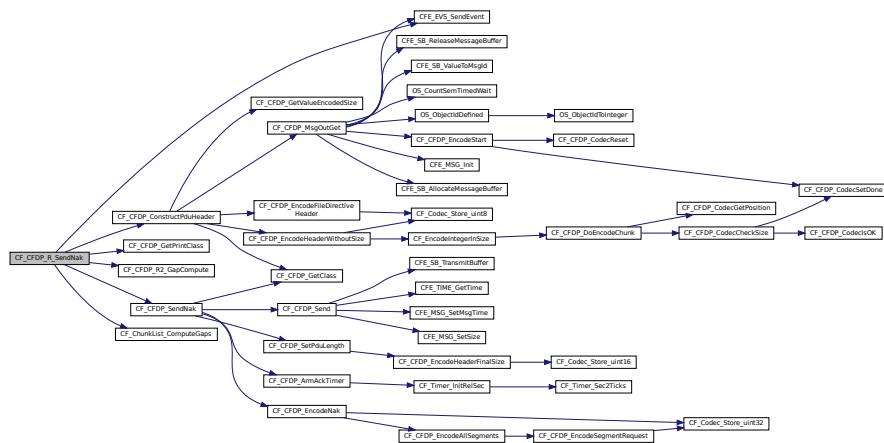
Definition at line 280 of file cf\_cfdp\_r.c.

References CF\_AppData, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_FileDirective\_NAK, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R2\_GapCompute(), CF\_CFDP\_R\_REQUEST\_MD\_INF\_EID, CF\_CFDP\_SendNak(), CF\_ChunkList\_ComputeGaps(), CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_ChunkWrapper::chunks, CF\_Transaction::chunks, CF\_AppData\_t::config\_table, CF\_ChunkList::count, CF\_HkChannel\_Data::counters, CF\_Transaction::flags, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, CF

\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_ChunkList::max\_chunks, CF\_Flags\_Rx::md\_recv, CF\_Logical\_IntHeader::nak, CF\_HkSent::nak\_segment\_requests, CF\_Logical\_SegmentList::num\_segments, CF↔\_Logical\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_start, CF\_HkPacket::Payload, CF↔History::peer\_eid, CF\_StateFlags::rx, CF\_Logical\_PduNak::scope\_end, CF\_Logical\_PduNak::scope\_start, CF↔Logical\_PduNak::segment\_list, CF\_Logical\_SegmentList::segments, CF\_HkCounters::sent, CF\_History::seq\_num, and CF\_History::src\_eid.

Referenced by CF\_CFDP\_R\_Tick\_Maintenance().

Here is the call graph for this function:



**12.40.2.15 CF\_CFDP\_R\_SubstateRecvEof()** void CF\_CFDP\_R\_SubstateRecvEof (   
`CF_Transaction_t * txn,`  
`CF_Logical_PduBuffer_t * ph )`

Processing receive EOF common functionality for R1/R2.

#### Description

This function is used for both R1 and R2 EOF receive. It calls the unmarshaling function and then checks known transaction data against the PDU.

#### Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

#### Parameters

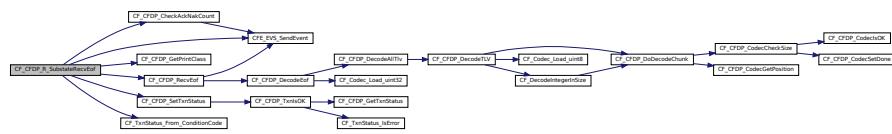
<code>txn</code>	Pointer to the transaction object
<code>ph</code>	Pointer to the PDU information

Definition at line 173 of file cf\_cfdp\_r.c.

References CF\_Logical\_PduEof::cc, CF\_AppData, CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_PDU\_EOF\_ERR\_EID, CF\_CFDP\_RecvEof(), CF\_CFDP\_SetTxnStatus(), CF\_TRACE, CF\_TxnStatus↔\_From\_ConditionCode(), CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED, CFE\_EVS\_EventType\_ERROR, CFE\_EVS↔\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel↔Data::counters, CF\_Logical\_PduEof::crc, CF\_Logical\_IntHeader::eof, CF\_Flags\_Rx::eof\_count, CF\_StateData::eof↔

`_crc, CF_StateData::eof_size, CF_HkRecv::error, CF_Transaction::flags, CF_Transaction::history, CF_AppData_t::hk, CF_Logical_PduBuffer::int_header, CF_HkPacket::Payload, CF_StateData::peer_cc, CF_HkCounters::recv, CF_StateFlags::rx, CF_History::seq_num, CF_Logical_PduEof::size, CF_History::src_eid, and CF_Transaction::state_data.`  
 Referenced by `CF_CFDP_R1_Recv()`, `CF_CFDP_R2_Recv()`, and `CF_CFDP_RecvHold()`.

Here is the call graph for this function:



**12.40.2.16 CF\_CFDP\_R\_SubstateRecvFileData()** void CF\_CFDP\_R\_SubstateRecvFileData (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

Process received file data for R.

## Description

Writes data into temp storage file

## **Assumptions, External Events, and Notes:**

txnid must not be NULL. ph must not be NULL.

## Parameters

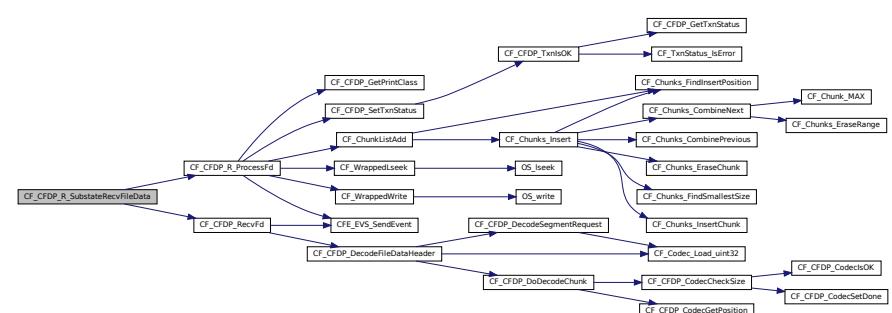
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 219 of file cf\_cfdp\_r.c.

References CF\_StateData::acknak\_count, CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_RecvFd(), CFE\_SUCCESS, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_R1\_Recv(), and CF\_CFDP\_R2\_Recv().

Here is the call graph for this function:



```
12.40.2.17 CF_CFDP_R_SubstateRecvMd() void CF_CFDP_R_SubstateRecvMd (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

Substate function to receive an MD.

#### Description

Receive and process an MD PDU

#### Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

#### Parameters

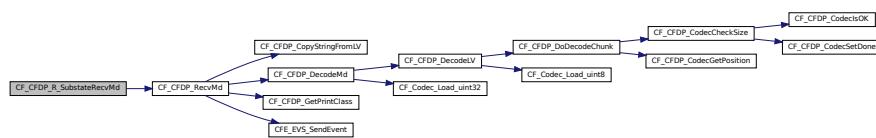
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 552 of file cf\_cfdp\_r.c.

References CF\_CFDP\_RecvMd(), CFE\_SUCCESS, CF\_Transaction::flags, CF\_Flags\_Rx::md\_recv, and CF\_State::Flags::rx.

Referenced by CF\_CFDP\_R1\_Recv(), and CF\_CFDP\_R2\_Recv().

Here is the call graph for this function:



```
12.40.2.18 CF_CFDP_R_Tick() void CF_CFDP_R_Tick (
    CF_Transaction_t * txn )
```

Perform tick (time-based) processing for R transactions.

#### Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send ACK, NAK, and FIN. It checks for inactivity timer and processes the ACK timer. The ACK timer is what triggers re-sends of PDUs that require acknowledgment.

#### Assumptions, External Events, and Notes:

txn must not be NULL. cont is unused, so may be NULL

#### Parameters

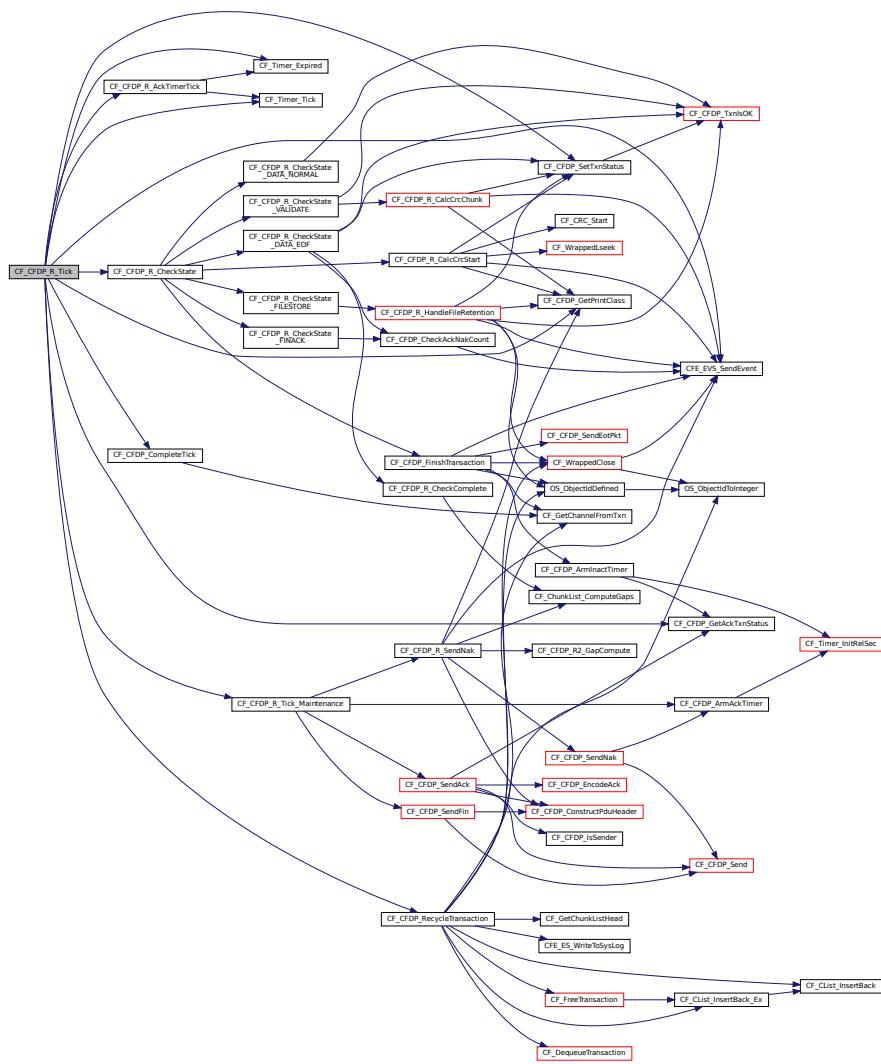
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 1060 of file cf\_cfdp\_r.c.

References CF\_AppData, CF\_CFDP\_AckTxnStatus\_ACTIVE, CF\_CFDP\_CompleteTick(), CF\_CFDP\_GetAckTxnStatus(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_CheckState(), CF\_CFDP\_R\_INACT\_TIMER\_ERR\_EID, CF\_CFDP\_R\_Tick\_Maintenance(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_SetTxnStatus(), CF\_Timer\_Expired(), CF\_Timer\_Tick(), CF\_TxnState\_HOLD, CF\_TxnStatus\_INACTIVITY\_DETECTED, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket::Payload::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_HkCounters::fault, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Flags\_Common::inactivity\_fired, CF\_HkFault::inactivity\_timer, CF\_Transaction::inactivity\_timer, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state.

Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



#### 12.40.2.19 CF\_CFDP\_R\_Tick\_Maintenance()

```
void CF_CFDP_R_Tick_Maintenance (
    CFE_Transaction_t * txn )
```

Generate protocol state PDUs as needed.

## Description

Generates the management PDUs such as FIN, NAK, and EOF-ACK

These PDUs are considered higher priority than file data

## **Assumptions, External Events, and Notes:**

txn must not be NULL.

## Parameters

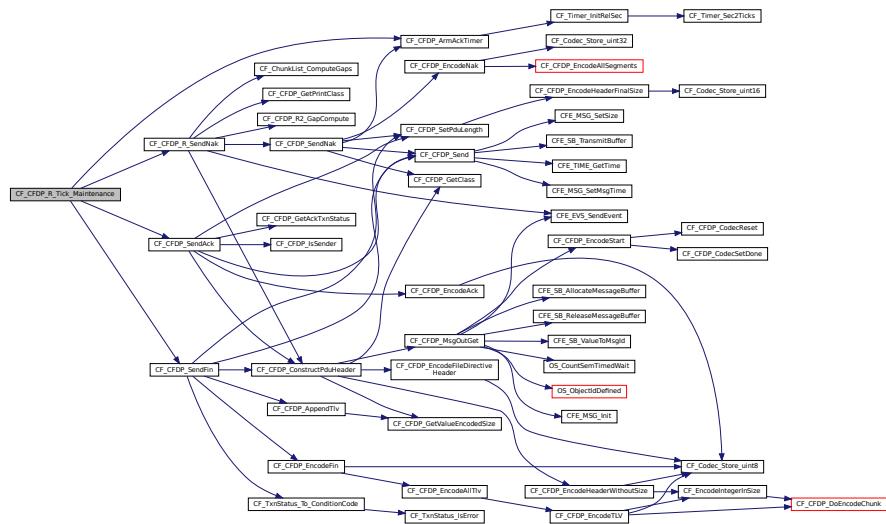
*txn* Pointer to the transaction object

Definition at line 1014 of file cf\_cfdp\_r.c.

References CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_R\_SendNak(), CF\_CFDP\_SendAck(), CF\_CFDP\_SendFin(), CFE\_SUCCESS, CF\_Flags\_Rx::eof\_ack\_count, CF\_Flags\_Rx::eof\_count, CF\_Transaction::flags, CF\_Transaction::reliable\_mode, CF\_StateFlags::rx, CF\_Flags\_Rx::send\_fin, and CF\_Flags\_Rx::send\_nak.

Referenced by CF\_CFDP\_R\_Tick().

Here is the call graph for this function:



## 12.41 apps/cf/fsw/src/cf cfdp s.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_s.h"
#include "cf_cfdp_dispatch.h"
#include <stdio.h>
#include <string.h>
```

```
#include "cf_assert.h"
```

## Functions

- `CFE_Status_t CF_CFDP_S_SendFileData (CF_Transaction_t *txn, uint32 foffs, uint32 bytes_to_read, uint8 calc_crc)`  
*Helper function to populate the PDU with file data and send it.*
- `void CF_CFDP_S_SubstateSendFileData (CF_Transaction_t *txn)`  
*Standard state function to send the next file data PDU for active transaction.*
- `void CF_CFDP_S_SubstateEarlyFin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*A FIN was received before file complete, so abandon the transaction.*
- `void CF_CFDP_S_SubstateRecvFin (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*S2 received FIN, so set flag to send FIN-ACK.*
- `void CF_CFDP_S2_SubstateNak (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*S2 NAK PDU received handling.*
- `void CF_CFDP_S2_SubstateEofAck (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*S2 received ACK PDU.*
- `void CF_CFDP_S1_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*S1 receive PDU processing.*
- `void CF_CFDP_S2_Recv (CF_Transaction_t *txn, CF_Logical_PduBuffer_t *ph)`  
*S2 receive PDU processing.*
- `void CF_CFDP_S_Init (CF_Transaction_t *txn)`  
*Initialize a transaction structure for S.*
- `void CF_CFDP_S_HandleFileRetention (CF_Transaction_t *txn)`  
*Remove/Move file after transaction.*
- `void CF_CFDP_S_AckTimerTick (CF_Transaction_t *txn)`  
*Perform acknowledgement timer tick (time-based) processing for S transactions.*
- `CF_TxSubState_t CF_CFDP_S_CheckState_DATA_NORMAL (CF_Transaction_t *txn)`
- `CF_TxSubState_t CF_CFDP_S1_CheckState_DATA_EOF (CF_Transaction_t *txn)`
- `CF_TxSubState_t CF_CFDP_S2_CheckState_DATA_EOF (CF_Transaction_t *txn)`
- `CF_TxSubState_t CF_CFDP_S_CheckState_DATA_EOF (CF_Transaction_t *txn)`
- `CF_TxSubState_t CF_CFDP_S_CheckState_FILESTORE (CF_Transaction_t *txn)`
- `void CF_CFDP_S_CheckState (CF_Transaction_t *txn)`  
*Run TX state machine for the transaction.*
- `void CF_CFDP_S_Tick (CF_Transaction_t *txn)`  
*Perform tick (time-based) processing for S transactions.*
- `void CF_CFDP_S_Tick_Maintenance (CF_Transaction_t *txn)`  
*Generate protocol messages for TX transactions.*
- `void CF_CFDP_S_Tick_Nak (CF_Transaction_t *txn)`  
*Generate file data PDUs from NAKs.*

### 12.41.1 Detailed Description

The CF Application CFDP send logic source file  
Handles all CFDP engine functionality specific to TX transactions.

### 12.41.2 Function Documentation

**12.41.2.1 CF\_CFDP\_S1\_CheckState\_DATA\_EOF()** CF\_TxSubState\_t CF\_CFDP\_S1\_CheckState\_DATA\_EOF ( CF\_Transaction\_t \* txn )

Definition at line 626 of file cf\_cfdp\_s.c.

## References Cited

References: CF\_CFBP\_FINI, IIC\_Status\_UNSET, CF\_Tx, CF\_XCubState\_ISELECTOR, CF\_Flags\_Common::close\_rq, CF\_StateFlags::com, CF\_Flags\_Tx::fin\_count, CF\_StateData::fin\_dc, CF\_Transaction::flags, CF\_Flags\_Common::is\_complete, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_CheckState\_DATA\_EOF().

```
12.41.2.2 CF_CFDP_S1_Recv() void CF_CFDP_S1_Recv (  
    CF_Transaction_t * txn,  
    CF_Logical_PduBuffer_t * ph )
```

S1 receive PDU processing.

## Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

## Parameters

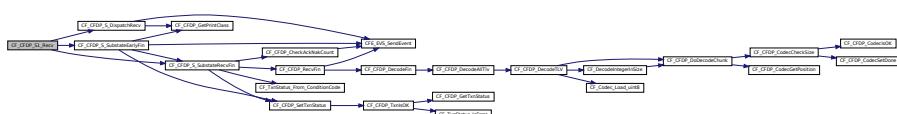
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 354 of file cf\_cfdp\_s.c.

References CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_TxSubState\_DATA\_EOF, CF\_TxSubState\_DATA\_NORMAL, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective and CF\_CFDP\_S\_SubstateRecvDispatchTable\_t::substate

Referenced by CE\_GEDP\_DispatchBcy()

Here is the call graph for this function:



**12.41.2.3 CF\_CFDP\_S2\_CheckState\_DATA\_EOF()** CF\_TxSubState\_t CF\_CFDP\_S2\_CheckState\_DATA\_EOF ( CF\_Transaction\_t \* txn )

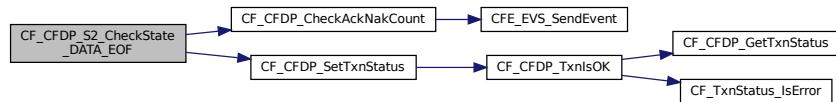
Definition at line 655 of file cf\_cfdp.s.c

References GE Flags Common::ack, t

References: CF\_Flags\_Common::ack\_time\_armed, CF\_StateData::ack\_max\_count, CF\_CFDP\_CheckTxnNaCount(), CF\_CFDP\_SetTxnStatus(), CF\_TxnStatus\_NAK\_LIMIT\_REACHED, CF\_TxSubState\_FILESTORE, CF\_StateFlags::com, CF\_Flags\_Tx::eof\_ack\_recv, CF\_Flags\_Tx::fin\_ack\_count, CF\_Flags\_Tx::fin\_count, CF\_Transaction::flags, CF\_Flags\_Common::is\_complete, CF\_Flags\_Tx::send\_eof, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_CheckState\_DATA\_EOF().

Here is the call graph for this function:



**12.41.2.4 CF\_CFDP\_S2\_Recv()** void CF\_CFDP\_S2\_Recv (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

S2 receive PDU processing.

## **Assumptions, External Events, and Notes:**

txn must not be NULL.

## Parameters

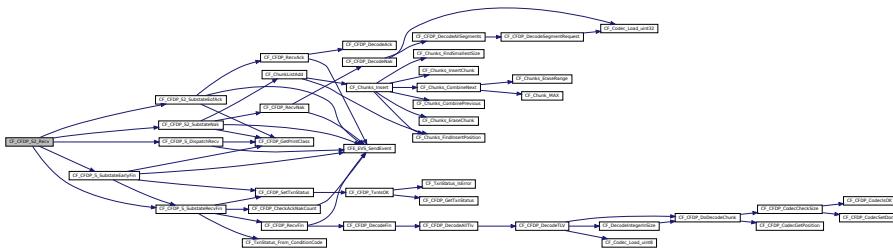
<i>txn</i>	Pointer to the transaction object
<i>pdu</i>	Pointer to the PDU information

Definition at line 375 of file cf\_cfdp\_s.c.

References CF\_CFDP\_FileDirective\_ACK, CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_FileDirective\_NAK, CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_TxSubState\_COMPLETE, CF\_TxSubState\_DATA\_EOF, CF\_TxSubState\_DATA\_NORMAL, CF\_TxSubState\_FILESTORE, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, and CF\_CFDP\_S\_SubstateRecvDispatchTable\_t::substate.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.41.2.5 CF\_CFDP\_S2\_SubstateEofAck()** void CF\_CFDP\_S2\_SubstateEofAck (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

S2 received ACK PDU.

### Description

Handles reception of an ACK PDU

### Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

### Parameters

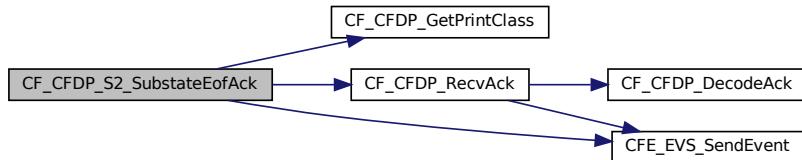
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 320 of file cf\_cfdp\_s.c.

References CF\_Logical\_IntHeader::ack, CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::cc, CF\_AppData, CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_RecvAck(), CF\_CFDP\_S\_PDU\_EOF\_←ERR\_EID, CF\_TRACE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Flags\_Tx::eof\_ack\_recv, CF\_HkRecv::error, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_HkPacket::Payload, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_History::src\_eid, CF\_StateFlags::tx, and CF\_Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_S2\_Recv().

Here is the call graph for this function:



### 12.41.2.6 CF\_CFDP\_S2\_SubstateNak()

```
void CF_CFDP_S2_SubstateNak (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

S2 NAK PDU received handling.

### Description

Stores the segment requests from the NAK packet in the chunks structure. These can be used to generate re-transmit filedata PDUs.

### Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

### Parameters

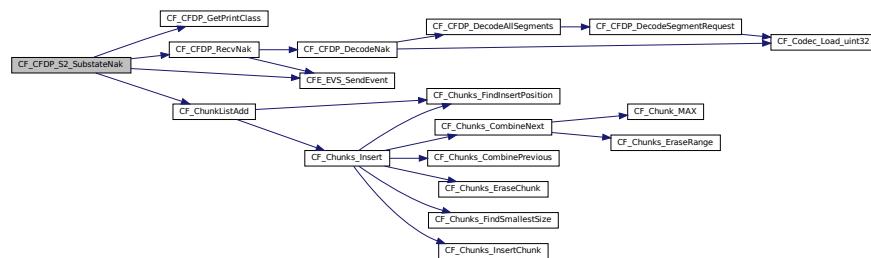
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 246 of file cf\_cfdp\_s.c.

References CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_RecvNak(), CF\_CFDP\_S\_INVALID\_SR\_ERR\_EID, CF\_CFDP\_S\_PDU\_NAK\_ERR\_EID, CF\_ChunkListAdd(), CF\_TRACE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_ChunkWrapper::chunks, CF\_Transaction::chunks, CF\_HkChannel\_Data::counters, CF\_HkRecv::error, CF\_Flags\_Tx::fd\_nak\_pending, CF\_Transaction::flags, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_IntHeader::nak, CF\_HkRecv::nak\_segment\_requests, CF\_Logical\_SegmentList::num\_segments, CF\_Logical\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_start, CF\_HkPacket::Payload, CF\_HkCounters::recv, CF\_Logical\_PduNak::segment\_list, CF\_Logical\_SegmentList::segments, CF\_Flags\_Tx::send\_md, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S2\_Recv().

Here is the call graph for this function:



**12.41.2.7 CF\_CFDP\_S\_AckTimerTick()** void CF\_CFDP\_S\_AckTimerTick ( CF\_Transaction\_t \* *txn* )

Perform acknowledgement timer tick (time-based) processing for S transactions.

### Description

This is invoked as part of overall timer tick processing if the transaction has some sort of acknowledgement pending from the remote.

### Assumptions, External Events, and Notes:

*txn* must not be NULL

### Parameters

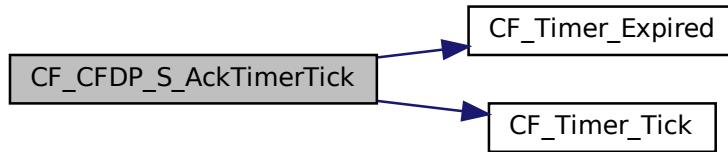
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 576 of file cf\_cfdp\_s.c.

References CF\_Transaction::ack\_timer, CF\_Flags\_Common::ack\_timer\_armed, CF\_Timer\_Expired(), CF\_Timer\_Tick(), CF\_StateFlags::com, CF\_Transaction::flags, and CF\_Transaction::reliable\_mode.

Referenced by CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



**12.41.2.8 CF\_CFDP\_S\_CheckState()** void CF\_CFDP\_S\_CheckState ( CF\_Transaction\_t \* txn )

Run TX state machine for the transaction.

#### Description

Checks flags on the transaction and execute state transitions

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

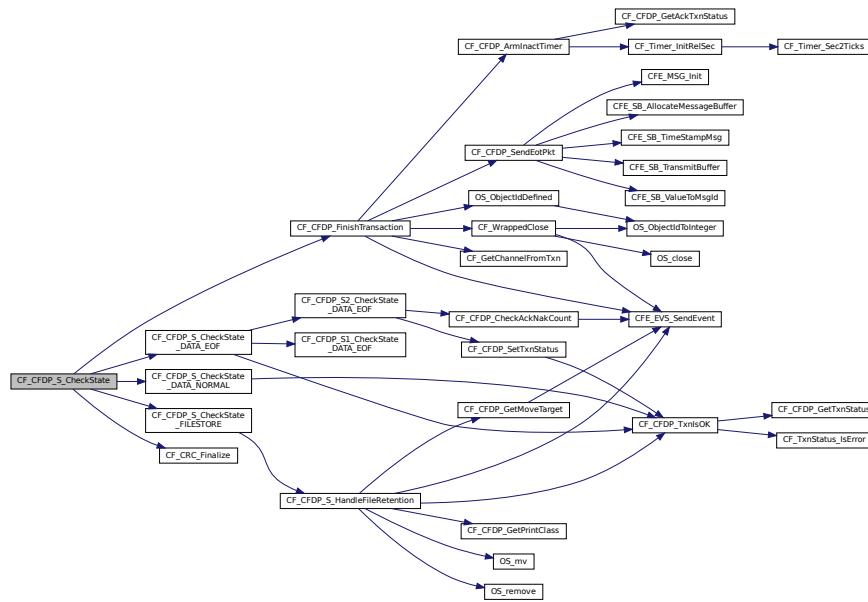
tx	Pointer to the transaction object
----	-----------------------------------

Definition at line 762 of file cf\_cfdp\_s.c.

References CF\_Flags\_Common::ack\_timer\_armed, CF\_StateData::acknak\_count, CF\_CFDP\_FinishTransaction(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_DATA\_NORMAL(), CF\_CFDP\_S\_CheckState\_FILESTORE(), CF\_CRC\_Finalize(), CF\_TRACE, CF\_TxSubState\_COMPLETE, CF\_TxSubState\_DATA\_EOF, CF\_TxSubState\_DATA\_NORMAL, CF\_TxSubState\_FILESTORE, CF\_StateFlags::com, CF\_Transaction::crc, CF\_Flags\_Common::crc\_complete, CF\_Transaction::flags, CF\_Flags\_Tx::send\_eof, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



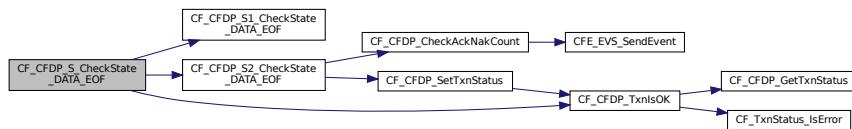
**12.41.2.9 CF\_CFDP\_S\_CheckState\_DATA\_EOF()** CF\_TxSubState\_t CF\_CFDP\_S\_CheckState\_DATA\_EOF ( CF\_Transaction\_t \* txn )

Definition at line 710 of file cf\_cfdp\_s.c.

References CF\_CFDP\_S1\_CheckState\_DATA\_EOF(), CF\_CFDP\_S2\_CheckState\_DATA\_EOF(), CF\_CFDP\_TxNlsOK(), CF\_TxSubState\_FILESTORE, CF\_Transaction::flags, CF\_Transaction::reliable\_mode, CF\_Flags\_Tx::send\_eof, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_CheckState().

Here is the call graph for this function:



**12.41.2.10 CF\_CFDP\_S\_CheckState\_DATA\_NORMAL()** CF\_TxSubState\_t CF\_CFDP\_S\_CheckState\_DATA\_NORMAL ( CF\_Transaction\_t \* txn )

Definition at line 601 of file cf\_cfdp.s.c.

References CF\_CFDP\_TxnIsOK(), CF\_TxSubState\_DATA\_EOF, CF\_TxSubState\_FILESTORE, CF\_Flags\_Tx::fin\_count, CF\_Transaction::flags, CF\_Transaction::foffs, CF\_Transaction::fsize, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_StateFlags::tx

Referenced by C\_E\_CEDP\_S\_CheckState()

Here is the call graph for this function:



#### 12.41.2.11 CF\_CFDP\_S\_CheckState\_FILESTORE()

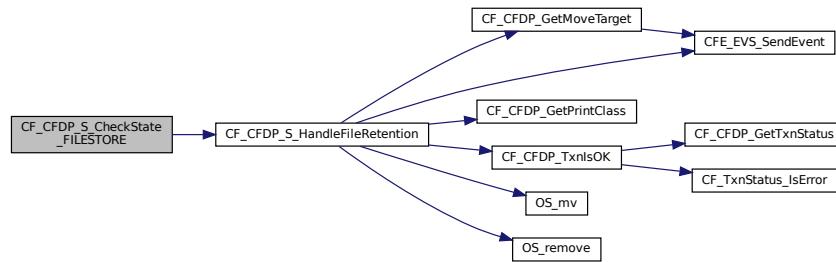
```
CF_TxSubState_t CF_CFDP_S_CheckState_FILESTORE (
    CF_Transaction_t * txn )
```

Definition at line 744 of file cf\_cfdp\_s.c.

References CF\_CFDP\_S\_HandleFileRetention(), CF\_TxSubState\_COMPLETE, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_S\_CheckState().

Here is the call graph for this function:



#### 12.41.2.12 CF\_CFDP\_S\_HandleFileRetention()

```
void CF_CFDP_S_HandleFileRetention (
    CF_Transaction_t * txn )
```

Remove/Move file after transaction.

Determines disposition of local file after file transfer completion.

For a sender:

- If the transfer is successful and the "keep" flag is false, then it applies the local file deletion policy (either delete directly or move to recycle dir)
- If the transfer is not successful or the "keep" flag is true, then do nothing

**Assumptions, External Events, and Notes:**

**Parameters**

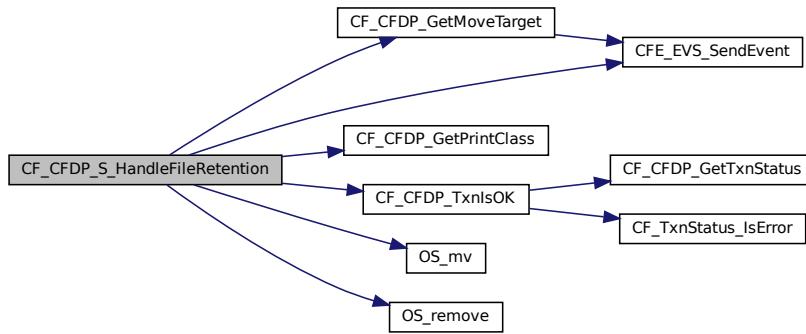
txn	Transaction object pointer
-----	----------------------------

Definition at line 495 of file cf\_cfdp\_s.c.

References CF\_AppData, CF\_CFDP\_FinDeliveryCode\_COMPLETE, CF\_CFDP\_FinFileStatus\_RETAINED, CF\_CFDP\_GetMoveTarget(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_FILE\_MOVED\_EID, CF\_CFDP\_S\_FILE\_REMOVED\_EID, CF\_CFDP\_TxnIsOK(), CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_MISSION\_MAX\_PATH\_LEN, CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_Flags\_Tx::cmd\_tx, CF\_StateFlags::com, CF\_AppData\_t::config\_table, CF\_ConfigTable::fail\_dir, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::history, CF\_Flags\_Common::is\_complete, CF\_Transaction::keep, CF\_ChannelConfig::move\_dir, OS\_mv(), OS\_remove(), CF\_Transaction::reliable\_mode, CF\_History::seq\_num, CF\_History::src\_eid, CF\_TxnFilenames::src\_filename, CF\_Transaction::state\_data, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_CheckState\_FILESTORE().

Here is the call graph for this function:



#### 12.41.2.13 CF\_CFDP\_S\_Init()

```
void CF_CFDP_S_Init (
    CF_Transaction_t * txn )
```

Initialize a transaction structure for S.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

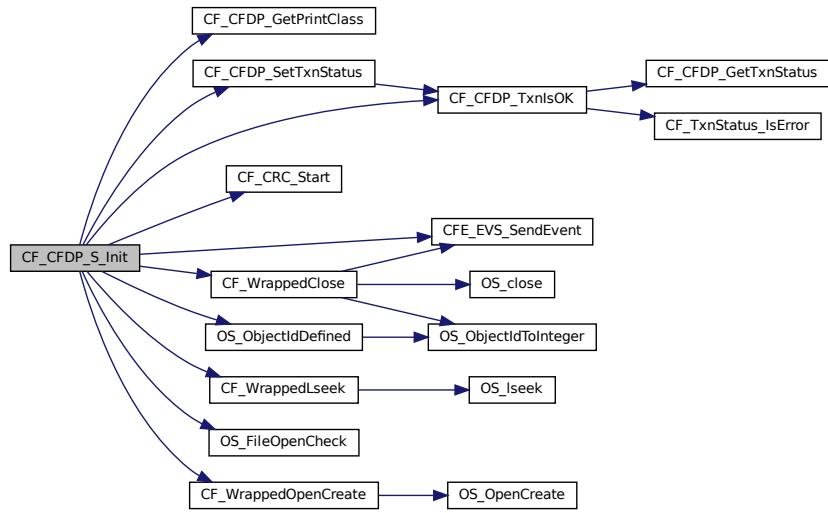
txn	Pointer to the transaction object
-----	-----------------------------------

Definition at line 402 of file cf\_cfdp\_s.c.

References CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_ALREADY\_OPEN\_ERR\_EID, CF\_CFDP\_S\_OPEN\_ERR\_EID, CF\_CFDP\_S\_SEEK\_BEG\_ERR\_EID, CF\_CFDP\_S\_SEEK\_END\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_CFDP\_TxnIsOK(), CF\_CRC\_Start(), CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_WrappedClose(), CF\_WrappedLseek(), CF\_WrappedOpenCreate(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_open, CF\_HkFault::file\_seek, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, OS\_FILE\_FLAG\_NONE, OS\_FileOpenCheck(), OS\_OBJECT\_ID\_UNDEFINED, OS\_ObjectIdDefined(), OS\_READ\_ONLY, OS\_SEEK\_END, OS\_SEEK\_SET, OS\_SUCCESS, CF\_HkPacket::Payload, CF\_Flags\_Tx::send\_md, CF\_History::seq\_num, CF\_History::src\_eid, CF\_TxnFilenames::src\_filename, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



#### 12.41.2.14 CF\_CFDP\_S\_SendFileData()

```

CF_Status_t CF_CFDP_S_SendFileData (
    CF_Transaction_t * txn,
    uint32 foffs,
    uint32 bytes_to_read,
    uint8 calc_crc )
  
```

Helper function to populate the PDU with file data and send it.

##### Description

This function checks the file offset cache and if the desired location is where the file offset is, it can skip a seek() call. The file is read into the filedata PDU and then the PDU is sent.

##### Assumptions, External Events, and Notes:

*txn* must not be NULL.

##### Returns

The number of bytes sent in the file data PDU (CFE\_SUCCESS, i.e. 0, if no bytes were processed), or CF\_ERROR on error

##### Parameters

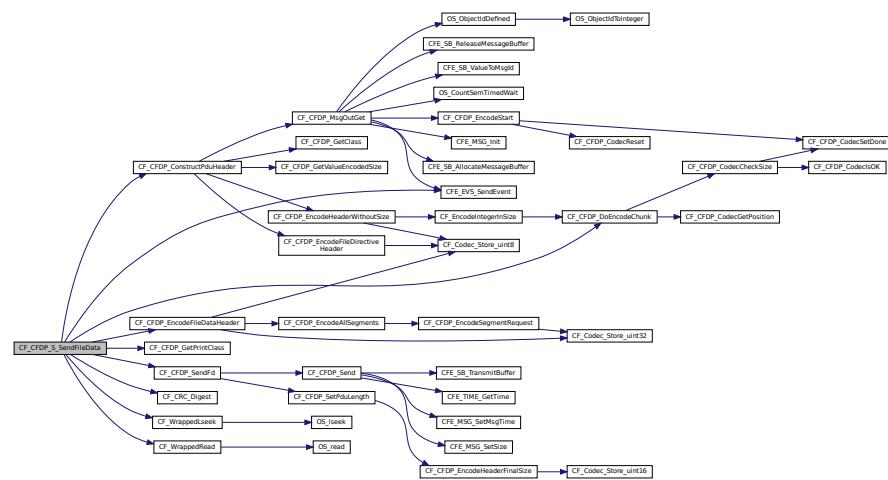
<i>txn</i>	Pointer to the transaction object
<i>foffs</i>	Position in file to send data from
<i>bytes_to_read</i>	Number of bytes to send (maximum)
<i>calc_crc</i>	Enable CRC/Checksum calculation

Definition at line 49 of file cf\_cfdp\_s.c.

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_DoEncode(), CF\_CFDP\_EncodeFileDataHeader(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_READ\_ERR\_EID, CF\_CFDP\_S\_SEEK\_FD\_ERR\_EID, CF\_CFDP\_SendFd(), CF\_CODEC\_GET\_REMAIN, CF\_CRC\_Digest(), CF\_ERROR, CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_WrappedLseek(), CF\_WrappedRead(), CFE\_EVS\_EventType\_Error, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_AppData\_t::config\_table, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_Logical\_PduFileDataHeader::data\_len, CF\_Logical\_PduFileDataHeader::data\_ptr, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_Logical\_InterruptHeader::fd, CF\_HkSent::file\_data\_bytes, CF\_HkFault::file\_read, CF\_HkFault::file\_seek, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_Logical\_PduFileDataHeader::offset, OS\_SEEK\_SET, CF\_ConfigTable::outgoing\_file\_chunk\_size, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdu\_header, CF\_History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_Logical\_PduHeader::segment\_meta\_flag, CF\_Hk\_Counters::sent, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_S\_SubstateSendFileData(), and CF\_CFDP\_S\_Tick\_Nak().

Here is the call graph for this function:



**12.41.2.15 CF\_CFDP\_S\_SubstateEarlyFin()** void CF\_CFDP\_S\_SubstateEarlyFin (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

A FIN was received before file complete, so abandon the transaction.

### **Assumptions, External Events, and Notes:**

tx must not be NULL. ph must not be NULL.

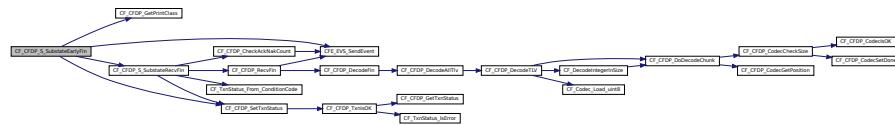
## Parameters

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 190 of file cf\_cfdp\_s.c.

References CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID, CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_SetTxnStatus(), CF\_TxnStatus\_EARLY\_FIN, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_

Transaction::history, CF\_History::seq\_num, and CF\_History::src\_eid.  
Referenced by CF\_CFDP\_S1\_Recv(), and CF\_CFDP\_S2\_Recv().  
Here is the call graph for this function:



#### 12.41.2.16 CF\_CFDP\_S\_SubstateRecvFin()

```
void CF_CFDP_S_SubstateRecvFin (
    CF_Transaction_t * txn,
    CF_Logical_PduBuffer_t * ph )
```

S2 received FIN, so set flag to send FIN-ACK.

**Assumptions, External Events, and Notes:**

txn must not be NULL. ph must not be NULL.

#### Parameters

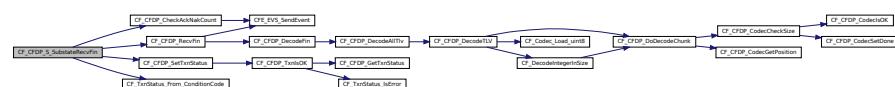
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 211 of file cf\_cfdp\_s.c.

References CF\_Logical\_PduFin::cc, CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_RecvFin(), CF\_CFDP\_SetTxnStatus(), CF\_TRACE, CF\_TxnStatus\_From\_ConditionCode(), CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED, CFE\_SUCCESS, CF\_Logical\_PduFin::delivery\_code, CF\_Logical\_PduFin::file\_status, CF\_Logical\_ExtHeader::fin, CF\_Flags\_Tx::fin\_count, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::flags, CF\_Logical\_PduBuffer::int\_header, CF\_StateData::peer\_cc, CF\_Transaction::state\_data, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_RecvHold(), CF\_CFDP\_S1\_Recv(), CF\_CFDP\_S2\_Recv(), and CF\_CFDP\_S\_SubstateEarlyFin().

Here is the call graph for this function:



#### 12.41.2.17 CF\_CFDP\_S\_SubstateSendFileData()

```
void CF_CFDP_S_SubstateSendFileData (
    CF_Transaction_t * txn )
```

Standard state function to send the next file data PDU for active transaction.

#### Description

During the transfer of active transaction file data PDUs, the file offset is saved. This function sends the next chunk of data. If the file offset equals the file size, then transition to the EOF state.

## Assumptions, External Events, and Notes:

`txn` must not be `NULL`.

## Parameters

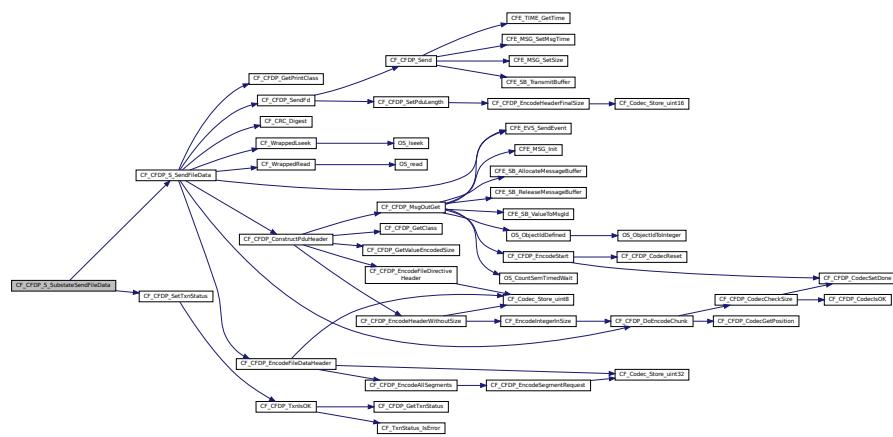
*txn* Pointer to the transaction object

Definition at line 163 of file cf\_cfdp\_s.c.

References CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SetTxnStatus(), CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TxnStatus\_READ\_FAILURE, CF\_Transaction::foffs, and CF\_Transaction::fsize.

Referenced by CF CFDP S Tick NewData().

Here is the call graph for this function:



**12.41.2.18 CF\_CFDP\_S\_Tick()** void CF\_CFDP\_S\_Tick ( CF\_Transaction\_t \* txn )

Perform tick (time-based) processing for S transactions.

## Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send EOF or FIN-ACK. If nothing else is sent, it checks to see if a NAK retransmit must occur.

## **Assumptions, External Events, and Notes:**

txn must not be NULL. cont is unused, so may be NULL

## Parameters

*txn* Pointer to the transaction object

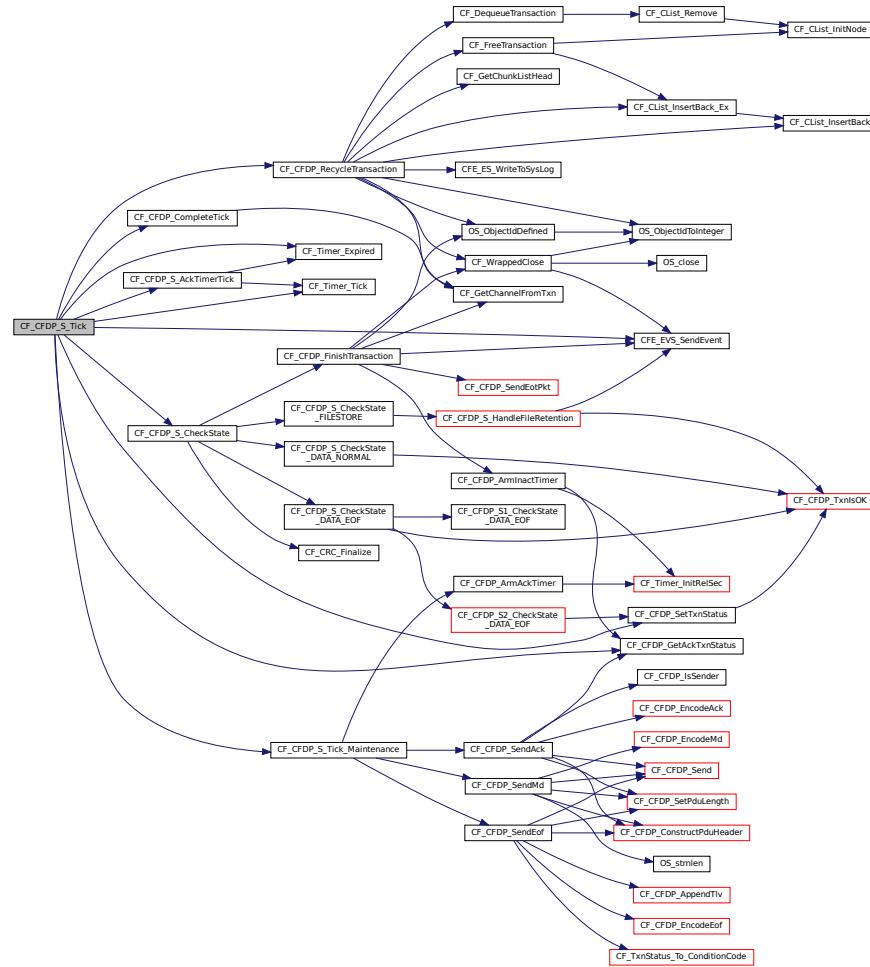
Definition at line 824 of file cf\_cfdp.s.c.

References CF\_AppData, CF\_CFDP\_AckTxnStatus\_ACTIVE, CF\_CFDP\_CompleteTick(), CF\_CFDP\_GetAckTxnStatus(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S\_AckTimerTick(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S

\_S\_INACT\_TIMER\_ERR\_EID, CF\_CFDP\_S\_Tick\_Maintenance(), CF\_CFDP\_SetTxnStatus(), CF\_Timer\_Expired(), CF\_Timer\_Tick(), CF\_TxnState\_HOLD, CF\_TxnStatus\_INACTIVITY\_DETECTED, CF\_TxSubState\_DATA\_NORMAL, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload<::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_HkCounters::fault, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Flags\_Common::inactivity\_fired, CF\_HkFault::inactivity\_timer, CF\_Transaction::inactivity\_timer, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, CF\_Transaction::state, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



#### 12.41.2.19 CF\_CFDP\_S\_Tick\_Maintenance()

```
void CF_CFDP_S_Tick_Maintenance (
    CF_Transaction_t * txn )
```

Generate protocol messages for TX transactions.

##### Description

This function is called at tick processing time to send pending protocol messages such as MD, EOF, and FIN-ACK

These messages are considered higher priority than any file data

### Assumptions, External Events, and Notes:

txn must not be NULL.

### Parameters

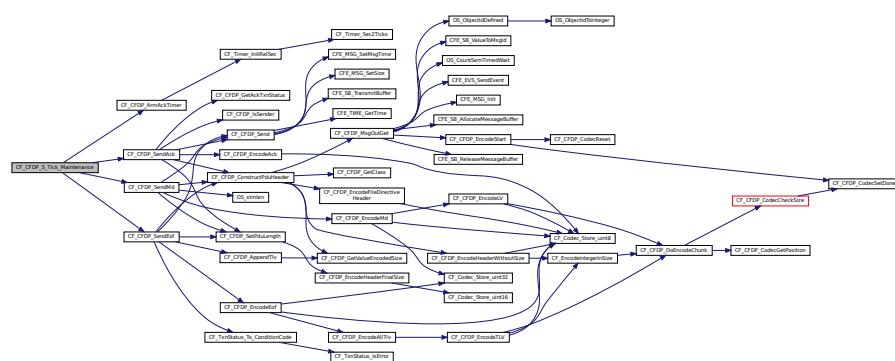
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 890 of file cf\_cfdp\_s.c.

References CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendMd(), CFE\_SUCCESS, CF\_Flags\_Tx::fin\_ack\_count, CF\_Flags\_Tx::fin\_count, CF\_Transaction::flags, CF\_Transaction::reliable\_mode, CF\_Flags\_Tx::send\_eof, CF\_Flags\_Tx::send\_md, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



**12.41.2.20 CF\_CFDP\_S\_Tick\_Nak()** void CF\_CFDP\_S\_Tick\_Nak ( CF\_Transaction\_t \* txn )

Generate file data PDUs from NAKs.

### Description

Generates file data PDUs for chunks that the peer has NAKed

Responding to NAK is considered higher priority than sending new file data, but lower priority than maintenance PDUs.

### Assumptions, External Events, and Notes:

txn must not be NULL.

### Parameters

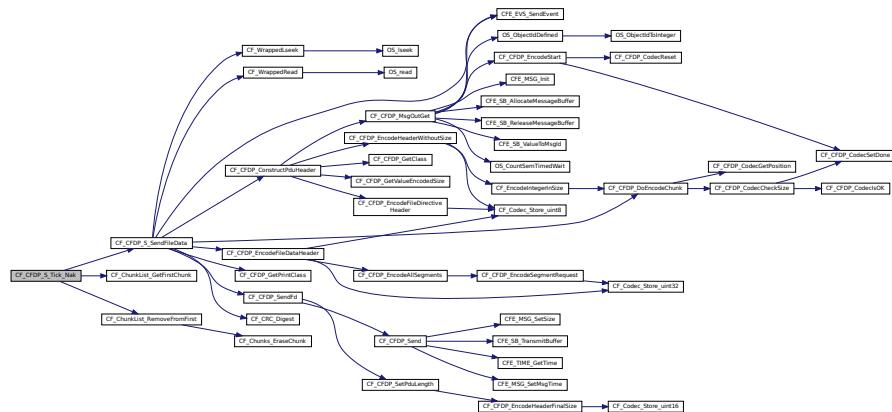
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 940 of file cf\_cfdp\_s.c.

References CF\_CFDP\_S\_SendFileData(), CF\_ChunkList\_GetFirstChunk(), CF\_ChunkList\_RemoveFromFirst(), CF\_ChunkWrapper::chunks, CF\_Transaction::chunks, CF\_Flags\_Tx::fd\_nak\_pending, CF\_Transaction::flags, CF\_Chunk::offset, CF\_Chunk::size, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



## 12.42 apps/cf/fsw/src/cf\_cfdp\_s.h File Reference

```
#include "cf_cfdp_types.h"
```

# Functions

- void **CF\_CFDP\_S1\_Recv** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*S1 receive PDU processing.*
  - void **CF\_CFDP\_S2\_Recv** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*S2 receive PDU processing.*
  - void **CF\_CFDP\_S\_AckTimerTick** (**CF\_Transaction\_t** \*txn)  
*Perform acknowledgement timer tick (time-based) processing for S transactions.*
  - void **CF\_CFDP\_S\_Tick** (**CF\_Transaction\_t** \*txn)  
*Perform tick (time-based) processing for S transactions.*
  - void **CF\_CFDP\_S\_Tick\_Maintenance** (**CF\_Transaction\_t** \*txn)  
*Generate protocol messages for TX transactions.*
  - void **CF\_CFDP\_S\_Tick\_Nak** (**CF\_Transaction\_t** \*txn)  
*Generate file data PDUs from NAKs.*
  - void **CF\_CFDP\_S\_Init** (**CF\_Transaction\_t** \*txn)  
*Initialize a transaction structure for S.*
  - **CFE\_Status\_t CF\_CFDP\_S\_SendFileData** (**CF\_Transaction\_t** \*txn, **uint32** foffs, **uint32** bytes\_to\_read, **uint8** calc\_crc)  
*Helper function to populate the PDU with file data and send it.*
  - void **CF\_CFDP\_S\_SubstateSendFileData** (**CF\_Transaction\_t** \*txn)  
*Standard state function to send the next file data PDU for active transaction.*
  - void **CF\_CFDP\_S\_SubstateEarlyFin** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*A FIN was received before file complete, so abandon the transaction.*
  - void **CF\_CFDP\_S\_SubstateRecvFin** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*S2 received FIN, so set flag to send FIN-ACK.*
  - void **CF\_CFDP\_S2\_SubstateNak** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*S2 NAK PDU received handling.*

- void **CF\_CFDP\_S2\_SubstateEofAck** (**CF\_Transaction\_t** \*txn, **CF\_Logical\_PduBuffer\_t** \*ph)  
*S2 received ACK PDU.*
  - void **CF\_CFDP\_S\_CheckState** (**CF\_Transaction\_t** \*txn)  
*Run TX state machine for the transaction.*
  - void **CF\_CFDP\_S\_HandleFileRetention** (**CF\_Transaction\_t** \*txn)  
*Remove/Move file after transaction.*

#### **12.42.1 Detailed Description**

## Implementation related to CFDP Send File transactions

This file contains various state handling routines for transactions which are sending a file.

## 12.42.2 Function Documentation

#### **12.42.2.1 CF\_CFDP\_S1\_Recv()** void CF\_CFDP\_S1\_Recv (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

S1 receive PDU processing.

### **Assumptions, External Events, and Notes:**

`txn` must not be `NULL`.

## Parameters

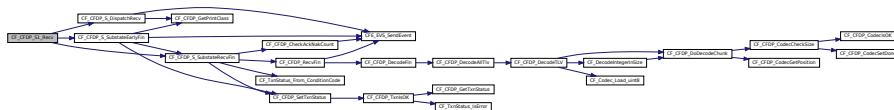
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 354 of file cf\_cfdp\_s.c.

References CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_TxSubState\_DATA\_EOF, CF\_TxSubState\_DATA\_NORMAL, CF\_CFDP\_FileDirectiveDispatchTable t::fdirective, and CF\_CFDP\_S\_SubstateRecvDispatchTable t::substate.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.42.2.2 CF\_CFDP\_S2\_Recv()** void CF\_CFDP\_S2\_Recv (

```
CF_Transaction_t * txn,  
CF_Logical_PduBuffer_t * ph )
```

S2 receive PDU processing.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

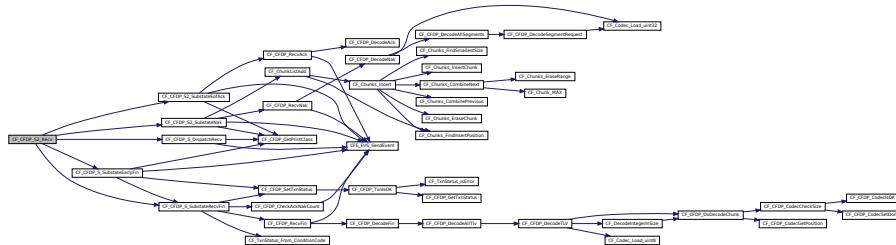
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 375 of file cf\_cfdp\_s.c.

References CF\_CFDP\_FileDirective\_ACK, CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_FileDirective\_NAK, CF\_CFDP\_S2\_SubstateEofAck(), CF\_CFDP\_S2\_SubstateNak(), CF\_CFDP\_S\_DispatchRecv(), CF\_CFDP\_S\_SubstateEarlyFin(), CF\_CFDP\_S\_SubstateRecvFin(), CF\_TxSubState\_COMPLETE, CF\_TxSubState\_DATA\_EOF, CF\_TxSubState\_DATA\_NORMAL, CF\_TxSubState\_FILESTORE, CF\_CFDP\_FileDirectiveDispatchTable\_t::fdirective, and CF\_CFDP\_S\_SubstateRecvDispatchTable\_t::substate.

Referenced by CF\_CFDP\_DispatchRecv().

Here is the call graph for this function:



**12.42.2.3 CF\_CFDP\_S2\_SubstateEofAck()** void CF\_CFDP\_S2\_SubstateEofAck ( CF\_Transaction\_t \* *txn*, CF\_Logical\_PduBuffer\_t \* *ph* )

S2 received ACK PDU.

**Description**

Handles reception of an ACK PDU

**Assumptions, External Events, and Notes:**

txn must not be NULL. ph must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

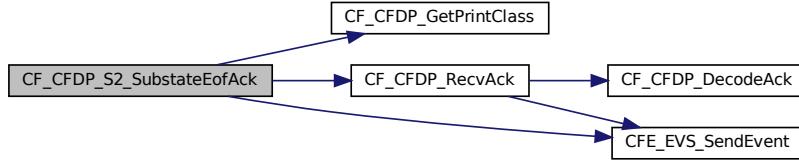
Definition at line 320 of file cf\_cfdp\_s.c.

References CF\_Logical\_IntHeader::ack, CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::cc, CF\_AppData, CF\_CFDP\_FileDirective\_EOF, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_RecvAck(), CF\_CFDP\_S\_PDU\_EOF->ERR\_EID, CF\_TRACE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction-

::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Flags\_Tx::eof\_ack\_recv, CF\_HkRecv::error, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_HkPacket::Payload, CF\_HkCounters::recv, CF\_History::seq\_num, CF\_History::src\_eid, CF\_StateFlags::tx, and CF\_Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_S2\_Recv().

Here is the call graph for this function:



**12.42.2.4 CF\_CFDP\_S2\_SubstateNak()** void CF\_CFDP\_S2\_SubstateNak ( CF\_Transaction\_t \* txn,  
CF\_Logical\_PduBuffer\_t \* ph )

S2 NAK PDU received handling.

#### Description

Stores the segment requests from the NAK packet in the chunks structure. These can be used to generate re-transmit filedata PDUs.

#### Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

#### Parameters

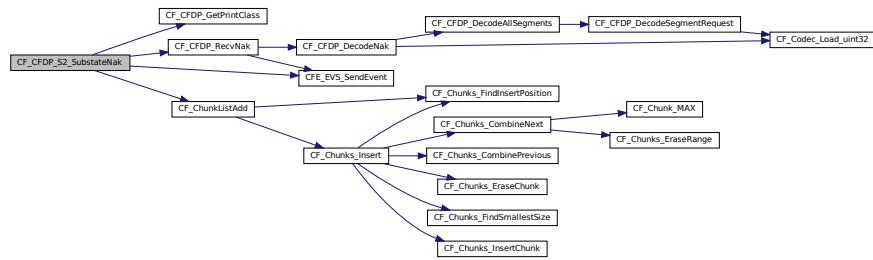
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 246 of file cf\_cfdp\_s.c.

References CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_RecvNak(), CF\_CFDP\_S\_INVALID\_SR\_ERR\_EID, CF\_CFDP\_S\_PDU\_NAK\_ERR\_EID, CF\_ChunkListAdd(), CF\_TRACE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_ChunkWrapper::chunks, CF\_Transaction::chunks, CF\_HkChannel\_Data::counters, CF\_HkRecv::error, CF\_Flags\_Tx::fd\_nak\_pending, CF\_Transaction::flags, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_Logical\_IntHeader::nak, CF\_HkRecv::nak\_segment\_requests, CF\_Logical\_SegmentList::num\_segments, CF\_Logical\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_start, CF\_HkPacket::Payload, CF\_HkCounters::recv, CF\_Logical\_PduNak::segment\_list, CF\_Logical\_SegmentList::segments, CF\_Flags\_Tx::send\_md, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S2\_Recv().

Here is the call graph for this function:



**12.42.2.5 CF\_CFDP\_S\_AckTimerTick()** `void CF_CFDP_S_AckTimerTick (`  
`CF_Transaction_t * txn )`

Perform acknowledgement timer tick (time-based) processing for S transactions.

#### Description

This is invoked as part of overall timer tick processing if the transaction has some sort of acknowledgement pending from the remote.

#### Assumptions, External Events, and Notes:

`txn` must not be NULL

#### Parameters

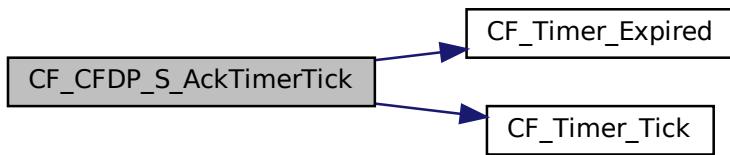
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 576 of file cf\_cfdp\_s.c.

References `CF_Transaction::ack_timer`, `CF_Flags_Common::ack_timer_armed`, `CF_Timer_Expired()`, `CF_Timer_Tick()`, `CF_StateFlags::com`, `CF_Transaction::flags`, and `CF_Transaction::reliable_mode`.

Referenced by `CF_CFDP_S_Tick()`.

Here is the call graph for this function:



**12.42.2.6 CF\_CFDP\_S\_CheckState()** void CF\_CFDP\_S\_CheckState ( CF\_Transaction\_t \* txn )

Run TX state machine for the transaction.

#### Description

Checks flags on the transaction and execute state transitions

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

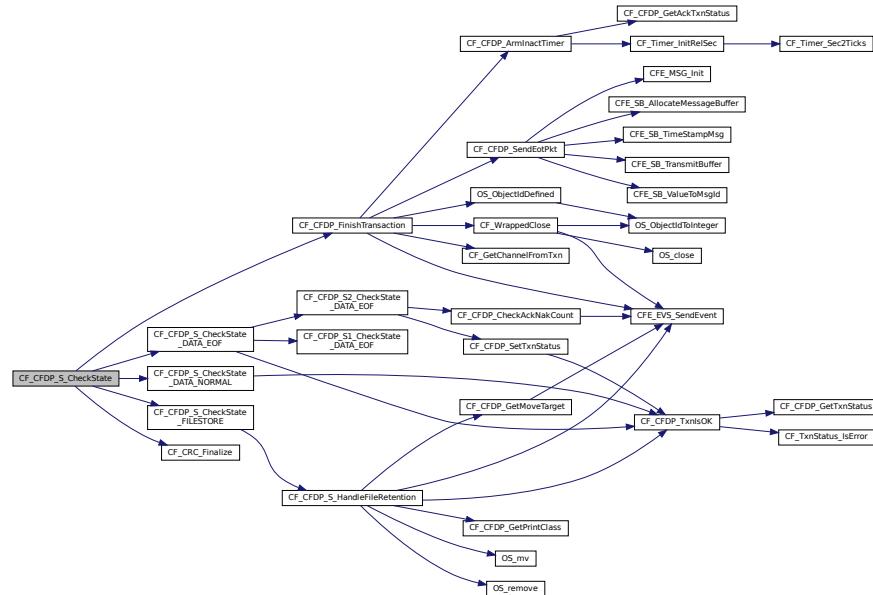
txn	Pointer to the transaction object
-----	-----------------------------------

Definition at line 762 of file cf\_cfdp\_s.c.

References CF\_Flags\_Common::ack\_timer\_armed, CF\_StateData::acknak\_count, CF\_CFDP\_FinishTransaction(), CF\_CFDP\_S\_CheckState\_DATA\_EOF(), CF\_CFDP\_S\_CheckState\_NORMAL(), CF\_CFDP\_S\_CheckState\_FILESTORE(), CF\_CRC\_Finalize(), CF\_TRACE, CF\_TxSubState\_COMPLETE, CF\_TxSubState\_DATA\_EOF, CF\_TxSubState\_DATA\_NORMAL, CF\_TxSubState\_FILESTORE, CF\_StateFlags::com, CF\_Transaction::crc, CF\_Flags\_Common::crc\_complete, CF\_Transaction::flags, CF\_Flags\_Tx::send\_eof, CF\_Transaction::state\_data, CF\_StateData::sub\_state, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_Tick().

Here is the call graph for this function:



**12.42.2.7 CF\_CFDP\_S\_HandleFileRetention()** void CF\_CFDP\_S\_HandleFileRetention ( CF\_Transaction\_t \* txn )

Remove/Move file after transaction.

Determines disposition of local file after file transfer completion.

For a sender:

- If the transfer is successful and the "keep" flag is false, then it applies the local file deletion policy (either delete directly or move to recycle dir)
- If the transfer is not successful or the "keep" flag is true, then do nothing

Assumptions, External Events, and Notes:

#### Parameters

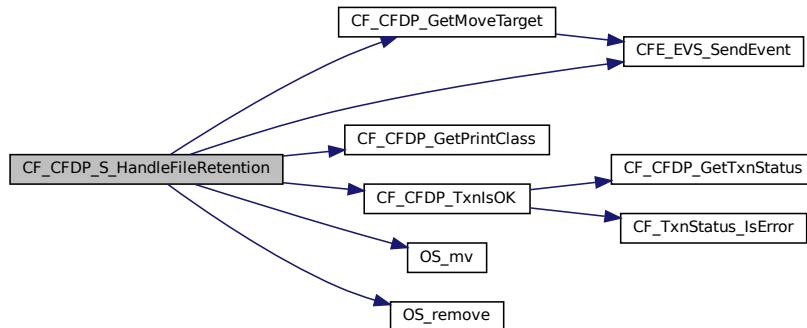
<code>txn</code>	Transaction object pointer
------------------	----------------------------

Definition at line 495 of file cf\_cfdp\_s.c.

References CF\_AppData, CF\_CFDP\_FinDeliveryCode\_COMPLETE, CF\_CFDP\_FinFileStatus\_RETAINED, CF\_CFDP\_GetMoveTarget(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_FILE\_MOVED\_EID, CF\_CFDP\_S\_FILE\_REMOVED\_EID, CF\_CFDP\_TxnIsOK(), CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_MISSION\_MAX\_PATH\_LEN, CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_Flags\_Tx::cmd\_tx, CF\_StateFlags::com, CF\_AppData\_t::config\_table, CF\_ConfigTable::fail\_dir, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::history, CF\_Flags\_Common::is\_complete, CF\_Transaction::keep, CF\_ChannelConfig::move\_dir, OS\_mv(), OS\_remove(), CF\_Transaction::reliable\_mode, CF\_History::seq\_num, CF\_History::src\_eid, CF\_TxnFilenames::src\_filename, CF\_Transaction::state\_data, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_S\_CheckState\_FILESTORE().

Here is the call graph for this function:



#### 12.42.2.8 CF\_CFDP\_S\_Init()

```
void CF_CFDP_S_Init (
    CF_Transaction_t * txn )
```

Initialize a transaction structure for S.

Assumptions, External Events, and Notes:

txn must not be NULL.

## Parameters

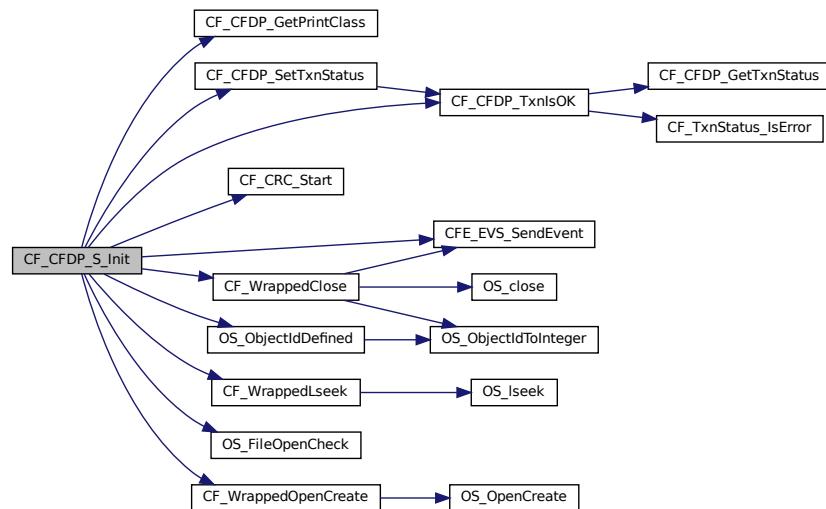
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 402 of file cf\_cfdp\_s.c.

References CF\_AppData, CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_ALREADY\_OPEN\_ERR\_EID, CF\_CFDP\_S\_OPEN\_ERR\_EID, CF\_CFDP\_S\_SEEK\_BEG\_ERR\_EID, CF\_CFDP\_S\_SEEK\_END\_ERR\_EID, CF\_CFDP\_SetTxnStatus(), CF\_CFDP\_TxnIsOK(), CF\_CRC\_Start(), CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_WrappedClose(), CF\_WrappedLseek(), CF\_WrappedOpenCreate(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_HkFault::file\_open, CF\_HkFault::file\_seek, CF\_Transaction::flags, CF\_History::fnames, CF\_Transaction::fsize, CF\_Transaction::history, CF\_AppData\_t::hk, OS\_FILE\_FLAG\_NONE, OS\_FileOpenCheck(), OS\_OBJECT\_ID\_UNDEFINED, OS\_ObjectIdDefined(), OS\_READ\_ONLY, OS\_SEEK\_END, OS\_SEEK\_SET, OS\_SUCCESS, CF\_HkPacket::Payload, CF\_Flags\_Tx::send\_md, CF\_History::seq\_num, CF\_History::src\_eid, CF\_TxnFilenames::src\_filename, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



### 12.42.2.9 CF\_CFDP\_S\_SendFileData() `CFE_Status_t CF_CFDP_S_SendFileData (`

```

CF_Transaction_t * txn,
uint32 foffs,
uint32 bytes_to_read,
uint8 calc_crc )

```

Helper function to populate the PDU with file data and send it.

#### Description

This function checks the file offset cache and if the desired location is where the file offset is, it can skip a seek() call. The file is read into the filedatal PDU and then the PDU is sent.

## Assumptions, External Events, and Notes:

txn must not be NULL.

## Returns

The number of bytes sent in the file data PDU (CFE\_SUCCESS, i.e. 0, if no bytes were processed), or CF\_ERROR on error

## Parameters

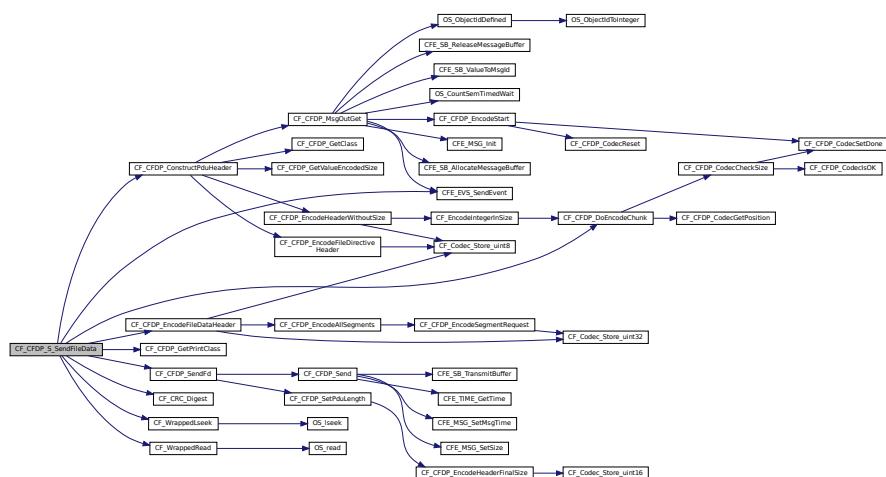
<i>txn</i>	Pointer to the transaction object
<i>foffs</i>	Position in file to send data from
<i>bytes_to_read</i>	Number of bytes to send (maximum)
<i>calc_crc</i>	Enable CRC/Checksum calculation

Definition at line 49 of file cf\_cfdp\_s.c.

References CF\_StateData::cached\_pos, CF\_AppData, CF\_CFDP\_ConstructPduHeader(), CF\_CFDP\_DoEncodeChunk(), CF\_CFDP\_EncodeFileDataHeader(), CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_READ\_ERR\_EID, CF\_CFDP\_S\_SEEK\_FD\_ERR\_EID, CF\_CFDP\_SendFd(), CF\_CODEC\_GET\_REMAIN, CF\_CRC\_Digest(), CF\_ERROR, CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_WrappedLseek(), CF\_WrappedRead(), CFE\_EVS\_EventType\_Error, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_AppData\_t::config\_table, CF\_HkChannel\_Data::counters, CF\_Transaction::crc, CF\_Logical\_PduFileDataHeader::data\_len, CF\_Logical\_PduFileDataHeader::data\_ptr, CF\_HkCounters::fault, CF\_Transaction::fd, CF\_Logical\_InterruptHeader::fd, CF\_HkSent::file\_data\_bytes, CF\_HkFault::file\_read, CF\_HkFault::file\_seek, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Logical\_PduBuffer::int\_header, CF\_ConfigTable::local\_eid, CF\_Logical\_PduFileDataHeader::offset, OS\_SEEK\_SET, CF\_ConfigTable::outgoing\_file\_chunk\_size, CF\_HkPacket::Payload, CF\_Logical\_PduBuffer::pdu\_header, CF\_History::peer\_eid, CF\_Logical\_PduBuffer::penc, CF\_Logical\_PduHeader::segment\_meta\_flag, CF\_HkCounters::sent, CF\_History::seq\_num, CF\_History::src\_eid, and CF\_Transaction::state\_data.

Referenced by CF\_CFDP\_S\_SubstateSendFileData(), and CF\_CFDP\_S\_Tick\_Nak().

Here is the call graph for this function:



**12.42.2.10 CF\_CFDP\_S\_SubstateEarlyFin()** void CF\_CFDP\_S\_SubstateEarlyFin (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

A FIN was received before file complete, so abandon the transaction.

#### Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

#### Parameters

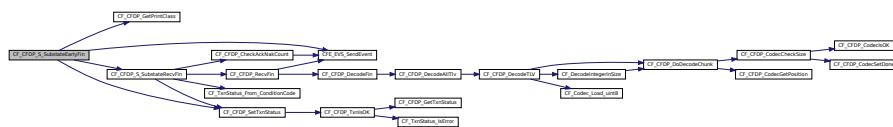
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 190 of file cf\_cfdp\_s.c.

References CF\_CFDP\_GetPrintClass(), CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID, CF\_CFDP\_S\_SubstateRecvFin(), CF\_CFDP\_SetTxnStatus(), CF\_TxnStatus\_EARLY\_FIN, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::history, CF\_History::seq\_num, and CF\_History::src\_eid.

Referenced by CF\_CFDP\_S1\_Recv(), and CF\_CFDP\_S2\_Recv().

Here is the call graph for this function:



#### 12.42.2.11 CF\_CFDP\_S\_SubstateRecvFin() void CF\_CFDP\_S\_SubstateRecvFin (

```
CF_Transaction_t * txn,
CF_Logical_PduBuffer_t * ph )
```

S2 received FIN, so set flag to send FIN-ACK.

#### Assumptions, External Events, and Notes:

txn must not be NULL. ph must not be NULL.

#### Parameters

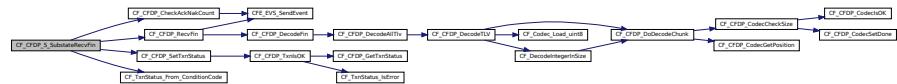
<i>txn</i>	Pointer to the transaction object
<i>ph</i>	Pointer to the PDU information

Definition at line 211 of file cf\_cfdp\_s.c.

References CF\_Logical\_PduFin::cc, CF\_CFDP\_CheckAckNakCount(), CF\_CFDP\_RecvFin(), CF\_CFDP\_SetTxnStatus(), CF\_TRACE, CF\_TxnStatus\_From\_ConditionCode(), CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED, CFE\_SUCCESS, CF\_Logical\_PduFin::delivery\_code, CF\_Logical\_PduFin::file\_status, CF\_Logical\_IntHeader::fin, CF\_Flags\_Tx::fin\_count, CF\_StateData::fin\_dc, CF\_StateData::fin\_fs, CF\_Transaction::flags, CF\_Logical\_PduBuffer::int\_header, CF\_StateData::peer\_cc, CF\_Transaction::state\_data, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_RecvHold(), CF\_CFDP\_S1\_Recv(), CF\_CFDP\_S2\_Recv(), and CF\_CFDP\_S\_SubstateEarlyFin().

Here is the call graph for this function:



**12.42.2.12 CF\_CFDP\_S\_SubstateSendFileData()** void CF\_CFDP\_S\_SubstateSendFileData (

```
CF_Transaction_t * txn )
```

Standard state function to send the next file data PDU for active transaction.

## Description

During the transfer of active transaction file data PDUs, the file offset is saved. This function sends the next chunk of data. If the file offset equals the file size, then transition to the EOF state.

## **Assumptions, External Events, and Notes:**

txn must not be NULL.

## Parameters

<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 163 of file cf\_cfdp\_s.c.

References CF\_CFDP\_S\_SendFileData(), CF\_CFDP\_SetTxnStatus(), CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, CF\_TxnStatus\_READ\_FAILURE, CF\_Transaction::foffs, and CF\_Transaction::fsize.

Referenced by CF\_CFDP\_S\_Tick\_NewData().

Here is the call graph for this function:

**12.42.2.13 CF\_CFDP\_S\_Tick()** void CF\_CFDP\_S\_Tick (

CF Transaction t \* txn )

Perform tick (time-based) processing for S transactions.

#### Description

This function is called on every transaction by the engine on every CF wakeup. This is where flags are checked to send EOF or FIN-ACK. If nothing else is sent, it checks to see if a NAK retransmit must occur.

#### Assumptions, External Events, and Notes:

txn must not be NULL. cont is unused, so may be NULL

#### Parameters

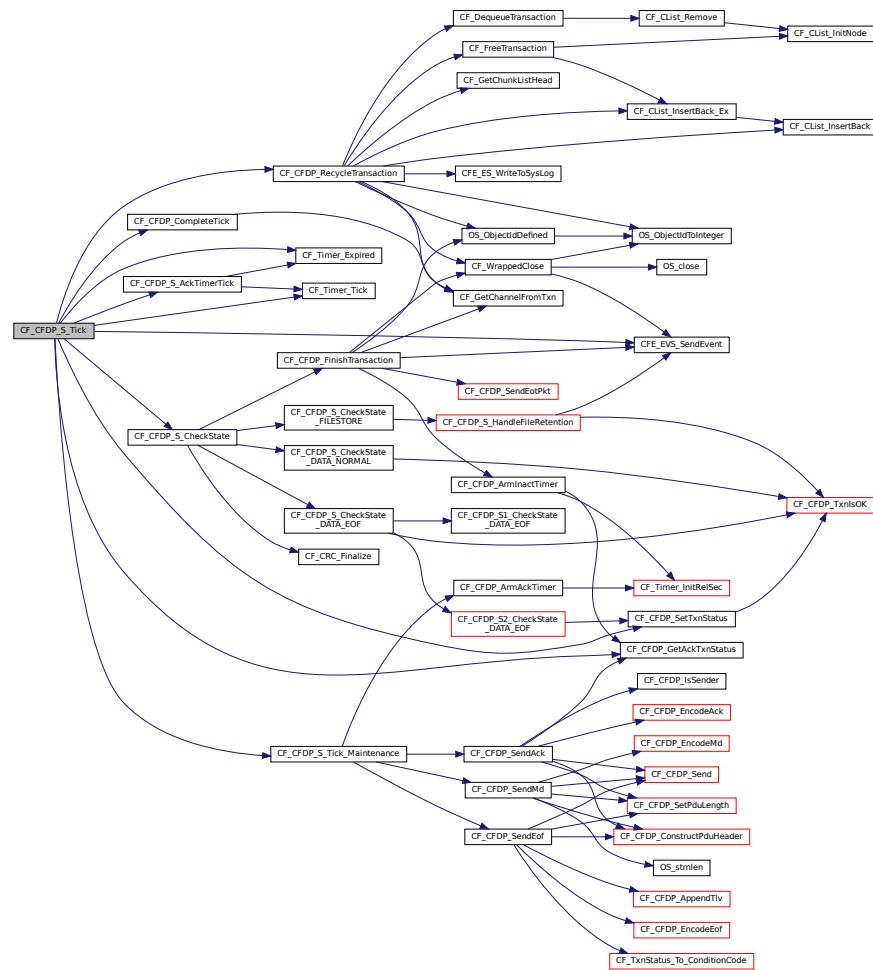
<i>txn</i>	Pointer to the transaction object
------------	-----------------------------------

Definition at line 824 of file cf\_cfdp\_s.c.

References CF\_AppData, CF\_CFDP\_AckTxnStatus\_ACTIVE, CF\_CFDP\_CompleteTick(), CF\_CFDP\_GetAckTxnStatus(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S\_AckTimerTick(), CF\_CFDP\_S\_CheckState(), CF\_CFDP\_S\_INACT\_TIMER\_ERR\_EID, CF\_CFDP\_S\_Tick\_Maintenance(), CF\_CFDP\_SetTxnStatus(), CF\_Timer\_Expired(), CF\_Timer\_Tick(), CF\_TxnState\_HOLD, CF\_TxnStatus\_INACTIVITY\_DETECTED, CF\_TxSubState\_DATA\_NORMAL, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_StateFlags::com, CF\_HkChannel\_Data::counters, CF\_HkCounters::fault, CF\_Transaction::flags, CF\_Transaction::history, CF\_AppData\_t::hk, CF\_Flags\_Common::inactivity\_fired, CF\_HkFault::inactivity\_timer, CF\_Transaction::inactivity\_timer, CF\_HkPacket::Payload, CF\_History::seq\_num, CF\_History::src\_eid, CF\_Transaction::state, CF\_Transaction::state\_data, and CF\_StateData::sub\_state.

Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



#### 12.42.2.14 CF\_CFDP\_S\_Tick\_Maintenance()

```
void CF_CFDP_S_Tick_Maintenance (
    CF_Transaction_t * txn )
```

Generate protocol messages for TX transactions.

##### Description

This function is called at tick processing time to send pending protocol messages such as MD, EOF, and FIN-ACK

These messages are considered higher priority than any file data

##### Assumptions, External Events, and Notes:

`txn` must not be NULL.

##### Parameters

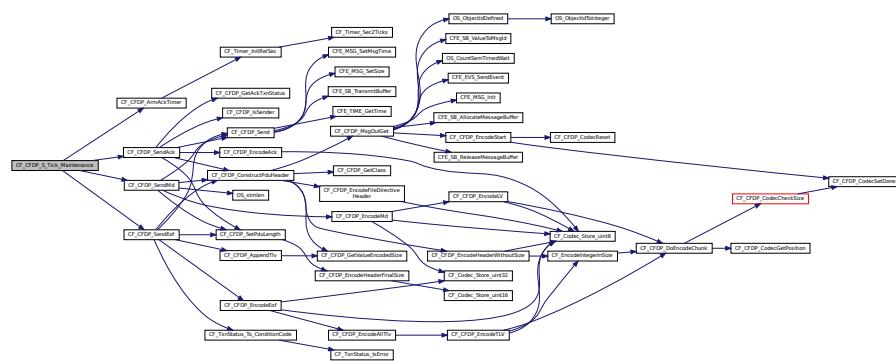
<code>txn</code>	Pointer to the transaction object
------------------	-----------------------------------

Definition at line 890 of file cf\_cfdp\_s.c.

References CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_FileDirective\_FIN, CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendMd(), CFE\_SUCCESS, CF\_Flags\_Tx::fin\_ack\_count, CF\_Flags\_Tx::fin\_count, CF\_Transaction::flags, CF\_Transaction::reliable\_mode, CF\_Flags\_Tx::send\_eof, CF\_Flags\_Tx::send\_md, and CF\_StateFlags::tx.

Referenced by CF CFDP S Tick().

Here is the call graph for this function:



**12.42.2.15 CF\_CFDP\_S\_Tick\_Nak()** void CF\_CFDP\_S\_Tick\_Nak ( CF\_Transaction\_t \* txn )

Generate file data PDUs from NAKs.

generated the data? Does it change that the peer has arrived?

Responding to NAK is considered higher priority than sending new file data, but lower priority than maintenance PDSS.

## Assumptions, External Events, and Notes.

*left* must not be NULL.

## Parameters

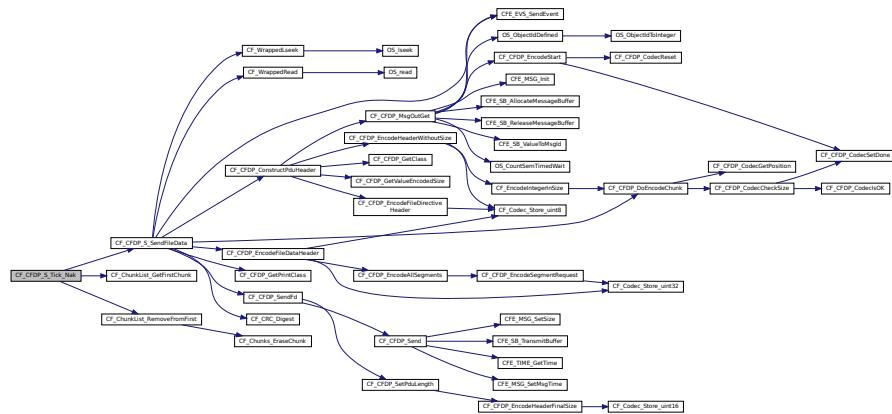
*txn* Pointer to the transaction object

Definition at line 940 of file cf\_cfdp\_s.c.

References CF\_CFDP\_S\_SendFileData(), CF\_ChunkList\_GetFirstChunk(), CF\_ChunkList\_RemoveFromFirst(), CF\_ChunkWrapper::chunks, CF\_Transaction::chunks, CF\_Flags\_Tx::fd\_nak\_pending, CF\_Transaction::flags, CF\_Chunk::offset, CF\_Chunk::size, and CF\_StateFlags::tx.

Referenced by CF\_CFDP\_TickTransactions().

Here is the call graph for this function:



## 12.43 apps/cf/fsw/src(cf\_cfdp\_sbintf.c File Reference)

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_app.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_cfdp_r.h"
#include "cf_cfdp_s.h"
#include "cf_cfdp_sbintf.h"
#include <string.h>
#include "cf_assert.h"
```

# Functions

- **CF\_Logical\_PduBuffer\_t \* CF\_CFDP\_MsgOutGet (const CF\_Transaction\_t \*txn, bool silent)**  
*Obtain a message buffer to construct a PDU inside.*
  - void **CF\_CFDP\_Send (uint8 chan\_num, const CF\_Logical\_PduBuffer\_t \*ph)**  
*Sends the current output buffer via the software bus.*
  - void **CF\_CFDP\_ReceiveMessage (CF\_Channel\_t \*chan)**  
*Process received message on channel PDU input pipe.*

### **12.43.1 Detailed Description**

This is the interface to the CFE Software Bus for CF transmit/recv. Specifically this implements 3 functions used by the CFDP engine:

- `CF_CFDP_MsgOutGet()` - gets a buffer prior to transmitting
  - `CF_CFDP_Send()` - sends the buffer from `CF_CFDP_MsgOutGet()`
  - `CF_CFDP_ReceiveMessage()` - gets a received message

These functions were originally part of the CFDP engine itself but were split into a separate file, both to improve testability as well as to (potentially) allow interfaces to message/packet services other than the CFE software bus. However, Note that in this version, there is still a fair amount of CFDP engine mixed into these functions that would have to be isolated. Also note that the creation and deletion of SB pipes is not yet moved into this file.

### 12.43.2 Function Documentation

**12.43.2.1 CF\_CFDP\_MsgOutGet()** `CF_Logical_PduBuffer_t* CF_CFDP_MsgOutGet (`  
    `const CF_Transaction_t * txn,`  
    `bool silent )`

Obtain a message buffer to construct a PDU inside.

#### Description

This performs the handshaking via semaphore with the consumer of the PDU. If the semaphore can be obtained, a software bus buffer is obtained and it is returned. If the semaphore is unavailable, then the current transaction is remembered for next engine cycle. If silent is true, then the event message is not printed in the case of no buffer available.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

<code>txn</code>	Pointer to the transaction object
<code>silent</code>	If true, suppresses error events if no message can be allocated

#### Returns

Pointer to a CF\_Logical\_PduBuffer\_t on success.

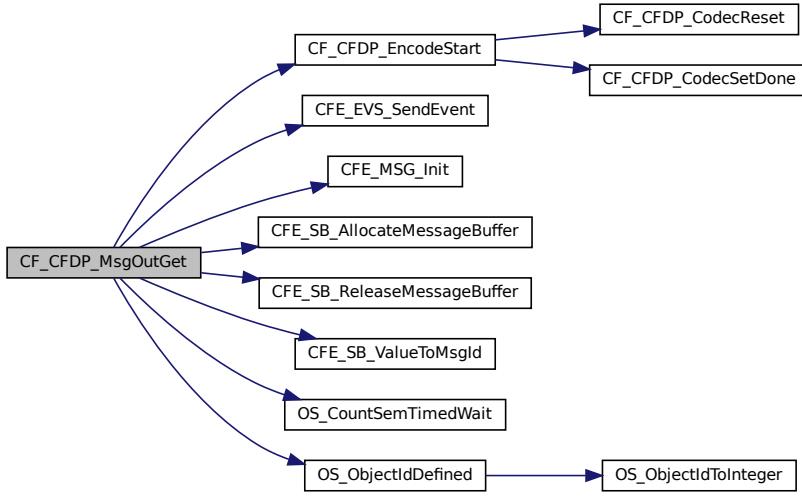
#### Return values

<code>NULL</code>	on error
-------------------	----------

Definition at line 61 of file cf\_cfdp\_sbintf.c.

References CF\_AppData, CF\_CFDP\_EncodeStart(), CF\_CFDP\_NO\_MSG\_ERR\_EID, CF\_MAX\_PDU\_SIZE, CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_Init(), CFE\_SB\_AllocateMessageBuffer(), CFE\_SB\_ReleaseMessageBuffer(), CFE\_SB\_ValueToMsgId(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_StateFlags::com, CF\_AppData\_t::config\_table, CF\_Output::encode, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_HkChannel\_Data::frozen, CF\_AppData\_t::hk, CF\_ChannelConfig::max\_outgoing\_messages\_per\_wakeup, CF\_ChannelConfig::mid\_output, CF\_Output::msg, CFE\_SB\_Msg::Msg, OS\_CountSemTimedWait(), OS\_ObjectIdDefined(), OS\_SUCCESS, CF\_Engine::out, CF\_Channel::outgoing\_counter, CF\_HkPacket::Payload, CF\_Channel::sem\_id, CF\_Flags\_Common::suspended, CF\_Channel::tx\_blocked, and CF\_Output::tx\_pdudata.  
Referenced by CF\_CFDP\_ConstructPduHeader().

Here is the call graph for this function:



#### 12.43.2.2 CF\_CFDP\_ReceiveMessage()

```
void CF_CFDP_ReceiveMessage (
    CF_Channel_t * chan )
```

Process received message on channel PDU input pipe.

**Assumptions, External Events, and Notes:**

chan must be a member of the array within the CF\_AppData global object

#### Parameters

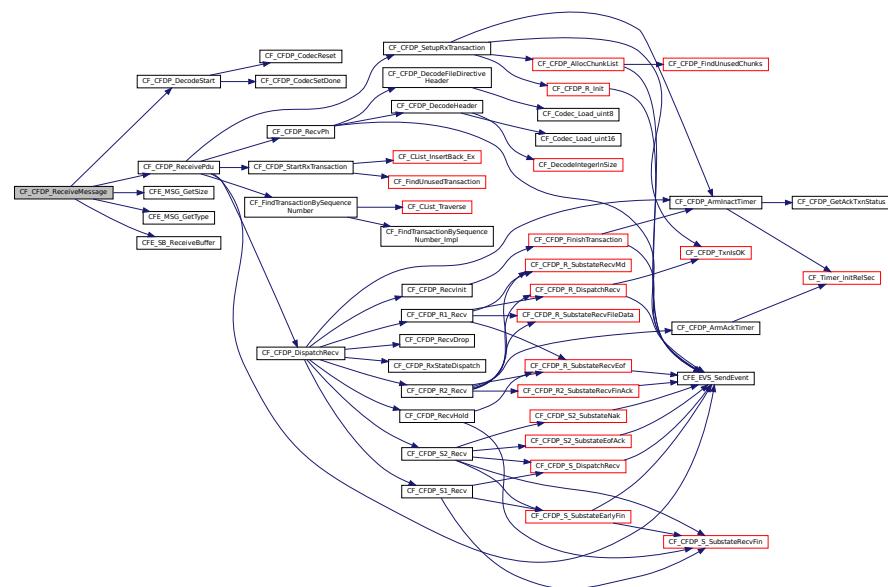
chan	Channel to receive message on
------	-------------------------------

Definition at line 179 of file cf\_cfdp\_sbintf.c.

References CF\_AppData, CF\_CFDP\_DecodeStart(), CF\_CFDP\_ReceivePdu(), CF\_PDU\_ENCAPSULATION\_EXTRAS\_TRAILING\_BYTES, CF\_PERF\_ID\_PDURCVD, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_MSG\_GetSize(), CFE\_MSG\_GetType(), CFE\_MSG\_Type\_Invalid, CFE\_MSG\_Type\_Tlm, CFE\_SB\_POLL, CFE\_SB\_ReceiveBuffer(), CFE\_SUCCESS, CF\_ConfigTable::chan, CF\_Engine::channels, CF\_AppData\_t::config\_table, CF\_Input::decode, CF\_AppData\_t::engine, CF\_Engine::in, CFE\_SB\_Msg::Msg, CF\_Channel::pipe, CF\_ChannelConfig::rx\_max\_messages\_per\_wakeup, and CF\_Input::rx\_pdudata.

Referenced by CF\_CFDP\_CycleEngine().

Here is the call graph for this function:



**12.43.2.3 CF\_CFDP\_Send()** void CF\_CFDP\_Send ( uint8 chan\_num,  
const CF\_Logical\_PduBuffer\_t \* ph )

Sends the current output buffer via the software bus.

#### Assumptions, External Events, and Notes:

The PDU in the output buffer is ready to transmit.

#### Parameters

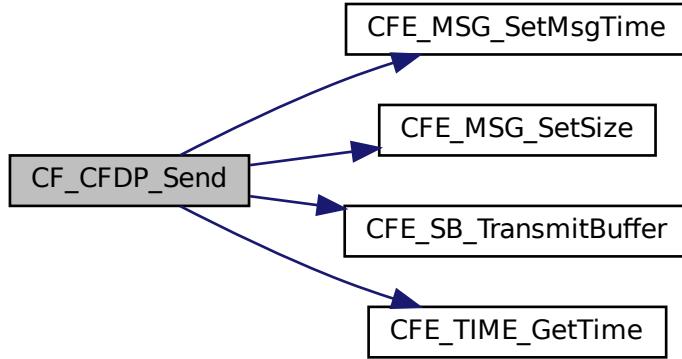
chan_num	Channel number for statistics/accounting purposes
ph	Pointer to PDU buffer to send

Definition at line 151 of file cf\_cfdp\_sbintf.c.

References CF\_AppData, CF\_Assert, CF\_NUM\_CHANNELS, CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES, CFE\_MSG\_SetMsgTime(), CFE\_MSG\_SetSize(), CFE\_SB\_TransmitBuffer(), CFE\_TIME\_GetTime(), CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduHeader::data\_encoded\_length, CF\_AppData\_t::engine, CF\_Logical\_PduHeader::header\_encoded\_length, CF\_AppData\_t::hk, CF\_Output::msg, CFE\_SB\_Msg::Msg, CF\_Engine::out, CF\_HkPacket::Payload, CF\_HkSent::pdu, CF\_Logical\_PduBuffer::pdu\_header, and CF\_HkCounters::sent.

Referenced by CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFd(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), and CF\_CFDP\_SendNak().

Here is the call graph for this function:



## 12.44 apps/cf/fsw/src/cf\_cfdp\_sbintf.h File Reference

```
#include "cf_cfdp_types.h"
```

### Data Structures

- struct `CF_PduCmdMsg`  
*PDU command encapsulation structure.*
- struct `CF_PduTlmMsg`  
*PDU send encapsulation structure.*

### TypeDefs

- typedef struct `CF_PduCmdMsg` `CF_PduCmdMsg_t`  
*PDU command encapsulation structure.*
- typedef struct `CF_PduTlmMsg` `CF_PduTlmMsg_t`  
*PDU send encapsulation structure.*

### Functions

- `CF_Logical_PduBuffer_t * CF_CFDP_MsgOutGet (const CF_Transaction_t *txnid, bool silent)`  
*Obtain a message buffer to construct a PDU inside.*
- `void CF_CFDP_Send (uint8 chan_num, const CF_Logical_PduBuffer_t *ph)`  
*Sends the current output buffer via the software bus.*
- `void CF_CFDP_ReceiveMessage (CF_Channel_t *chan)`  
*Process received message on channel PDU input pipe.*

### 12.44.1 Detailed Description

This is the interface to the CFE Software Bus for PDU transmit/recv.  
It may serve as a future point of abstraction to interface with message passing interfaces other than the CFE software bus, if necessary.

### 12.44.2 Typedef Documentation

#### 12.44.2.1 CF\_PduCmdMsg\_t `typedef struct CF_PduCmdMsg CF_PduCmdMsg_t`

PDU command encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "command" encapsulation (even though these are not really commands).

##### Note

this is only the definition of the header. In reality all messages are larger than this, up to CF\_MAX\_PDU\_SIZE.

#### 12.44.2.2 CF\_PduTlmMsg\_t `typedef struct CF_PduTlmMsg CF_PduTlmMsg_t`

PDU send encapsulation structure.

This encapsulates a CFDP PDU into a format that is sent or received over the software bus, adding "telemetry" encapsulation (even though these are not really telemetry items).

##### Note

this is only the definition of the header. In reality all messages are larger than this, up to CF\_MAX\_PDU\_SIZE.

### 12.44.3 Function Documentation

#### 12.44.3.1 CF\_CFDP\_MsgOutGet() `CF_Logical_PduBuffer_t* CF_CFDP_MsgOutGet (` `const CF_Transaction_t * txn,` `bool silent )`

Obtain a message buffer to construct a PDU inside.

##### Description

This performs the handshaking via semaphore with the consumer of the PDU. If the semaphore can be obtained, a software bus buffer is obtained and it is returned. If the semaphore is unavailable, then the current transaction is remembered for next engine cycle. If silent is true, then the event message is not printed in the case of no buffer available.

##### Assumptions, External Events, and Notes:

txn must not be NULL.

##### Parameters

<code>txn</code>	Pointer to the transaction object
<code>silent</code>	If true, suppresses error events if no message can be allocated

**Returns**

Pointer to a CF\_Logical\_PduBuffer\_t on success.

**Return values**

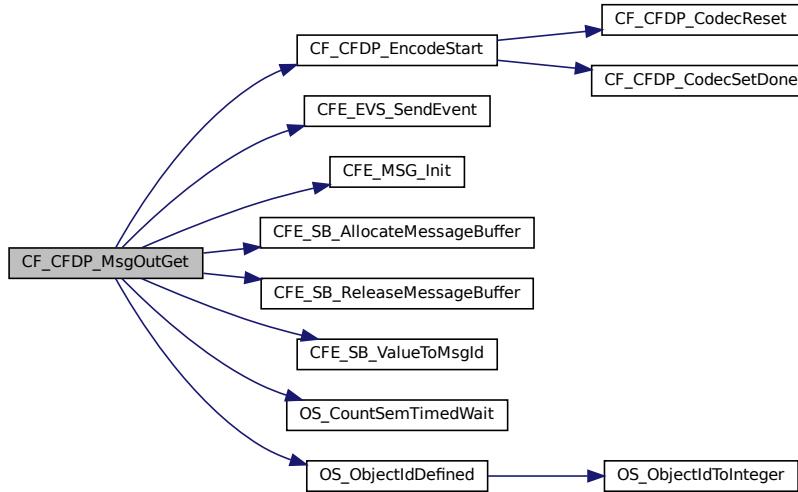
<code>NULL</code>	on error
-------------------	----------

Definition at line 61 of file cf\_cfdp\_sbintf.c.

References CF\_AppData, CF\_CFDP\_EncodeStart(), CF\_CFDP\_NO\_MSG\_ERR\_EID, CF\_MAX\_PDU\_SIZE, CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_Init(), CFE\_SB\_AllocateMessageBuffer(), CFE\_SB\_ReleaseMessageBuffer(), CFE\_SB\_ValueToMsgId(), CF\_ConfigTable::chan, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_StateFlags::com, CF\_AppData\_t::config\_table, CF\_Output::encode, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_HkChannel\_Data::frozen, CF\_AppData\_t::hk, CF\_ChannelConfig::max\_outgoing\_messages\_per\_wakeup, CF\_ChannelConfig::mid\_output, CF\_Output::msg, CFE\_SB\_Msg::Msg, OS\_CountSemTimedWait(), OS\_ObjectIdDefined(), OS\_SUCCESS, CF\_Engine::out, CF\_Channel::outgoing\_counter, CF\_HkPacket::Payload, CF\_Channel::sem\_id, CF\_Flags\_Common::suspended, CF\_Channel::tx\_blocked, and CF\_Output::tx\_pdudata.

Referenced by CF\_CFDP\_ConstructPduHeader().

Here is the call graph for this function:



### 12.44.3.2 CF\_CFDP\_ReceiveMessage()

```
void CF_CFDP_ReceiveMessage (
    CF_Channel_t * chan )
```

Process received message on channel PDU input pipe.

#### Assumptions, External Events, and Notes:

chan must be a member of the array within the CF\_AppData global object

## Parameters

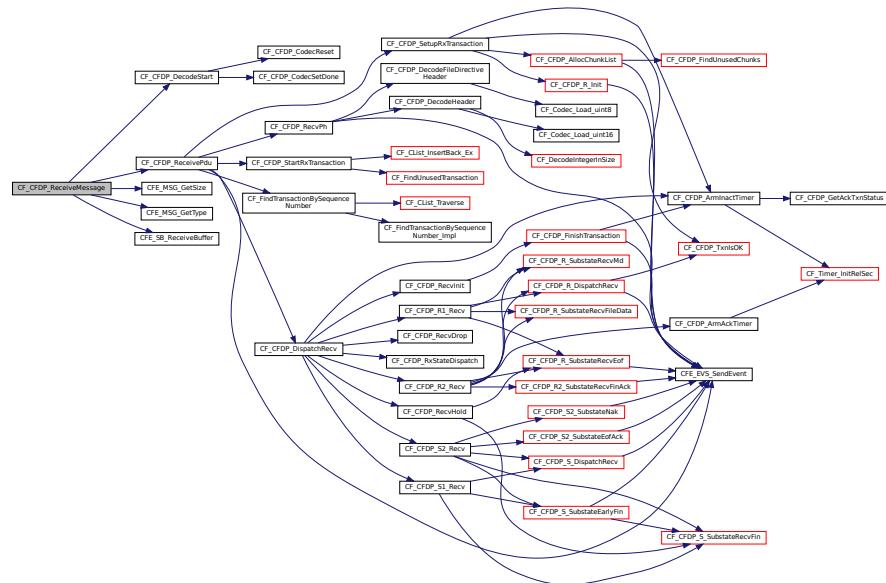
*chan* Channel to receive message on

Definition at line 179 of file cf\_cfdp\_sbintf.c.

References CF\_AppData, CF\_CFDP\_DecodeStart(), CF\_CFDP\_ReceivePdu(), CF\_PDU\_ENCAPSULATION\_EXTRASPACE, CF\_EXTRA\_TRAILING\_BYTES, CF\_PERF\_ID\_PDURCVD, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_MSG\_GetSize(), CFE\_MSG\_GetType(), CFE\_MSG\_Type\_Invalid, CFE\_MSG\_Type\_Tlm, CFE\_SB\_POLL, CFE\_SB\_ReceiveBuffer(), CFE\_SUCCESS, CF\_ConfigTable::chan, CF\_Engine::channels, CF\_AppData\_t::config\_table, CF\_Input::decode, CF\_AppData\_t::engine, CF\_Engine::in, CFE\_SB\_Msg::Msg, CF\_Channel::pipe, CF\_ChannelConfig::rx\_max\_messages\_per\_wakeup, and CF\_Input::rx\_pdudata.

Referenced by CF\_CFDP\_CycleEngine().

Here is the call graph for this function:



```
12.44.3.3 CF_CFDP_Send() void CF_CFDP_Send (  
    uint8 chan_num,  
    const CF_Logical_PduBuffer_t * ph
```

Sends the current output buffer via the software bus.

The T-DO in the output buffer is ready to transmit.

#### **Parameters**

---

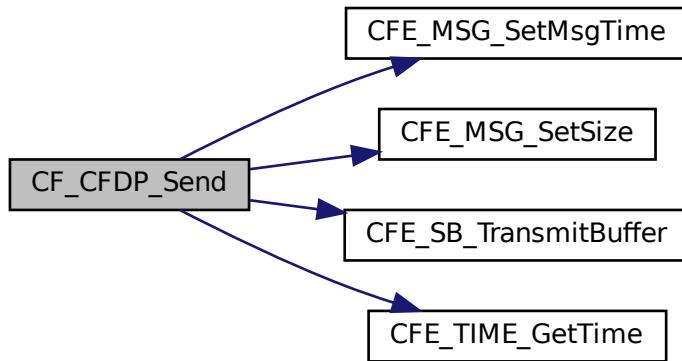
<i>chan_num</i>	Channel number for statistics/accounting purposes
<i>ph</i>	Pointer to PDU buffer to send

Definition at line 151 of file cf\_cfdp\_sbintf.c.

References CF\_AppData, CF\_Assert, CF\_NUM\_CHANNELS, CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES, CFE\_MSG\_SetMsgTime(), CFE\_MSG\_SetSize(), CFE\_SB\_TransmitBuffer(), CFE\_TIME\_GetTime(), CF\_HkPacket\_Payload::channel\_hk, CF\_HkChannel\_Data::counters, CF\_Logical\_PduHeader::data\_encoded\_length, CF\_AppData\_t::engine, CF\_Logical\_PduHeader::header\_encoded\_length, CF\_AppData\_t::hk, CF\_Output::msg, CFE\_SB\_Msg::Msg, CF\_Engine::out, CF\_HkPacket::Payload, CF\_HkSent::pdu, CF\_Logical\_PduBuffer::pdu\_header, and CF\_HkCounters::sent.

Referenced by CF\_CFDP\_SendAck(), CF\_CFDP\_SendEof(), CF\_CFDP\_SendFd(), CF\_CFDP\_SendFin(), CF\_CFDP\_SendMd(), and CF\_CFDP\_SendNak().

Here is the call graph for this function:



## 12.45 apps/cf/fsw/src(cf\_cfdp\_types.h File Reference

```
#include "common_types.h"
#include "cf_cfdp_pdu.h"
#include "cf_extern_typedefs.h"
#include "cf_platform_cfg.h"
#include "cf_msg.h"
#include "cf_clist.h"
#include "cf_chunk.h"
#include "cf_timer.h"
#include "cf_crc.h"
#include "cf_codec.h"
```

### Data Structures

- struct [CF\\_History](#)  
*CF History entry.*
- struct [CF\\_ChunkWrapper](#)  
*Wrapper around a CF\_ChunkList\_t object.*
- struct [CF\\_Playback](#)  
*CF Playback entry.*
- struct [CF\\_Poll](#)

- struct **CF\_Flags\_Common**  
*Data that applies to all types of transactions.*
- struct **CF\_Flags\_Rx**  
*Flags that apply to receive transactions.*
- struct **CF\_Flags\_Tx**  
*Flags that apply to send transactions.*
- union **CF\_StateFlags**  
*Summary of all possible transaction flags (tx and rx)*
- struct **CF\_StateData**  
*Summary of all possible transaction state information (tx and rx)*
- struct **CF\_Transaction**  
*Transaction state object.*
- struct **CF\_Channel**  
*Channel state object.*
- struct **CF\_Output**  
*CF engine output state.*
- struct **CF\_Input**  
*CF engine input state.*
- struct **CF\_Engine**  
*An engine represents a pairing to a local EID.*

## Macros

- #define **CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL**  
*Maximum possible number of transactions that may exist on a single CF channel.*
- #define **CF\_NUM\_TRANSACTIONS** (**CF\_NUM\_CHANNELS** \* **CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL**)  
*Maximum possible number of transactions that may exist in the CF application.*
- #define **CF\_NUM\_HISTORIES** (**CF\_NUM\_CHANNELS** \* **CF\_NUM\_HISTORIES\_PER\_CHANNEL**)  
*Maximum possible number of history entries that may exist in the CF application.*
- #define **CF\_NUM\_CHUNKS\_ALL\_CHANNELS** (**CF\_TOTAL\_CHUNKS** \* **CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL**)  
*Maximum possible number of chunk entries that may exist in the CF application.*

## Typedefs

- typedef struct **CF\_History** **CF\_History\_t**  
*CF History entry.*
- typedef struct **CF\_ChunkWrapper** **CF\_ChunkWrapper\_t**  
*Wrapper around a **CF\_ChunkList\_t** object.*
- typedef struct **CF\_Playback** **CF\_Playback\_t**  
*CF Playback entry.*
- typedef struct **CF\_Poll** **CF\_Poll\_t**  
*CF Poll entry.*
- typedef struct **CF\_Flags\_Common** **CF\_Flags\_Common\_t**  
*Data that applies to all types of transactions.*
- typedef struct **CF\_Flags\_Rx** **CF\_Flags\_Rx\_t**  
*Flags that apply to receive transactions.*
- typedef struct **CF\_Flags\_Tx** **CF\_Flags\_Tx\_t**

*Flags that apply to send transactions.*

- **typedef union CF\_StateFlags CF\_StateFlags\_t**  
*Summary of all possible transaction flags (tx and rx)*
- **typedef struct CF\_StateData CF\_StateData\_t**  
*Summary of all possible transaction state information (tx and rx)*
- **typedef struct CF\_Transaction CF\_Transaction\_t**  
*Transaction state object.*
- **typedef struct CF\_Channel CF\_Channel\_t**  
*Channel state object.*
- **typedef struct CF\_Output CF\_Output\_t**  
*CF engine output state.*
- **typedef struct CF\_Input CF\_Input\_t**  
*CF engine input state.*
- **typedef struct CF\_Engine CF\_Engine\_t**  
*An engine represents a pairing to a local EID.*

## Enumerations

- **enum CF\_TxnState\_t {**  
**CF\_TxnState\_UNDEF = 0 , CF\_TxnState\_INIT = 1 , CF\_TxnState\_R1 = 2 , CF\_TxnState\_S1 = 3 ,**  
**CF\_TxnState\_R2 = 4 , CF\_TxnState\_S2 = 5 , CF\_TxnState\_DROP = 6 , CF\_TxnState\_HOLD = 7 ,**  
**CF\_TxnState\_INVALID = 8 }**  
*High-level state of a transaction.*
- **enum CF\_TxSubState\_t {**  
**CF\_TxSubState\_DATA\_NORMAL = 0 , CF\_TxSubState\_DATA\_EOF = 1 , CF\_TxSubState\_FILESTORE = 2 ,**  
**CF\_TxSubState\_COMPLETE = 3 ,**  
**CF\_TxSubState\_NUM\_STATES = 4 }**  
*Sub-state of a send file transaction.*
- **enum CF\_RxSubState\_t {**  
**CF\_RxSubState\_DATA\_NORMAL = 0 , CF\_RxSubState\_DATA\_EOF = 1 , CF\_RxSubState\_VALIDATE = 2 ,**  
**CF\_RxSubState\_FILESTORE = 3 ,**  
**CF\_RxSubState\_FINACK = 4 , CF\_RxSubState\_COMPLETE = 5 , CF\_RxSubState\_NUM\_STATES = 6 }**  
*Sub-state of a receive file transaction.*
- **enum CF\_Direction\_t { CF\_Direction\_RX = 0 , CF\_Direction\_TX = 1 , CF\_Direction\_NUM = 2 }**  
*Direction identifier.*
- **enum CF\_TxnStatus\_t {**  
**CF\_TxnStatus\_UNDEFINED = -1 , CF\_TxnStatus\_NO\_ERROR = CF\_CFDP\_ConditionCode\_NO\_ERROR ,**  
**CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED = CF\_CFDP\_ConditionCode\_POS\_ACK\_LIMIT\_REACHED ,**  
**CF\_TxnStatus\_KEEP\_ALIVE\_LIMIT\_REACHED = CF\_CFDP\_ConditionCode\_KEEP\_ALIVE\_LIMIT\_REACHED ,**  
**CF\_TxnStatus\_INVALID\_TRANSMISSION\_MODE = CF\_CFDP\_ConditionCode\_INVALID\_TRANSMISSION\_MODE ,**  
**CF\_TxnStatus\_FILESTORE\_REJECTION = CF\_CFDP\_ConditionCode\_FILESTORE\_REJECTION ,**  
**CF\_TxnStatus\_FILE\_CHECKSUM\_FAILURE = CF\_CFDP\_ConditionCode\_FILE\_CHECKSUM\_FAILURE ,**  
**CF\_TxnStatus\_FILE\_SIZE\_ERROR = CF\_CFDP\_ConditionCode\_FILE\_SIZE\_ERROR ,**  
**CF\_TxnStatus\_NAK\_LIMIT\_REACHED = CF\_CFDP\_ConditionCode\_NAK\_LIMIT\_REACHED , CF\_TxnStatus\_INACTIVITY\_DETECTED = CF\_CFDP\_ConditionCode\_INACTIVITY\_DETECTED ,**  
**CF\_TxnStatus\_INVALID\_FILE\_STRUCTURE = CF\_CFDP\_ConditionCode\_INVALID\_FILE\_STRUCTURE ,**  
**CF\_TxnStatus\_CHECK\_LIMIT\_REACHED = CF\_CFDP\_ConditionCode\_CHECK\_LIMIT\_REACHED ,**  
**CF\_TxnStatus\_UNSUPPORTED\_CHECKSUM\_TYPE = CF\_CFDP\_ConditionCode\_UNSUPPORTED\_CHECKSUM\_TYPE ,**  
**CF\_TxnStatus\_SUSPEND\_REQUEST\_RECEIVED = CF\_CFDP\_ConditionCode\_SUSPEND\_REQUEST\_RECEIVED ,**  
**CF\_TxnStatus\_CANCEL\_REQUEST\_RECEIVED = CF\_CFDP\_ConditionCode\_CANCEL\_REQUEST\_RECEIVED ,**  
**CF\_TxnStatus\_PROTOCOL\_ERROR = 16 ,**

```
CF_TxnStatus_ACK_LIMIT_NO_FIN = 17 , CF_TxnStatus_ACK_LIMIT_NO_EOF = 18 , CF_TxnStatus_NAK_RESPONSE_ERROR  
= 19 , CF_TxnStatus_SEND_EOF_FAILURE = 20 ,  
CF_TxnStatus_EARLY_FIN = 21 , CF_TxnStatus_READ_FAILURE = 22 , CF_TxnStatus_NO_RESOURCE = 23  
, CF_TxnStatus_MAX = 24 }
```

*Values for Transaction Status code.*

- enum `CF_TickState_t`{  
`CF_TickState_INIT` , `CF_TickState_RX_STATE` , `CF_TickState_TX_STATE` , `CF_TickState_TX_NAK` ,  
`CF_TickState_TX_FILEDATA` , `CF_TickState_TX_PEND` , `CF_TickState_COMPLETE` , `CF_TickState_NUM_TYPES`  
}

*Identifies the type of timer tick being processed.*

#### 12.45.1 Detailed Description

Macros and data types used across the CF application

##### Note

Functions should not be declared in this file. This should be limited to shared macros and data types only. For unit testing, functions should be declared only in a header file with the same name as the C file that defines that function.

#### 12.45.2 Macro Definition Documentation

##### 12.45.2.1 CF\_NUM\_CHUNKS\_ALL\_CHANNELS `#define CF_NUM_CHUNKS_ALL_CHANNELS (CF_TOTAL_CHUNKS * CF_NUM_TRANSACTIONS_PER_CHANNEL)`

Maximum possible number of chunk entries that may exist in the CF application.

Definition at line 66 of file cf\_cfdp\_types.h.

##### 12.45.2.2 CF\_NUM\_HISTORIES `#define CF_NUM_HISTORIES (CF_NUM_CHANNELS * CF_NUM_HISTORIES_PER_CHANNEL)`

Maximum possible number of history entries that may exist in the CF application.

Definition at line 61 of file cf\_cfdp\_types.h.

##### 12.45.2.3 CF\_NUM\_TRANSACTIONS `#define CF_NUM_TRANSACTIONS (CF_NUM_CHANNELS * CF_NUM_TRANSACTIONS_PER_CHANNEL)`

Maximum possible number of transactions that may exist in the CF application.

Definition at line 56 of file cf\_cfdp\_types.h.

##### 12.45.2.4 CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL `#define CF_NUM_TRANSACTIONS_PER_CHANNEL`

###### Value:

```
(CF_MAX_COMMANDED_PLAYBACK_FILES_PER_CHAN + CF_MAX_SIMULTANEOUS_RX +  
((CF_MAX_POLLING_DIR_PER_CHAN + CF_MAX_COMMANDED_PLAYBACK_DIRECTORIES_PER_CHAN) * \  
CF_NUM_TRANSACTIONS_PER_PLAYBACK))
```

Maximum possible number of transactions that may exist on a single CF channel.

Definition at line 48 of file cf\_cfdp\_types.h.

#### 12.45.3 Typedef Documentation

**12.45.3.1 CF\_Channel\_t** `typedef struct CF_Channel CF_Channel_t`  
Channel state object.

This keeps the state of CF channels

Each CF channel has a separate transaction list, PDU throttle, playback, and poll state, as well as separate addresses on the underlying message transport (e.g. SB).

**12.45.3.2 CF\_ChunkWrapper\_t** `typedef struct CF_ChunkWrapper CF_ChunkWrapper_t`  
Wrapper around a CF\_ChunkList\_t object.

This allows a CF\_ChunkList\_t to be stored within a CList data storage structure

**12.45.3.3 CF\_Engine\_t** `typedef struct CF_Engine CF_Engine_t`

An engine represents a pairing to a local EID.

Each engine can have at most CF\_MAX\_SIMULTANEOUS\_TRANSACTIONS

**12.45.3.4 CF\_Flags\_Common\_t** `typedef struct CF_Flags_Common CF_Flags_Common_t`  
Data that applies to all types of transactions.

**12.45.3.5 CF\_Flags\_Rx\_t** `typedef struct CF_Flags_Rx CF_Flags_Rx_t`  
Flags that apply to receive transactions.

**12.45.3.6 CF\_Flags\_Tx\_t** `typedef struct CF_Flags_Tx CF_Flags_Tx_t`  
Flags that apply to send transactions.

**12.45.3.7 CF\_History\_t** `typedef struct CF_History CF_History_t`  
CF History entry.

Records CF app operations for future reference

**12.45.3.8 CF\_Input\_t** `typedef struct CF_Input CF_Input_t`

CF engine input state.

Keeps the state of the current input PDU in the CF engine

**12.45.3.9 CF\_Output\_t** `typedef struct CF_Output CF_Output_t`

CF engine output state.

Keeps the state of the current output PDU in the CF engine

**12.45.3.10 CF\_Playback\_t** `typedef struct CF_Playback CF_Playback_t`

CF Playback entry.

Keeps the state of CF playback requests

**12.45.3.11 CF\_Poll\_t** `typedef struct CF_Poll CF_Poll_t`

CF Poll entry.

Keeps the state of CF directory polling

**12.45.3.12 CF\_StateData\_t** `typedef struct CF_StateData CF_StateData_t`

Summary of all possible transaction state information (tx and rx)

**12.45.3.13 CF\_StateFlags\_t** `typedef union CF_StateFlags CF_StateFlags_t`  
Summary of all possible transaction flags (tx and rx)

**12.45.3.14 CF\_Transaction\_t** `typedef struct CF_Transaction CF_Transaction_t`  
Transaction state object.

This keeps the state of CF file transactions

#### 12.45.4 Enumeration Type Documentation

**12.45.4.1 CF\_Direction\_t** `enum CF_Direction_t`

Direction identifier.

Differentiates between send and receive history entries

Enumerator

CF_Direction_RX	
CF_Direction_TX	
CF_Direction_NUM	

Definition at line 115 of file cf\_cfdp\_types.h.

**12.45.4.2 CF\_RxSubState\_t** `enum CF_RxSubState_t`

Sub-state of a receive file transaction.

Enumerator

CF_RxSubState_DATA_NORMAL	waiting for more PDUs, no EOF received yet (normal recv)
CF_RxSubState_DATA_EOF	Got an EOF, filling in remaining gaps (NAKs may be sent)
CF_RxSubState_VALIDATE	Checking the CRC on the complete file
CF_RxSubState_FILESTORE	Performing file store ops
CF_RxSubState_FINACK	pending final fin/fin-ack exchange
CF_RxSubState_COMPLETE	Transaction is done
CF_RxSubState_NUM_STATES	

Definition at line 99 of file cf\_cfdp\_types.h.

**12.45.4.3 CF\_TickState\_t** `enum CF_TickState_t`

Identifies the type of timer tick being processed.

Enumerator

CF_TickState_INIT	
CF_TickState_RX_STATE	
CF_TickState_TX_STATE	
CF_TickState_TX_NAK	
CF_TickState_TX_FILEDATA	

## Enumerator

CF_TickState_TX_PEND	
CF_TickState_COMPLETE	
CF_TickState_NUM_TYPES	

Definition at line 360 of file cf\_cfdp\_types.h.

**12.45.4.4 CF\_TxnState\_t enum CF\_TxnState\_t**

High-level state of a transaction.

## Enumerator

CF_TxnState_UNDEF	State assigned to an unused object on the free list.
CF_TxnState_INIT	State assigned to a newly allocated transaction object.
CF_TxnState_R1	Receive file as class 1.
CF_TxnState_S1	Send file as class 1.
CF_TxnState_R2	Receive file as class 2.
CF_TxnState_S2	Send file as class 2.
CF_TxnState_DROP	State where all PDUs are dropped.
CF_TxnState_HOLD	State assigned to a transaction after freeing it.
CF_TxnState_INVALID	Marker value for the highest possible state number.

Definition at line 71 of file cf\_cfdp\_types.h.

**12.45.4.5 CF\_TxnStatus\_t enum CF\_TxnStatus\_t**

Values for Transaction Status code.

This enum defines the possible values representing the result of a transaction. This is a superset of the condition codes defined in CCSDS book 727 (condition codes) but with additional values for local conditions that the blue book does not have, such as protocol/state machine or decoding errors.

The values here are designed to not overlap with the condition codes defined in the blue book, but can be translated to one of those codes for the purposes of FIN/ACK/EOF PDUs.

## Enumerator

CF_TxnStatus_UNDEFINED	The undefined status is a placeholder for new transactions before a value is set.
CF_TxnStatus_NO_ERROR	
CF_TxnStatus_POS_ACK_LIMIT_REACHED	
CF_TxnStatus_KEEP_ALIVE_LIMIT_REACHED	
CF_TxnStatus_INVALID_TRANSMISSION_MODE	
CF_TxnStatus_FILESTORE_REJECTION	
CF_TxnStatus_FILE_CHECKSUM_FAILURE	
CF_TxnStatus_FILE_SIZE_ERROR	
CF_TxnStatus_NAK_LIMIT_REACHED	
CF_TxnStatus_INACTIVITY_DETECTED	
CF_TxnStatus_INVALID_FILE_STRUCTURE	
CF_TxnStatus_CHECK_LIMIT_REACHED	

**Enumerator**

CF_TxnStatus_UNSUPPORTED_CHECKSUM_TYPE	
CF_TxnStatus_SUSPEND_REQUEST_RECEIVED	
CF_TxnStatus_CANCEL_REQUEST_RECEIVED	
CF_TxnStatus_PROTOCOL_ERROR	
CF_TxnStatus_ACK_LIMIT_NO_FIN	
CF_TxnStatus_ACK_LIMIT_NO_EOF	
CF_TxnStatus_NAK_RESPONSE_ERROR	
CF_TxnStatus_SEND_EOF_FAILURE	
CF_TxnStatus_EARLY_FIN	
CF_TxnStatus_READ_FAILURE	
CF_TxnStatus_NO_RESOURCE	
CF_TxnStatus_MAX	

Definition at line 135 of file cf\_cfdp\_types.h.

**12.45.4.6 CF\_TxSubState\_t enum CF\_TxSubState\_t**

Sub-state of a send file transaction.

**Enumerator**

CF_TxSubState_DATA_NORMAL	sending the initial MD directive and file data
CF_TxSubState_DATA_EOF	Sent an EOF, waiting on EOF-ACK and FIN (or NAK)
CF_TxSubState_FILESTORE	Performing file store ops
CF_TxSubState_COMPLETE	Transaction is done
CF_TxSubState_NUM_STATES	

Definition at line 87 of file cf\_cfdp\_types.h.

**12.46 apps/cf/fsw/src(cf\_chunk.c File Reference**

```
#include <string.h>
#include "cf_verify.h"
#include "cf_assert.h"
#include "cf_chunk.h"
```

**Functions**

- void **CF\_Chunks\_EraseRange** (**CF\_ChunkList\_t** \*chunks, **CF\_ChunkIdx\_t** start, **CF\_ChunkIdx\_t** end)  
*Erase a range of chunks.*
- void **CF\_Chunks\_EraseChunk** (**CF\_ChunkList\_t** \*chunks, **CF\_ChunkIdx\_t** erase\_index)  
*Erase a single chunk.*
- void **CF\_Chunks\_InsertChunk** (**CF\_ChunkList\_t** \*chunks, **CF\_ChunkIdx\_t** index\_before, const **CF\_Chunk\_t** \*chunk)  
*Insert a chunk before index\_before.*
- **CF\_ChunkIdx\_t CF\_Chunks\_FindInsertPosition** (**CF\_ChunkList\_t** \*chunks, const **CF\_Chunk\_t** \*chunk)  
*Finds where a chunk should be inserted in the chunks.*

- int `CF_Chunks_CombinePrevious (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`  
*Possibly combines the given chunk with the previous chunk.*
- `CFE_Status_t CF_Chunks_CombineNext (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`  
*Possibly combines the given chunk with the next chunk.*
- `CF_ChunkIdx_t CF_Chunks_FindSmallestSize (const CF_ChunkList_t *chunks)`  
*Finds the smallest size out of all chunks.*
- void `CF_Chunks_Insert (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`  
*Insert a chunk.*
- void `CF_ChunkListAdd (CF_ChunkList_t *chunks, CF_ChunkOffset_t offset, CF_ChunkSize_t size)`  
*Public function to add a chunk.*
- void `CF_ChunkList_RemoveFromFirst (CF_ChunkList_t *chunks, CF_ChunkSize_t size)`  
*Public function to remove some amount of size from the first chunk.*
- const `CF_Chunk_t * CF_ChunkList_GetFirstChunk (const CF_ChunkList_t *chunks)`  
*Public function to get the entire first chunk from the list.*
- void `CF_ChunkListInit (CF_ChunkList_t *chunks, CF_ChunkIdx_t max_chunks, CF_Chunk_t *chunks_mem)`  
*Initialize a CF\_ChunkList\_t structure.*
- void `CF_ChunkListReset (CF_ChunkList_t *chunks)`  
*Resets a chunks structure.*
- uint32 `CF_ChunkList_ComputeGaps (const CF_ChunkList_t *chunks, CF_ChunkIdx_t max_gaps, CF_ChunkSize_t total, CF_ChunkOffset_t start, CF_ChunkList_ComputeGapFn_t compute_gap_fn, void *opaque)`  
*Compute gaps between chunks, and call a callback for each.*

### 12.46.1 Detailed Description

The CF Application chunks (sparse gap tracking) logic file

This class handles the complexity of sparse gap tracking so that the CFDP engine doesn't need to worry about it. Information is given to the class and when needed calculations are made internally to help the engine build NAK packets. Received NAK segment requests are stored in this class as well and used for re-transmit processing. This is intended to be mostly a generic purpose class used by CF.

### 12.46.2 Function Documentation

#### 12.46.2.1 `CF_ChunkList_ComputeGaps()` `uint32 CF_ChunkList_ComputeGaps (`

```

const CF_ChunkList_t * chunks,
CF_ChunkIdx_t max_gaps,
CF_ChunkSize_t total,
CF_ChunkOffset_t start,
CF_ChunkList_ComputeGapFn_t compute_gap_fn,
void * opaque )
```

Compute gaps between chunks, and call a callback for each.

##### Description

This function walks over all chunks and computes the gaps between. It can exit early if the calculated gap start is larger than the desired total.

##### Assumptions, External Events, and Notes:

chunks must not be NULL. compute\_gap\_fn is a valid function address.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>max_gaps</i>	Maximum number of gaps to compute
<i>total</i>	Size of the entire file
<i>start</i>	Beginning offset for gap computation
<i>compute_gap_fn</i>	Callback function to be invoked for each gap
<i>opaque</i>	Opaque pointer to pass through to callback function

**Returns**

The number of computed gaps.

Definition at line 362 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, CF\_ChunkList::count, CF\_Chunk::offset, and CF\_Chunk::size.  
Referenced by CF\_CFDP\_R\_CheckComplete(), and CF\_CFDP\_R\_SendNak().

**12.46.2.2 CF\_ChunkList\_GetFirstChunk()** const CF\_Chunk\_t\* CF\_ChunkList\_GetFirstChunk ( const CF\_ChunkList\_t \* *chunks* )

Public function to get the entire first chunk from the list.

This returns the first chunk from the list, or NULL if the list is empty.

**Note**

The chunk remains on the list - this call does not consume or remove the chunk from the list.

**Assumptions, External Events, and Notes:**

*chunks* must not be NULL.

**Returns**

Pointer to first chunk from the CF\_ChunkList\_t object

**Return values**

<i>NULL</i>	if the list was empty
-------------	-----------------------

Definition at line 325 of file cf\_chunk.c.

References CF\_ChunkList::chunks, and CF\_ChunkList::count.  
Referenced by CF\_CFDP\_S\_Tick\_Nak().

**12.46.2.3 CF\_ChunkList\_RemoveFromFirst()** void CF\_ChunkList\_RemoveFromFirst ( CF\_ChunkList\_t \* *chunks*, CF\_ChunkSize\_t *size* )

Public function to remove some amount of size from the first chunk.

### Description

This may remove the chunk entirely. This function is to satisfy the use-case where data is retrieved from the structure in-order and once consumed should be removed.

### Assumptions, External Events, and Notes:

chunks must not be NULL and list must not be empty

### Note

Good computer science would have a generic remove function, but that's much more complex than we need for the use case. We aren't trying to make chunks a general purpose reusable module, so just take the simple case that we need.

### Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>size</i>	Size to remove

Definition at line 299 of file cf\_chunk.c.

References CF\_Chunks\_EraseChunk(), CF\_ChunkList::chunks, CF\_Chunk::offset, and CF\_Chunk::size.

Referenced by CF\_CFDP\_S\_Tick\_Nak().

Here is the call graph for this function:



**12.46.2.4 CF\_ChunkListAdd()** void CF\_ChunkListAdd (

- CF\_ChunkList\_t \* *chunks*,
- CF\_ChunkOffset\_t *offset*,
- CF\_ChunkSize\_t *size* )

Public function to add a chunk.

### Assumptions, External Events, and Notes:

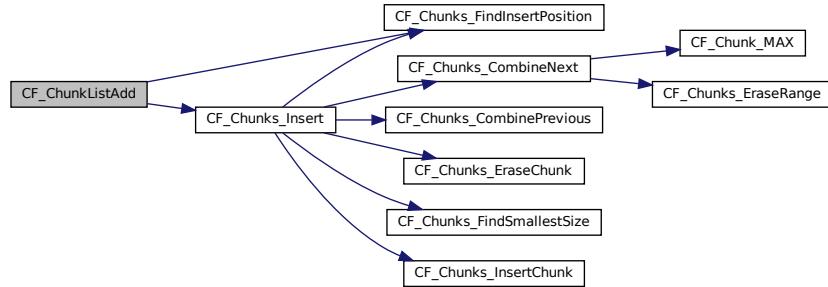
chunks must not be NULL.

### Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>offset</i>	Offset of chunk to add
<i>size</i>	Size of chunk to add

Definition at line 280 of file cf\_chunk.c.

References CF\_Assert, CF\_Chunks\_FindInsertPosition(), and CF\_Chunks\_Insert().  
Referenced by CF\_CFDP\_R\_ProcessFd(), and CF\_CFDP\_S2\_SubstateNak().  
Here is the call graph for this function:



#### 12.46.2.5 CF\_ChunkListInit()

```
void CF_ChunkListInit (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t max_chunks,
    CF_Chunk_t * chunks_mem )
```

Initialize a CF\_ChunkList\_t structure.

**Assumptions, External Events, and Notes:**

chunks must not be NULL. chunks\_mem must not be NULL.

#### Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object to initialize
<i>max_chunks</i>	Maximum number of entries in the chunks_mem array
<i>chunks_mem</i>	Array of CF_Chunk_t objects with length of max_chunks

Definition at line 336 of file cf\_chunk.c.

References CF\_Assert, CF\_ChunkListReset(), CF\_ChunkList::chunks, and CF\_ChunkList::max\_chunks.

Referenced by CF\_CFDP\_InitEngine().

Here is the call graph for this function:



**12.46.2.6 CF\_ChunkListReset()** `void CF_ChunkListReset (`  
 `CF_ChunkList_t * chunks )`

Resets a chunks structure.

All chunks are removed from the list, but the max\_chunks and chunk memory pointers are retained. This returns the chunk list to the same state as it was after the initial call to [CF\\_ChunkListInit\(\)](#).

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<code>chunks</code>	Pointer to CF_ChunkList_t object
---------------------	----------------------------------

Definition at line 350 of file cf\_chunk.c.

References CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_ChunkList::max\_chunks.

Referenced by CF\_ChunkListInit().

**12.46.2.7 CF\_Chunks\_CombineNext()** `CFE_Status_t CF_Chunks_CombineNext (`  
 `CF_ChunkList_t * chunks,`  
 `CF_ChunkIdx_t i,`  
 `const CF_Chunk_t * chunk )`

Possibly combines the given chunk with the next chunk.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

**Parameters**

<code>chunks</code>	Pointer to CF_ChunkList_t object
<code>i</code>	Index of chunk to combine
<code>chunk</code>	Chunk data to combine

**Returns**

boolean code indicating if chunks were combined

**Return values**

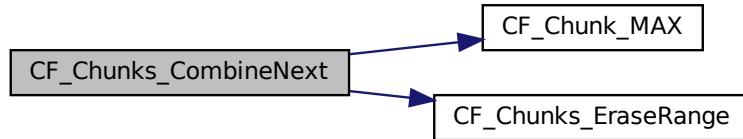
<code>1</code>	if combined with another chunk
<code>0</code>	if not combined

Definition at line 170 of file cf\_chunk.c.

References CF\_ASSERT, CF\_Chunk\_MAX(), CF\_Chunks\_EraseRange(), CF\_ChunkList::chunks, CF\_ChunkList::count, CF\_Chunk::offset, and CF\_Chunk::size.

Referenced by CF\_Chunks\_Insert().

Here is the call graph for this function:



#### 12.46.2.8 CF\_Chunks\_CombinePrevious()

```
int CF_Chunks_CombinePrevious (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t i,
    const CF_Chunk_t * chunk )
```

Possibly combines the given chunk with the previous chunk.

Assumptions, External Events, and Notes:

chunks must not be NULL, chunk must not be NULL.

#### Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Index of chunk to combine
<i>chunk</i>	Chunk data to combine

#### Returns

boolean code indicating if chunks were combined

#### Return values

1	if combined with another chunk
0	if not combined

Definition at line 133 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, CF\_Chunk::offset, and CF\_Chunk::size.

Referenced by CF\_Chunks\_Insert().

#### 12.46.2.9 CF\_Chunks\_EraseChunk()

```
void CF_Chunks_EraseChunk (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t erase_index )
```

Erase a single chunk.

**Note**

This changes the chunk IDs of all chunks that follow items in the list after the `erase_index` will be shifted/moved to close the gap.

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>erase_index</i>	chunk ID to erase

Definition at line 63 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, and CF\_ChunkList::count.

Referenced by CF\_ChunkList\_RemoveFromFirst(), and CF\_Chunks\_Insert().

```
12.46.2.10 CF_Chunks_EraseRange() void CF_Chunks_EraseRange (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t start,
    CF_ChunkIdx_t end )
```

Erase a range of chunks.

Items in the list starting at the end index will be shifted/moved to close the gap. If end <= start nothing will be done (OK to pass matching start/end as a no-op) If end == list size, list from start on will be erased (nothing moved)

Example: list = {a, b, c, d, e} EraseRange index start 1, end 3 list = {a, d, e}

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>start</i>	Starting chunk ID to erase (inclusive)
<i>end</i>	Ending chunk ID (exclusive, this chunk will not be erased)

Definition at line 45 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, and CF\_ChunkList::count.

Referenced by CF\_Chunks\_CombineNext().

```
12.46.2.11 CF_Chunks_FindInsertPosition() CF_ChunkIdx_t CF_Chunks_FindInsertPosition (
    CF_ChunkList_t * chunks,
    const CF_Chunk_t * chunk )
```

Finds where a chunk should be inserted in the chunks.

**Description**

This is a C version of std::lower\_bound from C++ algorithms.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>chunk</i>	Chunk data to insert

**Returns**

an index to the first chunk that is greater than or equal to the requested's offset.

Definition at line 101 of file cf\_chunk.c.

References CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_Chunk::offset.

Referenced by CF\_ChunkListAdd(), and CF\_Chunks\_Insert().

**12.46.2.12 CF\_Chunks\_FindSmallestSize()** `CF_ChunkIdx_t CF_Chunks_FindSmallestSize (`  
`const CF_ChunkList_t * chunks )`

Finds the smallest size out of all chunks.

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
---------------	----------------------------------

**Returns**

The chunk index with the smallest size.

**Return values**

<i>0</i>	if the chunk list is empty
----------	----------------------------

Definition at line 214 of file cf\_chunk.c.

References CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_Chunk::size.

Referenced by CF\_Chunks\_Insert().

**12.46.2.13 CF\_Chunks\_Insert()** `void CF_Chunks_Insert (`  
`CF_ChunkList_t * chunks,`  
`CF_ChunkIdx_t i,`  
`const CF_Chunk_t * chunk )`

Insert a chunk.

**Description**

Inserts the chunk at the specified location. May combine with an existing chunk if contiguous.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

**Parameters**

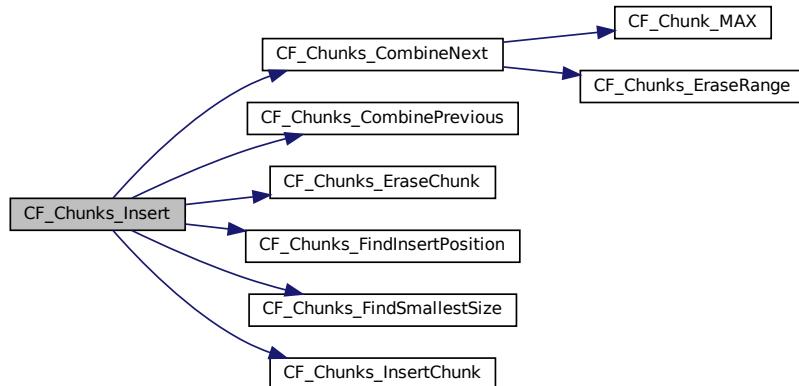
<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Position to insert chunk at
<i>chunk</i>	Chunk data to insert

Definition at line 236 of file cf\_chunk.c.

References CF\_Chunks\_CombineNext(), CF\_Chunks\_CombinePrevious(), CF\_Chunks\_EraseChunk(), CF\_Chunks→\_FindInsertPosition(), CF\_Chunks\_FindSmallestSize(), CF\_Chunks\_InsertChunk(), CF\_ChunkList::chunks, CF→\_ChunkList::count, CF\_ChunkList::max\_chunks, and CF\_Chunk::size.

Referenced by CF\_ChunkListAdd().

Here is the call graph for this function:

**12.46.2.14 CF\_Chunks\_InsertChunk()**

```
void CF_Chunks_InsertChunk (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t index_before,
    const CF_Chunk_t * chunk )
```

Insert a chunk before index\_before.

**Note**

This changes the chunk IDs of all chunks that follow items in the list after the index\_before will be shifted/moved to open a gap.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>index_before</i>	position to insert at - this becomes the ID of the inserted chunk
<i>chunk</i>	Chunk data to insert (copied)

Definition at line 80 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_ChunkList::max\_chunks.  
Referenced by CF\_Chunks\_Insert().

## 12.47 apps/cf/fsw/src(cf\_chunk.h File Reference

```
#include "cfe.h"
```

### Data Structures

- struct [CF\\_Chunk](#)  
*Pairs an offset with a size to identify a specific piece of a file.*
- struct [CF\\_ChunkList](#)  
*A list of CF\_Chunk\_t pairs.*

### TypeDefs

- typedef uint32 [CF\\_ChunkIdx\\_t](#)
- typedef uint32 [CF\\_ChunkOffset\\_t](#)
- typedef uint32 [CF\\_ChunkSize\\_t](#)
- typedef struct [CF\\_Chunk](#) [CF\\_Chunk\\_t](#)  
*Pairs an offset with a size to identify a specific piece of a file.*
- typedef struct [CF\\_ChunkList](#) [CF\\_ChunkList\\_t](#)  
*A list of CF\_Chunk\_t pairs.*
- typedef void(\* [CF\\_ChunkList\\_ComputeGapFn\\_t](#)) (const [CF\\_ChunkList\\_t](#) \*cs, const [CF\\_Chunk\\_t](#) \*chunk, void \*opaque)  
*Function for use with CF\_ChunkList\_ComputeGaps()*

### Functions

- static [CF\\_ChunkOffset\\_t](#) [CF\\_Chunk\\_MAX](#) ([CF\\_ChunkOffset\\_t](#) a, [CF\\_ChunkOffset\\_t](#) b)  
*Selects the larger of the two passed-in offsets.*
- void [CF\\_ChunkListInit](#) ([CF\\_ChunkList\\_t](#) \*chunks, [CF\\_ChunkIdx\\_t](#) max\_chunks, [CF\\_Chunk\\_t](#) \*chunks\_mem)  
*Initialize a CF\_ChunkList\_t structure.*
- void [CF\\_ChunkListAdd](#) ([CF\\_ChunkList\\_t](#) \*chunks, [CF\\_ChunkOffset\\_t](#) offset, [CF\\_ChunkSize\\_t](#) size)  
*Public function to add a chunk.*
- void [CF\\_ChunkListReset](#) ([CF\\_ChunkList\\_t](#) \*chunks)  
*Resets a chunks structure.*
- void [CF\\_ChunkList\\_RemoveFromFirst](#) ([CF\\_ChunkList\\_t](#) \*chunks, [CF\\_ChunkSize\\_t](#) size)  
*Public function to remove some amount of size from the first chunk.*
- const [CF\\_Chunk\\_t](#) \* [CF\\_ChunkList\\_GetFirstChunk](#) (const [CF\\_ChunkList\\_t](#) \*chunks)  
*Public function to get the entire first chunk from the list.*
- uint32 [CF\\_ChunkList\\_ComputeGaps](#) (const [CF\\_ChunkList\\_t](#) \*chunks, [CF\\_ChunkIdx\\_t](#) max\_gaps, [CF\\_ChunkSize\\_t](#) total, [CF\\_ChunkOffset\\_t](#) start, [CF\\_ChunkList\\_ComputeGapFn\\_t](#) compute\_gap\_fn, void \*opaque)

*Compute gaps between chunks, and call a callback for each.*

- void `CF_Chunks_EraseRange (CF_ChunkList_t *chunks, CF_ChunkIdx_t start, CF_ChunkIdx_t end)`  
*Erase a range of chunks.*
- void `CF_Chunks_EraseChunk (CF_ChunkList_t *chunks, CF_ChunkIdx_t erase_index)`  
*Erase a single chunk.*
- void `CF_Chunks_InsertChunk (CF_ChunkList_t *chunks, CF_ChunkIdx_t index_before, const CF_Chunk_t *chunk)`  
*Insert a chunk before index\_before.*
- `CF_ChunkIdx_t CF_Chunks_FindInsertPosition (CF_ChunkList_t *chunks, const CF_Chunk_t *chunk)`  
*Finds where a chunk should be inserted in the chunks.*
- int `CF_Chunks_CombinePrevious (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`  
*Possibly combines the given chunk with the previous chunk.*
- `CFE_Status_t CF_Chunks_CombineNext (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`  
*Possibly combines the given chunk with the next chunk.*
- `CF_ChunkIdx_t CF_Chunks_FindSmallestSize (const CF_ChunkList_t *chunks)`  
*Finds the smallest size out of all chunks.*
- void `CF_Chunks_Insert (CF_ChunkList_t *chunks, CF_ChunkIdx_t i, const CF_Chunk_t *chunk)`  
*Insert a chunk.*

### 12.47.1 Detailed Description

The CF Application chunks (spare gap tracking) header file

### 12.47.2 Typedef Documentation

#### 12.47.2.1 `CF_Chunk_t` `typedef struct CF_Chunk CF_Chunk_t`

Pairs an offset with a size to identify a specific piece of a file.

#### 12.47.2.2 `CF_ChunkIdx_t` `typedef uint32 CF_ChunkIdx_t`

Definition at line 31 of file cf\_chunk.h.

#### 12.47.2.3 `CF_ChunkList_ComputeGapFn_t` `typedef void(* CF_ChunkList_ComputeGapFn_t) (const CF_ChunkList_t`

`*cs, const CF_Chunk_t *chunk, void *opaque)`

Function for use with `CF_ChunkList_ComputeGaps()`

#### Parameters

<code>cs</code>	Pointer to the <code>CF_ChunkList_t</code> object
<code>chunk</code>	Pointer to the chunk being currently processed
<code>opaque</code>	Opaque pointer passed through from initial call

Definition at line 63 of file cf\_chunk.h.

#### 12.47.2.4 `CF_ChunkList_t` `typedef struct CF_ChunkList CF_ChunkList_t`

A list of CF\_Chunk\_t pairs.

This list is ordered by chunk offset, from lowest to highest

#### 12.47.2.5 **CF\_ChunkOffset\_t** `typedef uint32 CF_ChunkOffset_t`

Definition at line 32 of file cf\_chunk.h.

#### 12.47.2.6 **CF\_ChunkSize\_t** `typedef uint32 CF_ChunkSize_t`

Definition at line 33 of file cf\_chunk.h.

### 12.47.3 Function Documentation

**12.47.3.1 CF\_Chunk\_MAX()** `static CF_ChunkOffset_t CF_Chunk_MAX (`  
`CF_ChunkOffset_t a,`  
`CF_ChunkOffset_t b ) [inline], [static]`

Selects the larger of the two passed-in offsets.

#### Parameters

<i>a</i>	First chunk offset
<i>b</i>	Second chunk offset

#### Returns

the larger CF\_ChunkOffset\_t value

Definition at line 72 of file cf\_chunk.h.

Referenced by CF\_Chunks\_CombineNext().

**12.47.3.2 CF\_ChunkList\_ComputeGaps()** `uint32 CF_ChunkList_ComputeGaps (`  
`const CF_ChunkList_t * chunks,`  
`CF_ChunkIdx_t max_gaps,`  
`CF_ChunkSize_t total,`  
`CF_ChunkOffset_t start,`  
`CF_ChunkList_ComputeGapFn_t compute_gap_fn,`  
`void * opaque )`

Compute gaps between chunks, and call a callback for each.

#### Description

This function walks over all chunks and computes the gaps between. It can exit early if the calculated gap start is larger than the desired total.

#### Assumptions, External Events, and Notes:

chunks must not be NULL. compute\_gap\_fn is a valid function address.

#### Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
---------------	----------------------------------

**Parameters**

<i>max_gaps</i>	Maximum number of gaps to compute
<i>total</i>	Size of the entire file
<i>start</i>	Beginning offset for gap computation
<i>compute_gap_fn</i>	Callback function to be invoked for each gap
<i>opaque</i>	Opaque pointer to pass through to callback function

**Returns**

The number of computed gaps.

Definition at line 362 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, CF\_ChunkList::count, CF\_Chunk::offset, and CF\_Chunk::size.  
Referenced by CF\_CFDP\_R\_CheckComplete(), and CF\_CFDP\_R\_SendNak().

**12.47.3.3 CF\_ChunkList\_GetFirstChunk()** const CF\_Chunk\_t\* CF\_ChunkList\_GetFirstChunk ( const CF\_ChunkList\_t \* chunks )

Public function to get the entire first chunk from the list.

This returns the first chunk from the list, or NULL if the list is empty.

**Note**

The chunk remains on the list - this call does not consume or remove the chunk from the list.

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Returns**

Pointer to first chunk from the CF\_ChunkList\_t object

**Return values**

NULL	if the list was empty
------	-----------------------

Definition at line 325 of file cf\_chunk.c.

References CF\_ChunkList::chunks, and CF\_ChunkList::count.  
Referenced by CF\_CFDP\_S\_Tick\_Nak().

**12.47.3.4 CF\_ChunkList\_RemoveFromFirst()** void CF\_ChunkList\_RemoveFromFirst ( CF\_ChunkList\_t \* chunks, CF\_ChunkSize\_t size )

Public function to remove some amount of size from the first chunk.

**Description**

This may remove the chunk entirely. This function is to satisfy the use-case where data is retrieved from the structure in-order and once consumed should be removed.

**Assumptions, External Events, and Notes:**

chunks must not be NULL and list must not be empty

**Note**

Good computer science would have a generic remove function, but that's much more complex than we need for the use case. We aren't trying to make chunks a general purpose reusable module, so just take the simple case that we need.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>size</i>	Size to remove

Definition at line 299 of file cf\_chunk.c.

References CF\_Chunks\_EraseChunk(), CF\_ChunkList::chunks, CF\_Chunk::offset, and CF\_Chunk::size.

Referenced by CF\_CFDP\_S\_Tick\_Nak().

Here is the call graph for this function:



**12.47.3.5 CF\_ChunkListAdd()** void CF\_ChunkListAdd (   
     CF\_ChunkList\_t \* *chunks*,  
     CF\_ChunkOffset\_t *offset*,  
     CF\_ChunkSize\_t *size* )

Public function to add a chunk.

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

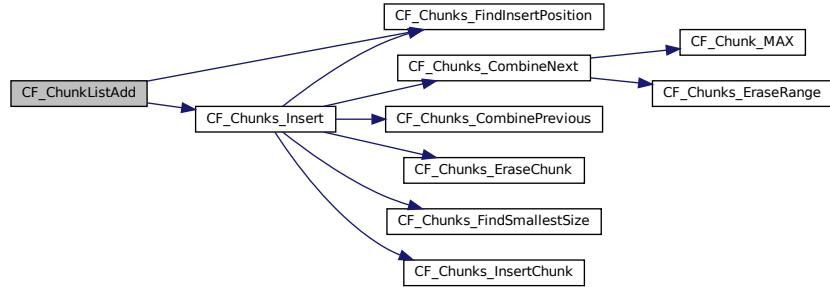
<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>offset</i>	Offset of chunk to add
<i>size</i>	Size of chunk to add

Definition at line 280 of file cf\_chunk.c.

References CF\_Assert, CF\_Chunks\_FindInsertPosition(), and CF\_Chunks\_Insert().

Referenced by CF\_CFDP\_R\_ProcessFd(), and CF\_CFDP\_S2\_SubstateNak().

Here is the call graph for this function:



#### **12.47.3.6 CF\_ChunkListInit()** void CF\_ChunkListInit (

```
CF_ChunkList_t * chunks,  
CF_ChunkIdx_t max_chunks,  
CF_Chunk_t * chunks_mem )
```

Initialize a CF\_ChunkList\_t structure.

### **Assumptions, External Events, and Notes:**

chunks must not be NULL. chunks\_mem must not be NULL.

## Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object to initialize
<i>max_chunks</i>	Maximum number of entries in the <i>chunks_mem</i> array
<i>chunks_mem</i>	Array of CF_Chunk_t objects with length of <i>max_chunks</i>

Definition at line 336 of file cf\_chunk.c.

References CF\_ASSERT, CF\_CHUNKLISTRESET(), CF\_CHUNKLIST::CHUNKS, and CF\_CHUNKLIST::MAX\_CHUNKS.

Referenced by CF\_CFDP\_InitEngine().

Here is the call graph for this function:



**12.47.3.7 CF\_ChunkListReset()** void CF\_ChunkListReset ( CF\_ChunkList\_t \* *chunks* )

Resets a chunks structure.

All chunks are removed from the list, but the max\_chunks and chunk memory pointers are retained. This returns the chunk list to the same state as it was after the initial call to [CF\\_ChunkListInit\(\)](#).

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
---------------	----------------------------------

Definition at line 350 of file cf\_chunk.c.

References CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_ChunkList::max\_chunks.

Referenced by CF\_ChunkListInit().

```
12.47.3.8 CF_Chunks_CombineNext() CFE_Status_t CF_Chunks_CombineNext (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t i,
    const CF_Chunk_t * chunk )
```

Possibly combines the given chunk with the next chunk.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Index of chunk to combine
<i>chunk</i>	Chunk data to combine

**Returns**

boolean code indicating if chunks were combined

**Return values**

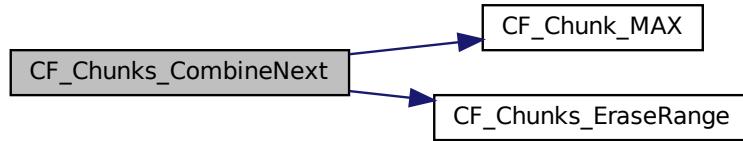
1	if combined with another chunk
0	if not combined

Definition at line 170 of file cf\_chunk.c.

References CF\_Assert, CF\_Chunk\_MAX(), CF\_Chunks\_EraseRange(), CF\_ChunkList::chunks, CF\_ChunkList::count, CF\_Chunk::offset, and CF\_Chunk::size.

Referenced by CF\_Chunks\_Insert().

Here is the call graph for this function:



#### 12.47.3.9 CF\_Chunks\_CombinePrevious()

```
int CF_Chunks_CombinePrevious (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t i,
    const CF_Chunk_t * chunk )
```

Possibly combines the given chunk with the previous chunk.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

#### Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Index of chunk to combine
<i>chunk</i>	Chunk data to combine

#### Returns

boolean code indicating if chunks were combined

#### Return values

1	if combined with another chunk
0	if not combined

Definition at line 133 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, CF\_Chunk::offset, and CF\_Chunk::size.

Referenced by CF\_Chunks\_Insert().

#### 12.47.3.10 CF\_Chunks\_EraseChunk()

```
void CF_Chunks_EraseChunk (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t erase_index )
```

Erase a single chunk.

**Note**

This changes the chunk IDs of all chunks that follow items in the list after the `erase_index` will be shifted/moved to close the gap.

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>erase_index</i>	chunk ID to erase

Definition at line 63 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, and CF\_ChunkList::count.

Referenced by CF\_ChunkList\_RemoveFromFirst(), and CF\_Chunks\_Insert().

```
12.47.3.11 CF_Chunks_EraseRange() void CF_Chunks_EraseRange (
    CF_ChunkList_t * chunks,
    CF_ChunkIdx_t start,
    CF_ChunkIdx_t end )
```

Erase a range of chunks.

Items in the list starting at the end index will be shifted/moved to close the gap. If `end <= start` nothing will be done (OK to pass matching start/end as a no-op) If `end == list size`, list from start on will be erased (nothing moved)

Example: list = {a, b, c, d, e} EraseRange index start 1, end 3 list = {a, d, e}

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>start</i>	Starting chunk ID to erase (inclusive)
<i>end</i>	Ending chunk ID (exclusive, this chunk will not be erased)

Definition at line 45 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, and CF\_ChunkList::count.

Referenced by CF\_Chunks\_CombineNext().

```
12.47.3.12 CF_Chunks_FindInsertPosition() CF_ChunkIdx_t CF_Chunks_FindInsertPosition (
    CF_ChunkList_t * chunks,
    const CF_Chunk_t * chunk )
```

Finds where a chunk should be inserted in the chunks.

**Description**

This is a C version of std::lower\_bound from C++ algorithms.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>chunk</i>	Chunk data to insert

**Returns**

an index to the first chunk that is greater than or equal to the requested's offset.

Definition at line 101 of file cf\_chunk.c.

References CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_Chunk::offset.

Referenced by CF\_ChunkListAdd(), and CF\_Chunks\_Insert().

**12.47.3.13 CF\_Chunks\_FindSmallestSize()** `CF_ChunkIdx_t CF_Chunks_FindSmallestSize (`

`const CF_ChunkList_t * chunks )`

Finds the smallest size out of all chunks.

**Assumptions, External Events, and Notes:**

chunks must not be NULL.

**Parameters**

<i>chunks</i>	Pointer to CF_ChunkList_t object
---------------	----------------------------------

**Returns**

The chunk index with the smallest size.

**Return values**

<i>0</i>	if the chunk list is empty
----------	----------------------------

Definition at line 214 of file cf\_chunk.c.

References CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_Chunk::size.

Referenced by CF\_Chunks\_Insert().

**12.47.3.14 CF\_Chunks\_Insert()** `void CF_Chunks_Insert (`

`CF_ChunkList_t * chunks,`  
`CF_ChunkIdx_t i,`  
`const CF_Chunk_t * chunk )`

Insert a chunk.

**Description**

Inserts the chunk at the specified location. May combine with an existing chunk if contiguous.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

**Parameters**

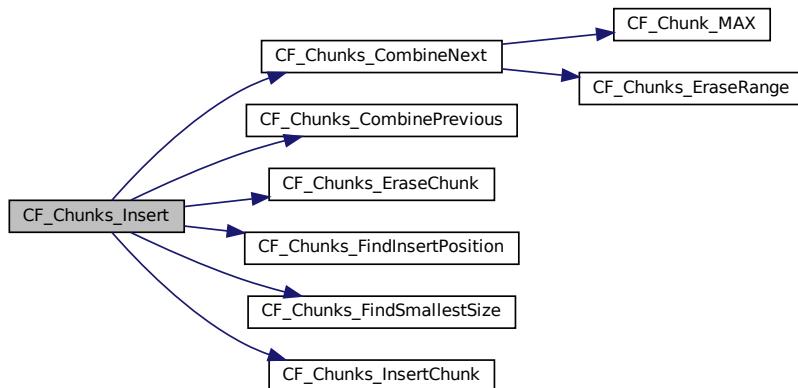
<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>i</i>	Position to insert chunk at
<i>chunk</i>	Chunk data to insert

Definition at line 236 of file cf\_chunk.c.

References CF\_Chunks\_CombineNext(), CF\_Chunks\_CombinePrevious(), CF\_Chunks\_EraseChunk(), CF\_Chunks→\_FindInsertPosition(), CF\_Chunks\_FindSmallestSize(), CF\_Chunks\_InsertChunk(), CF\_ChunkList::chunks, CF→\_ChunkList::count, CF\_ChunkList::max\_chunks, and CF\_Chunk::size.

Referenced by CF\_ChunkListAdd().

Here is the call graph for this function:



**12.47.3.15 CF\_Chunks\_InsertChunk()** `void CF_Chunks_InsertChunk (`  
 `CF_ChunkList_t * chunks,`  
 `CF_ChunkIdx_t index_before,`  
 `const CF_Chunk_t * chunk )`

Insert a chunk before index\_before.

**Note**

This changes the chunk IDs of all chunks that follow items in the list after the index\_before will be shifted/moved to open a gap.

**Assumptions, External Events, and Notes:**

chunks must not be NULL, chunk must not be NULL.

### Parameters

<i>chunks</i>	Pointer to CF_ChunkList_t object
<i>index_before</i>	position to insert at - this becomes the ID of the inserted chunk
<i>chunk</i>	Chunk data to insert (copied)

Definition at line 80 of file cf\_chunk.c.

References CF\_ASSERT, CF\_ChunkList::chunks, CF\_ChunkList::count, and CF\_ChunkList::max\_chunks.  
Referenced by CF\_Chunks\_Insert().

## 12.48 apps/cf/fsw/src(cf\_clist.c File Reference

```
#include "cf_clist.h"
#include "cf_assert.h"
```

### Functions

- void **CF\_CList\_InitNode** (CF\_CListNode\_t \*node)  
*Initialize a clist node.*
- void **CF\_CList\_InsertFront** (CF\_CListNode\_t \*\*head, CF\_CListNode\_t \*node)  
*Insert the given node into the front of a list.*
- void **CF\_CList\_InsertBack** (CF\_CListNode\_t \*\*head, CF\_CListNode\_t \*node)  
*Insert the given node into the back of a list.*
- CF\_CListNode\_t \* **CF\_CList\_Pop** (CF\_CListNode\_t \*\*head)  
*Remove the first node from a list and return it.*
- void **CF\_CList\_Remove** (CF\_CListNode\_t \*\*head, CF\_CListNode\_t \*node)  
*Remove the given node from the list.*
- void **CF\_CList\_InsertAfter** (CF\_CListNode\_t \*\*head, CF\_CListNode\_t \*start, CF\_CListNode\_t \*after)  
*Insert the given node into the last after the given start node.*
- void **CF\_CList\_Traverse** (CF\_CListNode\_t \*start, CF\_CListFn\_t fn, void \*context)  
*Traverse the entire list, calling the given function on all nodes.*
- void **CF\_CList\_Traverse\_R** (CF\_CListNode\_t \*end, CF\_CListFn\_t fn, void \*context)  
*Reverse list traversal, starting from end, calling given function on all nodes.*

### 12.48.1 Detailed Description

The CF Application circular list definition source file

This is a circular doubly-linked list implementation. It is used for all data structures in CF.

This file is intended to be a generic class that can be used in other apps.

### 12.48.2 Function Documentation

#### 12.48.2.1 CF\_CList\_InitNode() void CF\_CList\_InitNode (

```
    CF_CListNode_t * node )
```

Initialize a clist node.

Assumptions, External Events, and Notes:

node must not be NULL.

**Parameters**

<i>node</i>	Pointer to node structure to be initialized
-------------	---

Definition at line 40 of file cf\_clist.c.

References CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CFDP\_InitEngine(), CF\_CList\_Remove(), and CF\_FreeTransaction().

**12.48.2.2 CF\_CList\_InsertAfter()** void CF\_CList\_InsertAfter (

```
    CF_CListNode_t ** head,
    CF_CListNode_t * start,
    CF_CListNode_t * after )
```

Insert the given node into the last after the given start node.

**Assumptions, External Events, and Notes:**

head must not be NULL, node must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to remove from
<i>start</i>	Pointer to node to insert
<i>after</i>	Pointer to position to insert after

Definition at line 167 of file cf\_clist.c.

References CF\_Assert, CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CList\_InsertAfter\_Ex().

**12.48.2.3 CF\_CList\_InsertBack()** void CF\_CList\_InsertBack (

```
    CF_CListNode_t ** head,
    CF_CListNode_t * node )
```

Insert the given node into the back of a list.

**Assumptions, External Events, and Notes:**

head must not be NULL, node must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 81 of file cf\_clist.c.

References CF\_Assert, CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CFDP\_InitEngine(), CF\_CFDP\_RecycleTransaction(), CF\_CList\_InsertBack\_Ex(), and CF\_Move← Transaction().

**12.48.2.4 CF\_CList\_InsertFront()** void CF\_CList\_InsertFront (

```
    CF_CListNode_t ** head,
```

```
CF_CListNode_t * node )
```

Insert the given node into the front of a list.

**Assumptions, External Events, and Notes:**

head must not be NULL, node must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 52 of file cf\_clist.c.

References CF\_ASSERT, CF\_CListNode::next, and CF\_CListNode::prev.

#### 12.48.2.5 CF\_CList\_Pop() `CF_CListNode_t* CF_CList_Pop (`

```
CF_CListNode_t ** head )
```

Remove the first node from a list and return it.

**Assumptions, External Events, and Notes:**

head must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to remove from
-------------	--

**Returns**

The first node (now removed) in the list

**Return values**

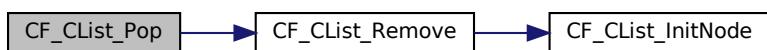
<code>NULL</code>	if list was empty.
-------------------	--------------------

Definition at line 111 of file cf\_clist.c.

References CF\_ASSERT, and CF\_CList\_Remove().

Referenced by CF\_CFDP\_FindUnusedChunks().

Here is the call graph for this function:



#### 12.48.2.6 CF\_CList\_Remove() `void CF_CList_Remove (`

```
CF_CListNode_t ** head,
CF_CListNode_t * node )
```

Remove the given node from the list.

#### Assumptions, External Events, and Notes:

head must not be NULL, node must not be NULL.

#### Parameters

<i>head</i>	Pointer to head of list to remove from
<i>node</i>	Pointer to node to remove

Definition at line 132 of file cf\_clist.c.

References CF\_ASSERT, CF\_CList\_InitNode(), CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CList\_Pop(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), and CF\_MoveTransaction().

Here is the call graph for this function:



#### 12.48.2.7 CF\_CList\_Traverse()

```
void CF_CList_Traverse (
    CF_CListNode_t * start,
    CF_CListFn_t fn,
    void * context )
```

Traverse the entire list, calling the given function on all nodes.

#### Assumptions, External Events, and Notes:

start may be NULL, fn must be a valid function, context may be NULL.

#### Note

on traversal it's ok to delete the current node, but do not delete other nodes in the same list!!

#### Parameters

<i>start</i>	List to traverse (first node)
<i>fn</i>	Callback function to invoke for each node
<i>context</i>	Opaque pointer to pass to callback

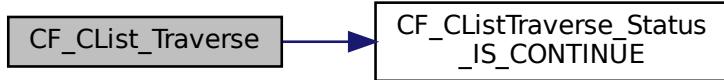
Definition at line 188 of file cf\_clist.c.

References CF\_CListTraverse\_Status\_IS\_CONTINUE(), and CF\_CListNode::next.

Referenced by CF\_CFDP\_DisableEngine(), CF\_CFDP\_TickTransactions(), CF\_DoPurgeQueue(), CF\_Find←

`TransactionBySequenceNumber()`, `CF_TraverseAllTransactions()`, `CF_WriteHistoryQueueDataToFile()`, and `CF_WriteTxnQueueDataToFile()`.

Here is the call graph for this function:



```

12.48.2.8 CF_CList_Traverse_R() void CF_CList_Traverse_R (
    CF_CListNode_t * end,
    CF_CListFn_t fn,
    void * context )
  
```

Reverse list traversal, starting from end, calling given function on all nodes.

**Assumptions, External Events, and Notes:**

end may be NULL. fn must be a valid function, context may be NULL.

#### Note

traverse\_R will work backwards from the parameter's prev, and end on param

#### Parameters

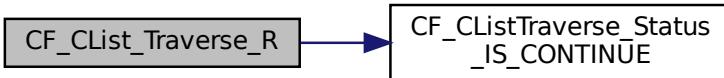
<i>end</i>	List to traverse (last node)
<i>fn</i>	Callback function to invoke for each node
<i>context</i>	Opaque pointer to pass to callback

Definition at line 228 of file cf\_clist.c.

References `CF_CListTraverse_Status_IS_CONTINUE()`, and `CF_CListNode::prev`.

Referenced by `CF_InsertSortPrio()`.

Here is the call graph for this function:



## 12.49 apps/cf/fsw/src/cf\_clist.h File Reference

```
#include <stddef.h>
#include <stdbool.h>
```

### Data Structures

- struct `CF_CListNode`

*Node link structure.*

### Macros

- `#define CF_CLIST_CONT CF_CListTraverse_Status_CONTINUE`  
*Constant indicating to continue traversal.*
- `#define CF_CLIST_EXIT CF_CListTraverse_Status_EXIT`  
*Constant indicating to stop traversal.*
- `#define container_of(ptr, type, member) ((type *)((char *)(ptr) - (char *)offsetof(type, member)))`  
*Obtains a pointer to the parent structure.*

### Typedefs

- `typedef struct CF_CListNode CF_CListNode_t`  
*Circular linked list node links.*
- `typedef CF_CListTraverse_Status_t(* CF_CListFn_t) (CF_CListNode_t *node, void *context)`  
*Callback function type for use with `CF_CList_Traverse()`*

### Enumerations

- enum `CF_CListTraverse_Status_t { CF_CListTraverse_Status_CONTINUE = 0 , CF_CListTraverse_Status_EXIT = 1 }`

### Functions

- `static bool CF_CListTraverse_Status_IS_CONTINUE (CF_CListTraverse_Status_t stat)`
- `void CF_CList_InitNode (CF_CListNode_t *node)`  
*Initialize a clist node.*
- `void CF_CList_InsertFront (CF_CListNode_t **head, CF_CListNode_t *node)`  
*Insert the given node into the front of a list.*
- `void CF_CList_InsertBack (CF_CListNode_t **head, CF_CListNode_t *node)`  
*Insert the given node into the back of a list.*
- `void CF_CList_Remove (CF_CListNode_t **head, CF_CListNode_t *node)`  
*Remove the given node from the list.*
- `CF_CListNode_t * CF_CList_Pop (CF_CListNode_t **head)`  
*Remove the first node from a list and return it.*
- `void CF_CList_InsertAfter (CF_CListNode_t **head, CF_CListNode_t *start, CF_CListNode_t *after)`  
*Insert the given node into the last after the given start node.*
- `void CF_CList_Traverse (CF_CListNode_t *start, CF_CListFn_t fn, void *context)`  
*Traverse the entire list, calling the given function on all nodes.*
- `void CF_CList_Traverse_R (CF_CListNode_t *end, CF_CListFn_t fn, void *context)`  
*Reverse list traversal, starting from end, calling given function on all nodes.*

### 12.49.1 Detailed Description

The CF Application circular list header file

### 12.49.2 Macro Definition Documentation

#### 12.49.2.1 CF\_CLIST\_CONT #define CF\_CLIST\_CONT CF\_CListTraverse\_Status\_CONTINUE

Constant indicating to continue traversal.

Definition at line 38 of file cf\_clist.h.

#### 12.49.2.2 CF\_CLIST\_EXIT #define CF\_CLIST\_EXIT CF\_CListTraverse\_Status\_EXIT

Constant indicating to stop traversal.

Definition at line 39 of file cf\_clist.h.

#### 12.49.2.3 container\_of #define container\_of(

```
    ptr,  
    type,  
    member ) ((type *) ((char *) (ptr) - (char *) offsetof(type, member)))
```

Obtains a pointer to the parent structure.

Given a pointer to a CF\_CListNode\_t object which is known to be a member of a larger container, this converts the pointer to that of the parent.

Definition at line 69 of file cf\_clist.h.

### 12.49.3 Typedef Documentation

#### 12.49.3.1 CF\_CListFn\_t typedef CF\_CListTraverse\_Status\_t (\* CF\_CListFn\_t) (CF\_CListNode\_t \*node, void \*context)

Callback function type for use with [CF\\_CList\\_Traverse\(\)](#)

#### Parameters

<i>node</i>	Current node being traversed
<i>context</i>	Opaque pointer passed through from initial call

#### Returns

integer status code indicating whether to continue traversal

#### Return values

<a href="#">CF_CLIST_CONT</a>	Indicates to continue traversing the list
<a href="#">CF_CLIST_EXIT</a>	Indicates to stop traversing the list

Definition at line 81 of file cf\_clist.h.

**12.49.3.2 CF\_CListNode\_t** `typedef struct CF_CListNode CF_CListNode_t`  
Circular linked list node links.  
Definition at line 44 of file cf\_clist.h.

#### 12.49.4 Enumeration Type Documentation

##### 12.49.4.1 CF\_CListTraverse\_Status\_t `enum CF_CListTraverse_Status_t`

Enumerator

<code>CF_CListTraverse_Status_CONTINUE</code>	
<code>CF_CListTraverse_Status_EXIT</code>	

Definition at line 32 of file cf\_clist.h.

#### 12.49.5 Function Documentation

##### 12.49.5.1 CF\_CList\_InitNode() `void CF_CList_InitNode (` `CF_CListNode_t * node )`

Initialize a clist node.

**Assumptions, External Events, and Notes:**

node must not be NULL.

**Parameters**

<code>node</code>	Pointer to node structure to be initialized
-------------------	---

Definition at line 40 of file cf\_clist.c.

References CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CFDP\_InitEngine(), CF\_CList\_Remove(), and CF\_FreeTransaction().

##### 12.49.5.2 CF\_CList\_InsertAfter() `void CF_CList_InsertAfter (` `CF_CListNode_t ** head,` `CF_CListNode_t * start,` `CF_CListNode_t * after )`

Insert the given node into the last after the given start node.

**Assumptions, External Events, and Notes:**

head must not be NULL, node must not be NULL.

**Parameters**

<code>head</code>	Pointer to head of list to remove from
<code>start</code>	Pointer to node to insert
<code>after</code>	Pointer to position to insert after

Definition at line 167 of file cf\_clist.c.

References CF\_ASSERT, CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CList\_InsertAfter\_Ex().

```
12.49.5.3 CF_CList_InsertBack() void CF_CList_InsertBack (
    CF_CListNode_t ** head,
    CF_CListNode_t * node )
```

Insert the given node into the back of a list.

**Assumptions, External Events, and Notes:**

head must not be NULL, node must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 81 of file cf\_clist.c.

References CF\_ASSERT, CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CFDP\_InitEngine(), CF\_CFDP\_RecycleTransaction(), CF\_CList\_InsertBack\_Ex(), and CF\_MoveFromTransaction().

```
12.49.5.4 CF_CList_InsertFront() void CF_CList_InsertFront (
    CF_CListNode_t ** head,
    CF_CListNode_t * node )
```

Insert the given node into the front of a list.

**Assumptions, External Events, and Notes:**

head must not be NULL, node must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to insert into
<i>node</i>	Pointer to node to insert

Definition at line 52 of file cf\_clist.c.

References CF\_ASSERT, CF\_CListNode::next, and CF\_CListNode::prev.

```
12.49.5.5 CF_CList_Pop() CF_CListNode_t* CF_CList_Pop (
    CF_CListNode_t ** head )
```

Remove the first node from a list and return it.

**Assumptions, External Events, and Notes:**

head must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to remove from
-------------	--

**Returns**

The first node (now removed) in the list

**Return values**

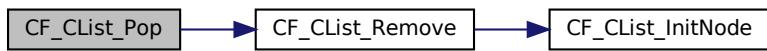
<i>NULL</i>	if list was empty.
-------------	--------------------

Definition at line 111 of file cf\_clist.c.

References CF\_ASSERT, and CF\_CList\_Remove().

Referenced by CF\_CFDP\_FindUnusedChunks().

Here is the call graph for this function:



**12.49.5.6 CF\_CList\_Remove()** void CF\_CList\_Remove (   
     CF\_CListNode\_t \*\* *head*,  
     CF\_CListNode\_t \* *node* )

Remove the given node from the list.

**Assumptions, External Events, and Notes:**

*head* must not be NULL, *node* must not be NULL.

**Parameters**

<i>head</i>	Pointer to head of list to remove from
<i>node</i>	Pointer to node to remove

Definition at line 132 of file cf\_clist.c.

References CF\_ASSERT, CF\_CList\_InitNode(), CF\_CListNode::next, and CF\_CListNode::prev.

Referenced by CF\_CList\_Pop(), CF\_CList\_Remove\_Ex(), CF\_DequeueTransaction(), and CF\_MoveTransaction().

Here is the call graph for this function:



**12.49.5.7 CF\_CList\_Traverse()** `void CF_CList_Traverse (`  
 `CF_CListNode_t * start,`  
 `CF_CListFn_t fn,`  
 `void * context )`

Traverse the entire list, calling the given function on all nodes.

#### Assumptions, External Events, and Notes:

start may be NULL, fn must be a valid function, context may be NULL.

#### Note

on traversal it's ok to delete the current node, but do not delete other nodes in the same list!!

#### Parameters

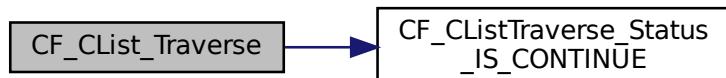
<code>start</code>	List to traverse (first node)
<code>fn</code>	Callback function to invoke for each node
<code>context</code>	Opaque pointer to pass to callback

Definition at line 188 of file cf\_clist.c.

References CF\_CListTraverse\_Status\_IS\_CONTINUE(), and CF\_CListNode::next.

Referenced by CF\_CFDP\_DisableEngine(), CF\_CFDP\_TickTransactions(), CF\_DoPurgeQueue(), CF\_FindTransactionBySequenceNumber(), CF\_TraverseAllTransactions(), CF\_WriteHistoryQueueDataToFile(), and CF\_WriteTxnQueueDataToFile().

Here is the call graph for this function:



```
12.49.5.8 CF_CList_Traverse_R() void CF_CList_Traverse_R (
```

CF_CListNode_t * end,	
CF_CListFn_t fn,	
void * context )	

Reverse list traversal, starting from end, calling given function on all nodes.

**Assumptions, External Events, and Notes:**

end may be NULL. fn must be a valid function, context may be NULL.

**Note**

traverse\_R will work backwards from the parameter's prev, and end on param

**Parameters**

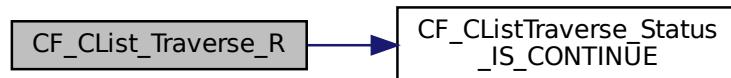
<i>end</i>	List to traverse (last node)
<i>fn</i>	Callback function to invoke for each node
<i>context</i>	Opaque pointer to pass to callback

Definition at line 228 of file cf\_clist.c.

References CF\_CListTraverse\_Status\_IS\_CONTINUE(), and CF\_CListNode::prev.

Referenced by CF\_InsertSortPrio().

Here is the call graph for this function:



```
12.49.5.9 CF_CListTraverse_Status_IS_CONTINUE() static bool CF_CListTraverse_Status_IS_CONTINUE (
```

CF_CListTraverse_Status_t stat ) [inline], [static]	
---	--

Checks if the list traversal should continue

Definition at line 44 of file cf\_clist.h.

Referenced by CF\_CList\_Traverse(), and CF\_CList\_Traverse\_R().

## 12.50 apps/cf/fsw/src(cf\_cmd.c File Reference

```
#include "cf_app.h"
#include "cf_verify.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_utils.h"
#include "cf_version.h"
#include "cf_platform_cfg.h"
```

```
#include "cf_cfdp.h"
#include "cf_cmd.h"
#include <string.h>
```

## Functions

- **CFE\_Status\_t CF\_NoopCmd (const CF\_NoopCmd\_t \*msg)**  
*The no-operation command.*
- **CFE\_Status\_t CF\_ResetCountersCmd (const CF\_ResetCountersCmd\_t \*msg)**  
*The reset counters command.*
- **CFE\_Status\_t CF\_TxFileCmd (const CF\_TxFileCmd\_t \*msg)**  
*Ground command to start a file transfer.*
- **CFE\_Status\_t CF\_PlaybackDirCmd (const CF\_PlaybackDirCmd\_t \*msg)**  
*Ground command to start directory playback.*
- **CF ChanAction\_Status\_t CF\_DoChanAction (const CF\_UnionArgs\_Payload\_t \*data, const char \*errstr, CF ChanActionFn\_t fn, void \*context)**  
*Common logic for all channel-based commands.*
- **CF ChanAction\_Status\_t CF\_DoFreezeThaw (uint8 chan\_num, void \*arg)**  
*Channel action to set the frozen bit for a channel.*
- **CFE\_Status\_t CF\_FreezeCmd (const CF\_FreezeCmd\_t \*msg)**  
*Freeze a channel.*
- **CFE\_Status\_t CF\_ThawCmd (const CF\_ThawCmd\_t \*msg)**  
*Thaw a channel.*
- **CF\_Transaction\_t \* CF\_FindTransactionBySequenceNumberAllChannels (CF\_TransactionSeq\_t ts, CF\_EntityId\_t eid)**  
*Search for a transaction across all channels.*
- **int32 CF\_TsnChanAction (const CF\_Transaction\_Payload\_t \*data, const char \*cmdstr, CF\_TsnChanAction\_fn\_t fn, void \*context)**  
*Common logic for all transaction sequence number and channel commands.*
- **void CF\_DoSuspRes\_Txn (CF\_Transaction\_t \*txn, CF\_ChанAction\_SuspResArg\_t \*context)**  
*Set the suspended bit in a transaction.*
- **void CF\_DoSuspRes (const CF\_Transaction\_Payload\_t \*payload, uint8 action)**  
*Handle transaction suspend and resume commands.*
- **CFE\_Status\_t CF\_SuspendCmd (const CF\_SuspendCmd\_t \*msg)**  
*Handle transaction suspend command.*
- **CFE\_Status\_t CF\_ResumeCmd (const CF\_ResumeCmd\_t \*msg)**  
*Handle transaction resume command.*
- **void CF\_Cancel\_TxnCmd (CF\_Transaction\_t \*txn, void \*ignored)**  
*tsn chan action to cancel a transaction.*
- **CFE\_Status\_t CF\_CancelCmd (const CF\_CancelCmd\_t \*msg)**  
*Handle a cancel ground command.*
- **void CF\_Abandon\_TxnCmd (CF\_Transaction\_t \*txn, void \*ignored)**  
*tsn chan action to abandon a transaction.*
- **CFE\_Status\_t CF\_AbandonCmd (const CF\_AbandonCmd\_t \*msg)**  
*Handle an abandon ground command.*
- **CF ChanAction\_Status\_t CF\_DoEnableDisableDequeue (uint8 chan\_num, void \*arg)**  
*Sets the dequeue enable/disable flag for a channel.*

- `CFE_Status_t CF_EnableDequeueCmd` (`const CF_EnableDequeueCmd_t *msg`)  
*Handle an enable dequeue ground command.*
- `CFE_Status_t CF_DisableDequeueCmd` (`const CF_DisableDequeueCmd_t *msg`)  
*Handle a disable dequeue ground command.*
- `CF_ChAction_Status_t CF_DoEnableDisablePolldir` (`uint8 chan_num, void *arg`)  
*Sets the enable/disable flag for the specified polling directory.*
- `CFE_Status_t CF_EnableDirPollingCmd` (`const CF_EnableDirPollingCmd_t *msg`)  
*Enable a polling dir ground command.*
- `CFE_Status_t CF_DisableDirPollingCmd` (`const CF_DisableDirPollingCmd_t *msg`)  
*Disable a polling dir ground command.*
- `CF_CListTraverse_Status_t CF_PurgeHistory` (`CF_CListNode_t *node, void *arg`)  
*Purge the history queue for the given channel.*
- `CF_CListTraverse_Status_t CF_PurgeTransaction` (`CF_CListNode_t *node, void *ignored`)  
*Purge the pending transaction queue.*
- `CF_ChAction_Status_t CF_DoPurgeQueue` (`uint8 chan_num, void *arg`)  
*Channel action command to perform purge queue operations.*
- `CFE_Status_t CF_PurgeQueueCmd` (`const CF_PurgeQueueCmd_t *msg`)  
*Ground command to purge either the history or pending queues.*
- `CFE_Status_t CF_WriteQueueCmd` (`const CF_WriteQueueCmd_t *msg`)  
*Ground command to write a file with queue information.*
- `CF_ChAction_Status_t CF_ValidateChunkSizeCmd` (`CF_ChunkSize_t val, uint8 chan_num`)  
*Checks if the value is less than or equal to the max PDU size.*
- `CF_ChAction_Status_t CF_ValidateMaxOutgoingCmd` (`uint32 val, uint8 chan_num`)  
*Checks if the value is within allowable range as outgoing packets per wakeup.*
- `void CF_GetSetParamCmd` (`bool is_set, CF_GetSet_ValueID_t param_id, uint32 value, uint8 chan_num`)  
*Perform a configuration get/set operation.*
- `CFE_Status_t CF_SetParamCmd` (`const CF_SetParamCmd_t *msg`)  
*Ground command to set a configuration parameter.*
- `CFE_Status_t CF_GetParamCmd` (`const CF_GetParamCmd_t *msg`)  
*Ground command to set a configuration parameter.*
- `CFE_Status_t CF_EnableEngineCmd` (`const CF_EnableEngineCmd_t *msg`)  
*Ground command enable engine.*
- `CFE_Status_t CF_DisableEngineCmd` (`const CF_DisableEngineCmd_t *msg`)  
*Ground command disable engine.*
- `CFE_Status_t CF_SendHkCmd` (`const CF_SendHkCmd_t *msg`)  
*Send CF housekeeping packet.*
- `CFE_Status_t CF_WakeupCmd` (`const CF_WakeupCmd_t *msg`)  
*CF wakeup function.*

### 12.50.1 Detailed Description

The CF Application command handling source file

All ground commands are processed in this file. All supporting functions necessary to process the commands are also here.

### 12.50.2 Function Documentation

```
12.50.2.1 CF_Abandon_TxnCmd() void CF_Abandon_TxnCmd (
    CF_Transaction_t * txn,
    void * ignored )
```

tsn chan action to abandon a transaction.

This helper function is used with [CF\\_TsnChanAction\(\)](#) to abandon matched transactions

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

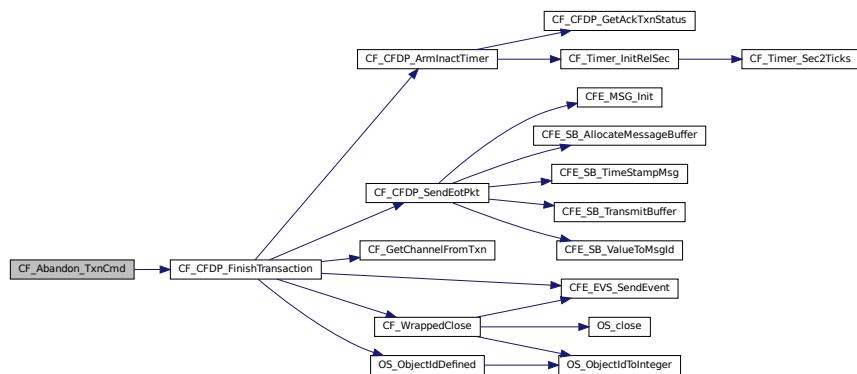
<i>txn</i>	Pointer to transaction object
<i>ignored</i>	Not used by this function

Definition at line 514 of file cf\_cmd.c.

References [CF\\_CFDP\\_FinishTransaction\(\)](#).

Referenced by [CF\\_AbandonCmd\(\)](#).

Here is the call graph for this function:



```
12.50.2.2 CF_AbandonCmd() CFE_Status_t CF_AbandonCmd (
    const CF_AbandonCmd_t * msg )
```

Handle an abandon ground command.

#### Assumptions, External Events, and Notes:

msg must not be NULL.

#### Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

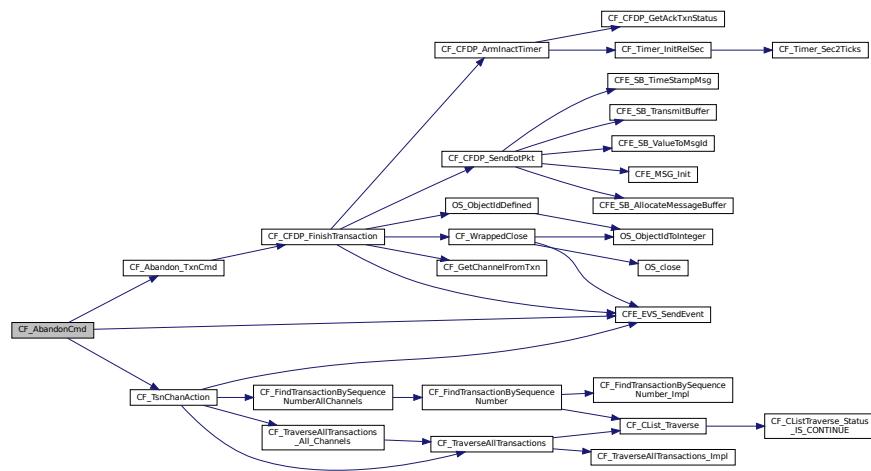
Definition at line 525 of file cf\_cmd.c.

References [CF\\_Abandon\\_TxnCmd\(\)](#), [CF\\_AppData](#), [CF\\_CMD\\_ABANDON\\_CHAN\\_ERR\\_EID](#), [CF\\_CMD\\_ABANDON\\_INF\\_EID](#), [CF\\_TsnChanAction\(\)](#), [CFE\\_EVS\\_EventType\\_ERROR](#), [CFE\\_EVS\\_EventType\\_INFORMATION](#), [CFE\\_EVS](#)

\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_AbandonCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.50.2.3 CF\_Cancel\_TxnCmd()** void CF\_Cancel\_TxnCmd (

```

    CFE_Transaction_t * txn,
    void * ignored )
  
```

tsn chan action to cancel a transaction.

This helper function is used with [CF\\_TsnChanAction\(\)](#) to cancel matched transactions

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<i>txn</i>	Pointer to transaction object
<i>ignored</i>	Not used by this function

Definition at line 479 of file cf\_cmd.c.

References CF\_CFDP\_CancelTransaction().

Referenced by CF\_CancelCmd().

Here is the call graph for this function:



**12.50.2.4 CF\_CancelCmd()** `CFE_Status_t CF_CancelCmd ( const CF_CancelCmd_t * msg )`

Handle a cancel ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

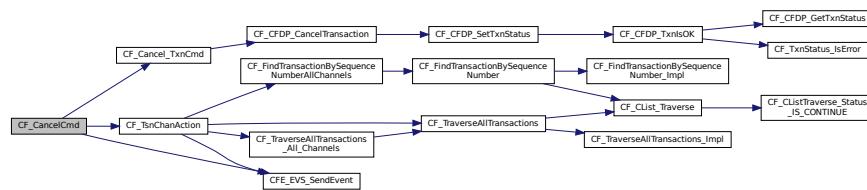
**Parameters**

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 490 of file cf\_cmd.c.

References CF\_AppData, CF\_Cancel\_TxnCmd(), CF\_CMD\_CANCEL\_CHAN\_ERR\_EID, CF\_CMD\_CANCEL\_INF\_EID, CF\_TsnChanAction(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_CancelCmd::Payload.

Here is the call graph for this function:



**12.50.2.5 CF\_DisableDequeueCmd()** `CFE_Status_t CF_DisableDequeueCmd ( const CF_DisableDequeueCmd_t * msg )`

Handle a disable dequeue ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

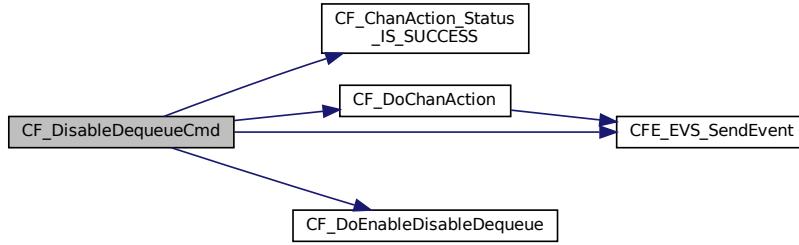
**Parameters**

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 588 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID, CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID, CF\_DoChanAction(), CF\_DoEnableDisableDequeue(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_DisableDequeueCmd::Payload.

Here is the call graph for this function:



**12.50.2.6 CF\_DisableDirPollingCmd()** `CFE_Status_t CF_DisableDirPollingCmd ( const CF_DisableDirPollingCmd_t * msg )`

Disable a polling dir ground command.

**Assumptions, External Events, and Notes:**

`msg` must not be NULL.

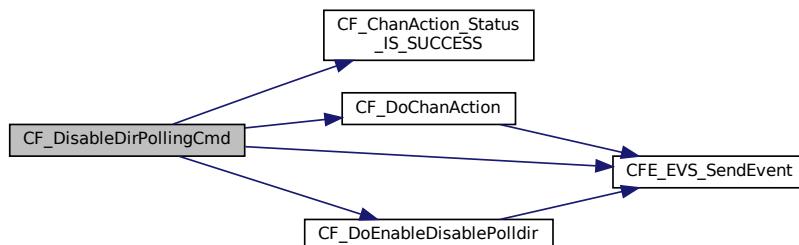
#### Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 673 of file cf\_cmd.c.

References `CF_AppData`, `CF_ChAction_Status_IS_SUCCESS()`, `CF_CMD_DISABLE_POLLDIR_ERR_EID`, `CF_CMD_DISABLE_POLLDIR_INF_EID`, `CF_DoChanAction()`, `CF_DoEnableDisablePolldir()`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_HkCmdCounters::cmd`, `CF_HkPacket_Payload::counters`, `CF_HkCmdCounters::err`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, and `CF_DisableDirPollingCmd::Payload`.

Here is the call graph for this function:



**12.50.2.7 CF\_DisableEngineCmd()** `CFE_Status_t CF_DisableEngineCmd ( const CF_DisableEngineCmd_t * msg )`

Ground command disable engine.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

**Parameters**

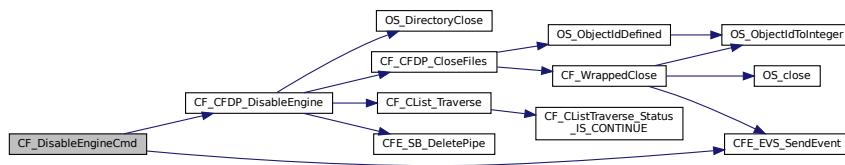
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1183 of file cf\_cmd.c.

References CF\_AppData, CF\_CFDP\_DisableEngine(), CF\_CMD\_DISABLE\_ENGINE\_INF\_EID, CF\_CMD\_ENG\_ALREADY\_DIS\_INF\_EID, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_Hk\_CmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_AppData\_t::hk, and CF\_HkPacket::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.50.2.8 CF\_DoChanAction()** `CF_ChanAction_Status_t CF_DoChanAction (`

```
const CF_UnionArgs_Payload_t * data,
const char * errstr,
CF_ChanActionFn_t fn,
void * context )
```

Common logic for all channel-based commands.

**Description**

All the commands that act on channels or have the special "all channels" parameter come through this function. This puts all common logic in one place. It does not handle the command accept or reject counters.

**Assumptions, External Events, and Notes:**

cmd must not be NULL, errstr must not be NULL, fn must be a valid function, context may be NULL.

**Parameters**

<code>data</code>	Pointer to payload being processed
<code>errstr</code>	String to be included in the EVS event if command should fail
<code>fn</code>	Callback action function to invoke for each affected channel
<code>context</code>	Opaque pointer to pass through to callback (not used in this function)

**Returns**

The return value from the given action function.

**Return values**

<i>CF_ChanAction_Status_ERROR</i>	on error
-----------------------------------	----------

Definition at line 221 of file cf\_cmd.c.

References CF\_UnionArgs\_Payload::byte, CF\_ALL\_CHANNELS, CF\_ChanAction\_Status\_ERROR, CF\_ChanAction\_Status\_SUCCESS, CF\_CMD\_CHAN\_PARAM\_ERR\_EID, CF\_NUM\_CHANNELS, CFE\_EVS\_EventType\_ERROR, and CFE\_EVS\_SendEvent().

Referenced by CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_FreezeCmd(), CF\_PurgeQueueCmd(), and CF\_ThawCmd().

Here is the call graph for this function:

**12.50.2.9 CF\_DoEnableDisableDequeue()** *CF\_ChanAction\_Status\_t* CF\_DoEnableDisableDequeue (

```

    uint8 chan_num,
    void * arg
)
```

Sets the dequeue enable/disable flag for a channel.

**Assumptions, External Events, and Notes:**

context must not be NULL.

**Parameters**

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be CF_ChanAction_BoolArg_t*

**Returns**

Always succeeds, so returns CF\_ChanAction\_Status\_SUCCESS.

Definition at line 549 of file cf\_cmd.c.

References CF\_ChanAction\_BoolArg::barg, CF\_AppData, CF\_ChanAction\_Status\_SUCCESS, CF\_ConfigTable::chan, CF\_AppData\_t::config\_table, and CF\_ChannelConfig::dequeue\_enabled.

Referenced by CF\_DisableDequeueCmd(), and CF\_EnableDequeueCmd().

**12.50.2.10 CF\_DoEnableDisablePolldir()** *CF\_ChanAction\_Status\_t* CF\_DoEnableDisablePolldir (

```
    uint8 chan_num,
    void * arg )
```

Sets the enable/disable flag for the specified polling directory.

**Assumptions, External Events, and Notes:**

context must not be NULL.

**Parameters**

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be CF_ChанAction_BoolMsgArg_t*

**Returns**

success/fail status code

**Return values**

<i>CF_ChанAction_Status_SUCCESS</i>	if successful
<i>CF_ChанAction_Status_ERROR</i>	if failed

Definition at line 613 of file cf\_cmd.c.

References CF\_ChанAction\_BoolMsgArg::barg, CF\_UnionArgs\_Payload::byte, CF\_ALL\_POLLDIRS, CF\_AppData, CF\_ChанAction\_Status\_ERROR, CF\_ChанAction\_Status\_SUCCESS, CF\_CMD\_POLLDIR\_INVALID\_ERR\_EID, CF\_MAX\_POLLING\_DIR\_PER\_CHAN, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_ConfigTable::chan, CF\_AppData\_t::config\_table, CF\_ChанAction\_BoolMsgArg::data, CF\_PollDir::enabled, and CF\_ChannelConfig::polldir. Referenced by CF\_DisableDirPollingCmd(), and CF\_EnableDirPollingCmd().

Here is the call graph for this function:



### 12.50.2.11 CF\_DoFreezeThaw() CF\_ChанAction\_Status\_t CF\_DoFreezeThaw (

```
    uint8 chan_num,
    void * arg )
```

Channel action to set the frozen bit for a channel.

**Assumptions, External Events, and Notes:**

context must not be NULL.

**Parameters**

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be CF_ChанAction_BoolArg_t*

**Returns**

Always succeeds, so returns CF\_ChAction\_Status\_SUCCESS.

Definition at line 257 of file cf\_cmd.c.

References CF\_ChAction\_BoolArg::barg, CF\_AppData, CF\_ChAction\_Status\_SUCCESS, CF\_HkPacket::Payload::channel\_hk, CF\_HkChannel\_Data::frozen, CF\_AppData\_t::hk, and CF\_HkPacket::Payload.

Referenced by CF\_FreezeCmd(), and CF\_ThawCmd().

**12.50.2.12 CF\_DoPurgeQueue()** [CF\\_ChAction\\_Status\\_t](#) CF\_DoPurgeQueue (   
    *uint8 chan\_num,*   
    *void \* arg* )

Channel action command to perform purge queue operations.

**Description**

Determines from the command parameters which queues to traverse and purge state.

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>chan_num</i>	CF channel number
<i>arg</i>	Pointer to purge queue command

**Returns**

integer status code indicating success or failure

**Return values**

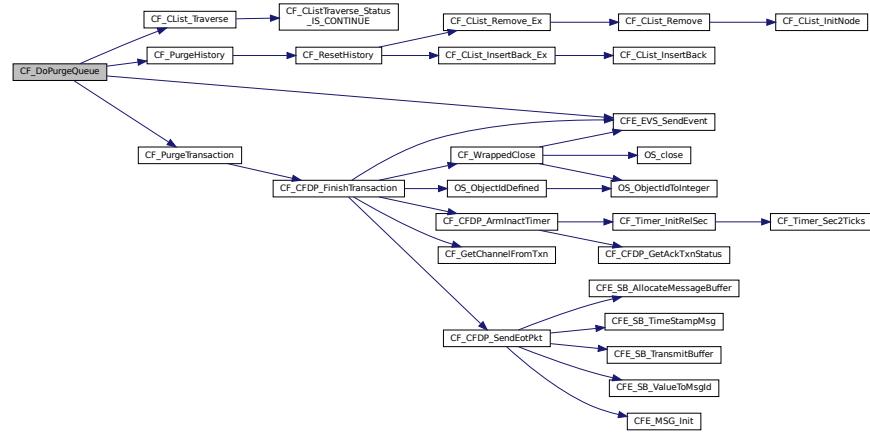
<i>CF_ChAction_Status_SUCCESS</i>	if successful
<i>CF_ChAction_Status_ERROR</i>	on error

Definition at line 727 of file cf\_cmd.c.

References CF\_UnionArgs\_Payload::byte, CF\_AppData, CF\_ChAction\_Status\_ERROR, CF\_ChAction\_Status::SUCCESS, CF\_CList\_Traverse(), CF\_CMD\_PURGE\_ARG\_ERR\_EID, CF\_PurgeHistory(), CF\_PurgeTransaction(), CF\_QueueIdx\_HIST, CF\_QueueIdx\_PEND, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Engine::channels, CF\_AppData\_t::engine, and CF\_Channel::qs.

Referenced by CF\_PurgeQueueCmd().

Here is the call graph for this function:



**12.50.2.13 CF\_DoSuspRes()**

```

void CF_DoSuspRes (
    const CF_Transaction_Payload_t * payload,
    uint8 action )
  
```

Handle transaction suspend and resume commands.

#### Description

This is called for both suspend and resume ground commands. It uses the `CF_TsnChanAction()` function to perform the command.

#### Assumptions, External Events, and Notes:

payload must not be NULL.

#### Parameters

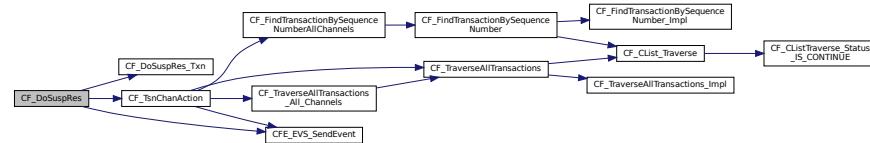
<code>payload</code>	Pointer to the command message
<code>action</code>	Action to take (suspend or resume)

Definition at line 414 of file cf\_cmd.c.

References `CF_AppData`, `CF_CMD_SUSPRES_CHAN_ERR_EID`, `CF_CMD_SUSPRES_INF_EID`, `CF_CMD_SUSPRES_SAME_INF_EID`, `CF_DoSuspRes_Txn()`, `CF_TsnChanAction()`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_EventType_INFORMATION`, `CFE_EVS_SendEvent()`, `CF_HkCmdCounters::cmd`, `CF_HkPacket_Payload::counters`, `CF_HkCmdCounters::err`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, and `CF_ChanAction_SuspResArg::same`.

Referenced by `CF_ResumeCmd()`, and `CF_SuspendCmd()`.

Here is the call graph for this function:



**12.50.2.14 CF\_DoSuspRes\_Txn()** `void CF_DoSuspRes_Txn (`  
 `CF_Transaction_t * txn,`  
 `CF_ChanAction_SuspResArg_t * context )`

Set the suspended bit in a transaction.

#### Assumptions, External Events, and Notes:

txn must not be NULL. context must not be NULL.

#### Parameters

<i>txn</i>	Pointer to the transaction object
<i>context</i>	Pointer to CF_ChanAction_SuspResArg_t structure from initial call

Definition at line 395 of file cf\_cmd.c.

References CF\_ChanAction\_SuspResArg::action, CF\_Assert, CF\_StateFlags::com, CF\_Transaction::flags, CF\_ChanAction\_SuspResArg::same, and CF\_Flags\_Common::suspended.

Referenced by CF\_DoSuspRes().

**12.50.2.15 CF\_EnableDequeueCmd()** `CFE_Status_t CF_EnableDequeueCmd (`  
 `const CF_EnableDequeueCmd_t * msg )`

Handle an enable dequeue ground command.

#### Assumptions, External Events, and Notes:

msg must not be NULL.

#### Parameters

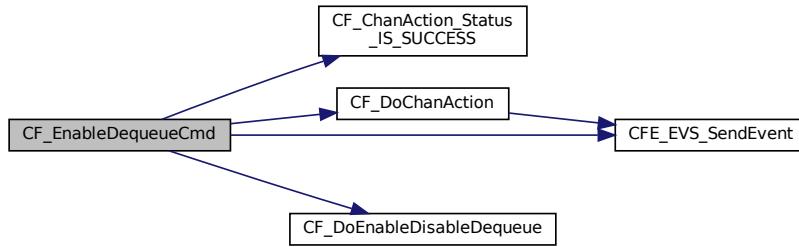
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 563 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID, CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID, CF\_DoChanAction(), CF\_DoEnableDisableDequeue(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_EnableDequeueCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.50.2.16 CF\_EnableDirPollingCmd()** `CFE_Status_t CF_EnableDirPollingCmd (const CF_EnableDirPollingCmd_t * msg )`

Enable a polling dir ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

#### Parameters

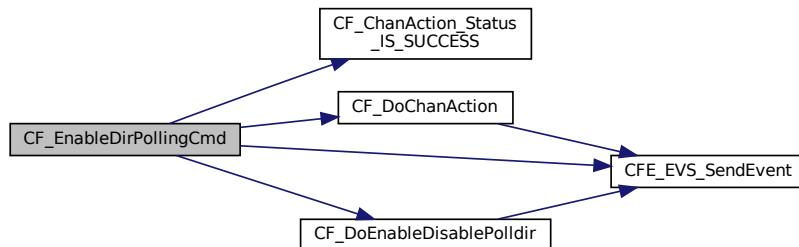
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 646 of file cf\_cmd.c.

References CF\_AppData, CF\_ChанAction\_Status\_IS\_SUCCESS(), CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID, CF\_CMD\_ENABLE\_POLLDIR\_INF\_EID, CF\_DoChanAction(), CF\_DoEnableDisablePolldir(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_EnableDirPollingCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



```
12.50.2.17 CF_EnableEngineCmd() CFE_Status_t CF_EnableEngineCmd (
    const CF_EnableEngineCmd_t * msg )
```

Ground command enable engine.

#### Assumptions, External Events, and Notes:

msg must not be NULL.

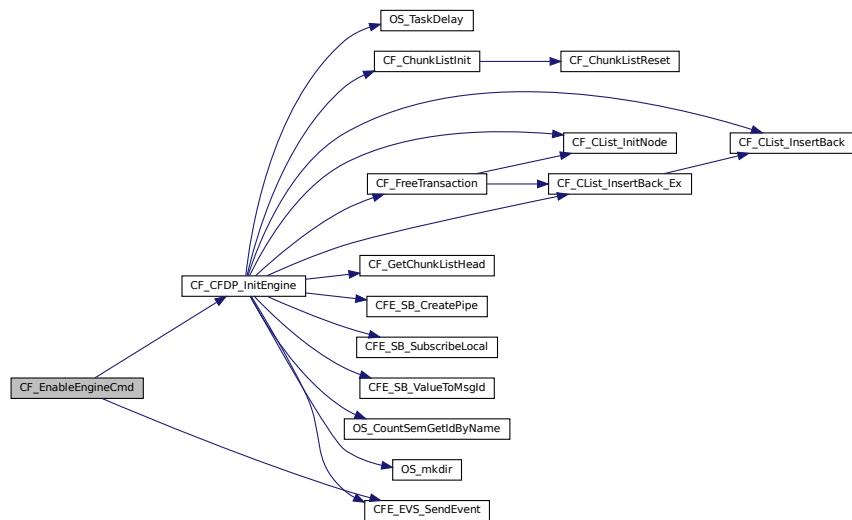
#### Parameters

msg	Pointer to command message
-----	----------------------------

Definition at line 1151 of file cf\_cmd.c.

References CF\_AppData, CF\_CFDP\_InitEngine(), CF\_CMD\_ENABLE\_ENGINE\_ERR\_EID, CF\_CMD\_ENABLE\_~ENGINE\_INF\_EID, CF\_CMD\_ENG\_ALREADY\_ENA\_INF\_EID, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_~Payload::counters, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, and CF\_HkPacket::Payload.

Here is the call graph for this function:



```
12.50.2.18 CF_FindTransactionBySequenceNumberAllChannels() CF_Transaction_t* CF_FindTransactionBySequenceNumberAllChannels (
    CF_TransactionSeq_t ts,
    CF_EntityId_t eid )
```

Search for a transaction across all channels.

#### Assumptions, External Events, and Notes:

None

**Parameters**

<i>ts</i>	Transaction sequence number to find
<i>eid</i>	Entity ID of the transaction

**Returns**

Pointer to the transaction, if found

**Return values**

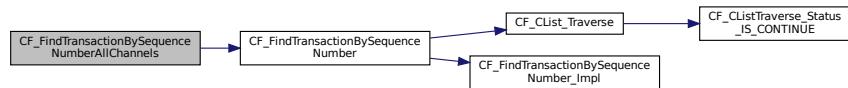
<i>NULL</i>	if transaction not found
-------------	--------------------------

Definition at line 319 of file cf\_cmd.c.

References CF\_AppData, CF\_FindTransactionBySequenceNumber(), CF\_NUM\_CHANNELS, CF\_Engine::channels, and CF\_AppData\_t::engine.

Referenced by CF\_TsnChanAction().

Here is the call graph for this function:



### 12.50.2.19 CF\_FreezeCmd() CFE\_Status\_t CF\_FreezeCmd (

const CF\_FreezeCmd\_t \* msg )

Freeze a channel.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

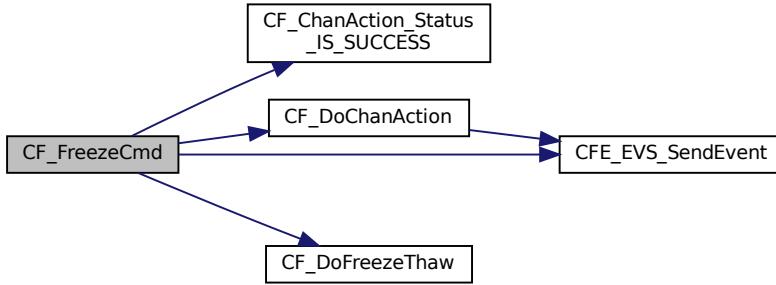
**Parameters**

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 271 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_FREEZE\_ERR\_EID, CF\_CMD\_FREEZE\_INF\_EID, CF\_DoChanAction(), CF\_DoFreezeThaw(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket::Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_FreezeCmd::Payload.

Here is the call graph for this function:



**12.50.2.20 CF\_GetParamCmd()** `CFE_Status_t CF_GetParamCmd (`  
`const CF_GetParamCmd_t * msg )`

Ground command to set a configuration parameter.

#### Assumptions, External Events, and Notes:

`msg` must not be NULL.

#### Parameters

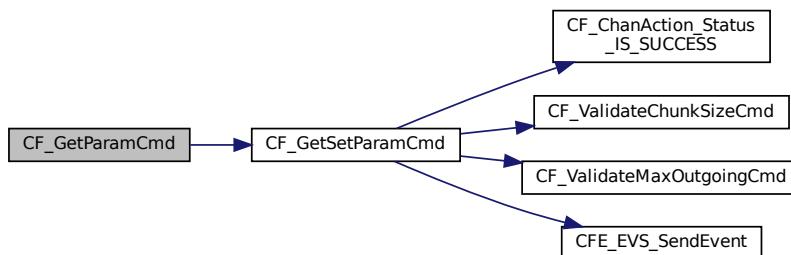
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1136 of file cf\_cmd.c.

References `CF_SetParamCmd()`, `CFE_SUCCESS`, `CF_GetParam_Payload::chan_num`, `CF_GetParam_Payload::key`, and `CF_GetParamCmd::Payload`.

Referenced by `CF_ProcessGroundCommand()`.

Here is the call graph for this function:



```
12.50.2.21 CF_SetParamCmd() void CF_SetParamCmd (
    bool is_set,
    CF_SetParam_ValueID_t param_id,
    uint32 value,
    uint8 chan_num )
```

Perform a configuration get/set operation.

For a set, this sets the value within the CF configuration. For a get, this generates an EVS event with the requested information.

#### Description

Combine get and set in one function with common logic.

#### Assumptions, External Events, and Notes:

None

#### Parameters

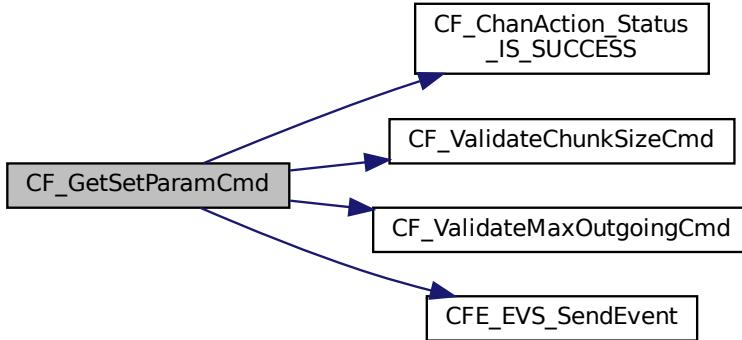
<i>is_set</i>	Whether to get (false) or set (true)
<i>param_id</i>	Parameter ID
<i>value</i>	Value to get/set
<i>chan_num</i>	Channel number to operate on

Definition at line 966 of file cf\_cmd.c.

References CF\_ChannelConfig::ack\_limit, CF\_ChannelConfig::ack\_timer\_s, CF\_AppData, CF\_ChAction\_Status\_IS\_SUCCESS(), CF\_CMD\_GETSET1\_INF\_EID, CF\_CMD\_GETSET2\_INF\_EID, CF\_CMD\_GETSET\_CHAN\_ERR\_EID, CF\_CMD\_GETSET\_PARAM\_ERR\_EID, CF\_CMD\_GETSET\_VALIDATE\_ERR\_EID, CF\_ERROR, CF\_SetParam\_ValueID\_ack\_limit, CF\_SetParam\_ValueID\_ack\_timer\_s, CF\_SetParam\_ValueID\_chan\_max\_outgoing\_messages\_per\_wakeup, CF\_SetParam\_ValueID\_inactivity\_timer\_s, CF\_SetParam\_ValueID\_local\_eid, CF\_SetParam\_ValueID\_nak\_limit, CF\_SetParam\_ValueID\_nak\_timer\_s, CF\_SetParam\_ValueID\_outgoing\_file\_chunk\_size, CF\_SetParam\_ValueID\_rx\_crc\_calc\_bytes\_per\_wakeup, CF\_SetParam\_ValueID\_ticks\_per\_second, CF\_NUM\_CHANNELS, CF\_ValidateChunkSizeCmd(), CF\_ValidateMaxOutgoingCmd(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_ConfigTable::chan, CF\_HkCmdCounters::cmd, CF\_AppData\_t::config\_table, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_ChannelConfig::inactivity\_timer\_s, CF\_ConfigTable::local\_eid, CF\_ChannelConfig::max\_outgoing\_messages\_per\_wakeup, CF\_ChannelConfig::nak\_limit, CF\_ChannelConfig::nak\_timer\_s, CF\_ConfigTable::outgoing\_file\_chunk\_size, CF\_HkPacket::Payload, CF\_ConfigTable::rx\_crc\_calc\_bytes\_per\_wakeup, and CF\_ConfigTable::ticks\_per\_second.

Referenced by CF\_GetParamCmd(), and CF\_SetParamCmd().

Here is the call graph for this function:



**12.50.2.22 CF\_NoopCmd()** `CFE_Status_t CF_NoopCmd (`  
    `const CF_NoopCmd_t * msg )`

The no-operation command.

#### Description

This function has a signature the same of all cmd\_ functions. This function simply prints an event message. Increments the command accept counter. The msg parameter is ignored in this one.

#### Assumptions, External Events, and Notes:

None

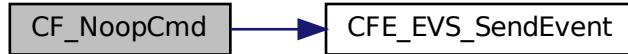
#### Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 48 of file cf\_cmd.c.

References CF\_AppData, CF\_MAJOR\_VERSION, CF\_MINOR\_VERSION, CF\_MISSION\_REV, CF\_NOOP\_INF\_EID, CF\_REVISION, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmd\_Counters::cmd, CF\_HkPacket\_Payload::counters, CF\_AppData\_t::hk, and CF\_HkPacket::Payload.

Here is the call graph for this function:



**12.50.2.23 CF\_PlaybackDirCmd()** `CF_Status_t CF_PlaybackDirCmd ( const CF_PlaybackDirCmd_t * msg )`

Ground command to start directory playback.

#### Description

This function has a signature the same of all cmd\_ functions. Increments the command accept or reject counter.

#### Assumptions, External Events, and Notes:

msg must not be NULL.

#### Parameters

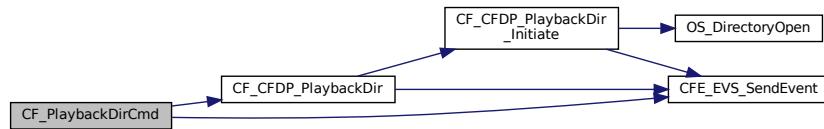
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 177 of file cf\_cmd.c.

References CF\_AppData, CF\_CFDP\_CLASS\_1, CF\_CFDP\_CLASS\_2, CF\_CFDP\_PlaybackDir(), CF\_CMD\_BAD\_PARAM\_ERR\_EID, CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID, CF\_CMD\_PLAYBACK\_DIR\_INF\_EID, CF\_NUM\_CHANNELS, CF\_TxFile\_Payload::cfdp\_class, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_TxFile\_Payload::chan\_num, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_TxFile\_Payload::dest\_id, CF\_TxFile\_Payload::dst\_filename, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_TxFile\_Payload::keep, CF\_HkPacket::Payload, CF\_PlaybackDirCmd::Payload, CF\_TxFile\_Payload::priority, and CF\_TxFile\_Payload::src\_filename.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.50.2.24 CF\_PurgeHistory()** `CF_CListTraverse_Status_t CF_PurgeHistory (`

```
CF_CListNode_t * node,
void * arg )
```

Purge the history queue for the given channel.

This helper function is used in conjunction with [CF\\_CList\\_Traverse\(\)](#)

#### Assumptions, External Events, and Notes:

node must not be NULL. chan must not be NULL.

#### Parameters

<i>node</i>	Current list node being traversed
<i>arg</i>	Channel pointer passed through as opaque object, must be CF_Channel_t*

#### Returns

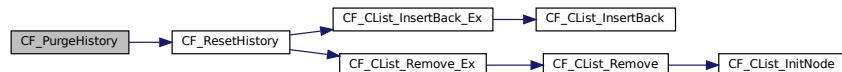
Always [CF\\_CLIST\\_CONT](#) to process all entries

Definition at line 700 of file cf\_cmd.c.

References [CF\\_CLIST\\_CONT](#), [CF\\_ResetHistory\(\)](#), and [container\\_of](#).

Referenced by [CF\\_DoPurgeQueue\(\)](#).

Here is the call graph for this function:



### 12.50.2.25 CF\_PurgeQueueCmd() [CFE\\_Status\\_t CF\\_PurgeQueueCmd \( const CF\\_PurgeQueueCmd\\_t \\* msg \)](#)

Ground command to purge either the history or pending queues.

#### Assumptions, External Events, and Notes:

msg must not be NULL.

#### Parameters

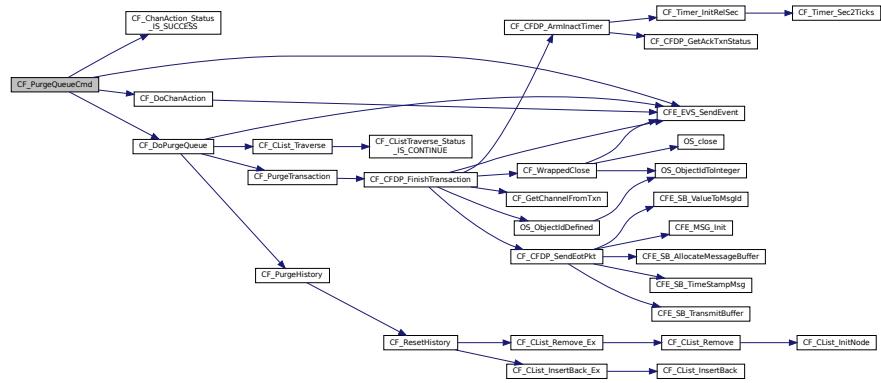
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 778 of file cf\_cmd.c.

References [CF\\_AppData](#), [CF\\_ChAction\\_Status\\_IS\\_SUCCESS\(\)](#), [CF\\_CMD\\_PURGE\\_QUEUE\\_ERR\\_EID](#), [CF\\_CMD\\_PURGE\\_QUEUE\\_INF\\_EID](#), [CF\\_DoChanAction\(\)](#), [CF\\_DoPurgeQueue\(\)](#), [CFE\\_EVT\\_EventType\\_ERROR](#), [CFE\\_EVT\\_EventType\\_INFORMATION](#), [CFE\\_EVS\\_SendEvent\(\)](#), [CFE\\_SUCCESS](#), [CF\\_HkCmdCounters::cmd](#), [CF\\_HkPacket\\_Payload::counters](#), [CF\\_HkCmdCounters::err](#), [CF\\_AppData\\_t::hk](#), [CF\\_HkPacket::Payload](#), and [CF\\_PurgeQueueCmd::Payload](#).

Referenced by [CF\\_ProcessGroundCommand\(\)](#).

Here is the call graph for this function:



**12.50.2.26 CF\_PurgeTransaction()** `CF_CListTraverse_Status_t CF_PurgeTransaction (`  
`CF_CListNode_t * node,`  
`void * ignored )`

Purge the pending transaction queue.

This helper function is used in conjunction with [CF\\_CList\\_Traverse\(\)](#)

**Assumptions, External Events, and Notes:**

node must not be NULL.

#### Parameters

<code>node</code>	Current list node being traversed
<code>ignored</code>	Not used by this implementation

#### Returns

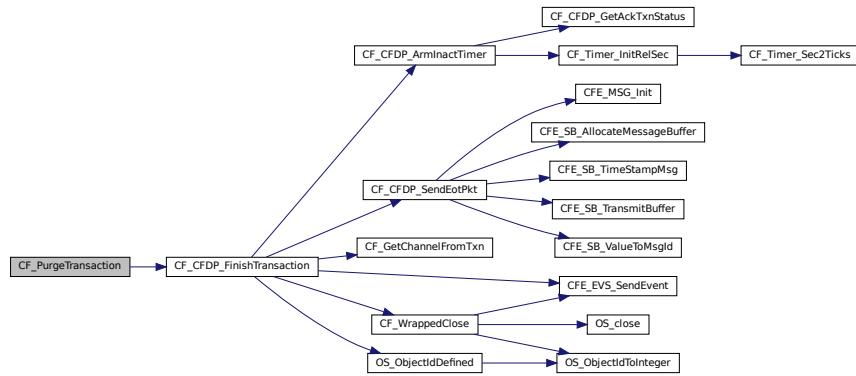
Always `CF_CLIST_CONT` to process all entries

Definition at line 714 of file cf\_cmd.c.

References `CF_CFDP_FinishTransaction()`, `CF_CLIST_CONT`, and `container_of`.

Referenced by `CF_DoPurgeQueue()`.

Here is the call graph for this function:



**12.50.2.27 CF\_ResetCountersCmd()** `CFE_Status_t CF_ResetCountersCmd ( const CF_ResetCountersCmd_t * msg )`

The reset counters command.

#### Description

This function has a signature the same of all cmd\_ functions. Resets the given counter, or all. Increments the command accept or reject counter. If the command counters are reset, then there is no increment.

#### Assumptions, External Events, and Notes:

msg must not be NULL.

#### Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 64 of file cf\_cmd.c.

References CF\_UnionArgs\_Payload::byte, CF\_AppData, CF\_CMD\_RESET\_INVALID\_ERR\_EID, CF\_NUM\_CHANNELS, CF\_Reset\_all, CF\_Reset\_command, CF\_Reset\_down, CF\_Reset\_fault, CF\_RESET\_INF\_EID, CF\_Reset\_up, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkPacket\_Payload::channel\_hk, CF\_HkCmdCounters::cmd, CF\_HkChannel\_Data::counters, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_HkCounters::fault, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_ResetCountersCmd::Payload, CF\_HkCounters::recv, and CF\_HkCounters::sent.  
Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.50.2.28 CF\_ResumeCmd()** `CF_Status_t CF_ResumeCmd ( const CF_ResumeCmd_t * msg )`

Handle transaction resume command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

#### Parameters

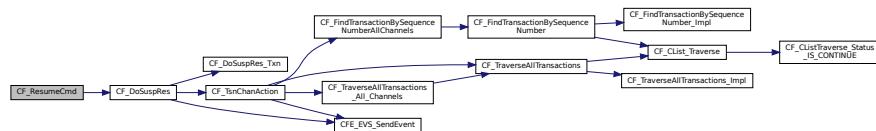
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 467 of file cf\_cmd.c.

References CF\_DoSuspRes(), CFE\_SUCCESS, and CF\_ResumeCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.50.2.29 CF\_SendHkCmd()** `CF_Status_t CF_SendHkCmd ( const CF_SendHkCmd_t * msg )`

Send CF housekeeping packet.

#### Description

The command to send the CF housekeeping packet

**Assumptions, External Events, and Notes:**

None

### Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

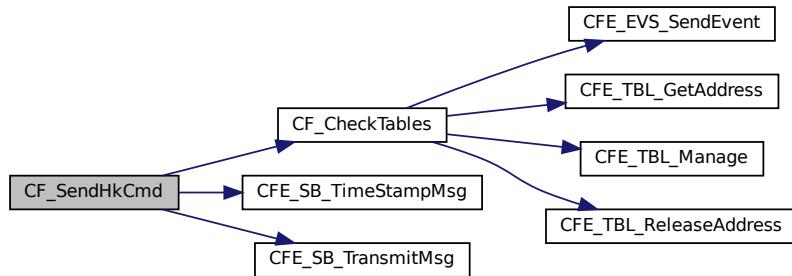
Definition at line 1207 of file cf\_cmd.c.

References CFE\_AppData, CFE\_CheckTables(), CFE\_SB\_TimeStampMsg(), CFE\_SB\_TransmitMsg(), CFE\_SUCCESS,

CF\_AppData\_t::hk, and CF\_HkPacket::TelemetryHeader.

Referenced by CF\_AppPipe().

Here is the call graph for this function:



**12.50.2.30 CF\_SetParamCmd()** `CFE_Status_t CF_SetParamCmd ( const CFE_SetParamCmd_t * msg )`

Ground command to set a configuration parameter.

**Assumptions, External Events, and Notes:**

`msg` must not be NULL.

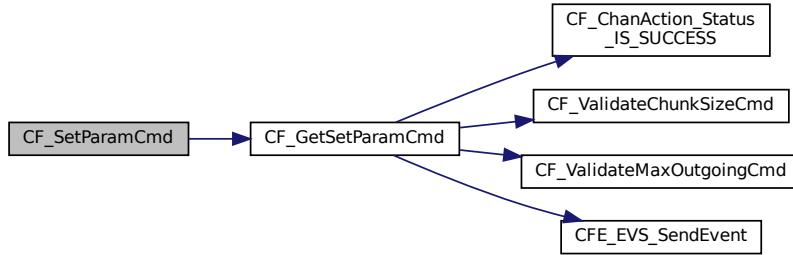
### Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1121 of file cf\_cmd.c.

References CFE\_GetSetParamCmd(), CFE\_SUCCESS, CF\_SetParam\_Payload::chan\_num, CF\_SetParam\_Payload::key, CF\_SetParamCmd::Payload, and CF\_SetParam\_Payload::value.

Here is the call graph for this function:



**12.50.2.31 CF\_SuspendCmd()** `CFE_Status_t CF_SuspendCmd ( const CF_SuspendCmd_t * msg )`

Handle transaction suspend command.

Assumptions, External Events, and Notes:

msg must not be NULL.

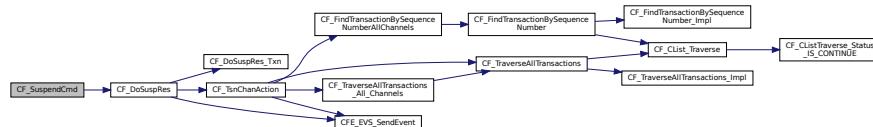
#### Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 455 of file cf\_cmd.c.

References CF\_DoSuspRes(), CFE\_SUCCESS, and CF\_SuspendCmd::Payload.

Here is the call graph for this function:



**12.50.2.32 CF\_ThawCmd()** `CFE_Status_t CF_ThawCmd ( const CF_ThawCmd_t * msg )`

Thaw a channel.

Assumptions, External Events, and Notes:

msg must not be NULL.

### Parameters

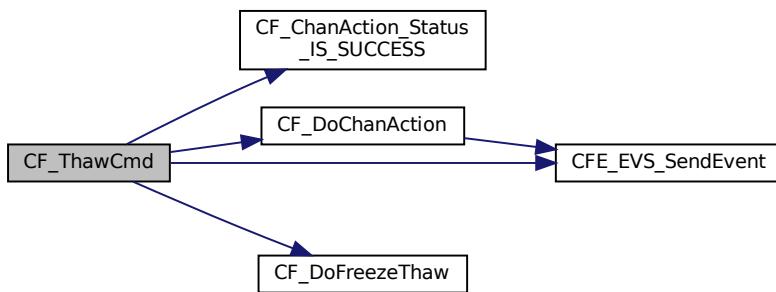
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 295 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_THAW\_ERR\_EID, CF\_CMD\_THAW\_INF\_EID, CF\_DoChanAction(), CF\_DoFreezeThaw(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_ThawCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



```

12.50.2.33 CF_TsnChanAction() int32 CF_TsnChanAction (
    const CF_Transaction_Payload_t * data,
    const char * cmdstr,
    CF_TsnChanAction_fn_t fn,
    void * context )
  
```

Common logic for all transaction sequence number and channel commands.

### Description

All the commands that on a transaction on a particular channel come through this function. This puts all common logic in one place. It does handle the command accept or reject counters.

### Assumptions, External Events, and Notes:

cmd must not be NULL, fn must be a valid function, context may be NULL.

### Parameters

<i>data</i>	Pointer to payload being processed
<i>cmdstr</i>	String to include in any generated EVS events
<i>fn</i>	Callback function to invoke for each matched transaction
<i>context</i>	Opaque object to pass through to the callback

**Returns**

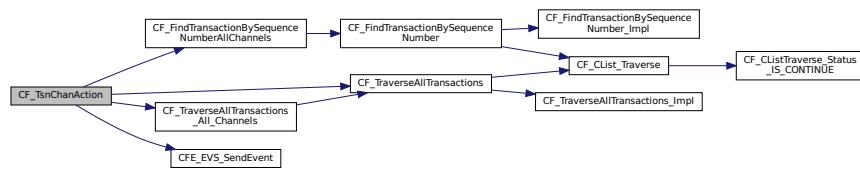
returns the number of transactions acted upon

Definition at line 347 of file cf\_cmd.c.

References CF\_ALL\_CHANNELS, CF\_AppData, CF\_CMD\_TRANS\_NOT\_FOUND\_ERR\_EID, CF\_CMD\_TSN\_←  
CHAN\_INVALID\_ERR\_EID, CF\_COMPOUND\_KEY, CF\_FindTransactionBySequenceNumberAllChannels(), CF←  
NUM\_CHANNELS, CF\_TraverseAllTransactions(), CF\_TraverseAllTransactions\_All\_Channels(), CFE\_EVS\_Event←  
Type\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction\_Payload::chan, CF\_Engine::channels, CF\_Transaction←  
Payload::eid, CF\_AppData\_t::engine, and CF\_Transaction\_Payload::ts.

Referenced by CF\_AbandonCmd(), CF\_CancelCmd(), and CF\_DoSuspRes().

Here is the call graph for this function:

**12.50.2.34 CF\_TxFileCmd() CFE\_Status\_t CF\_TxFileCmd (**

const CF\_TxFileCmd\_t \* msg )

Ground command to start a file transfer.

**Description**

This function has a signature the same of all cmd\_ functions. Increments the command accept or reject counter.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

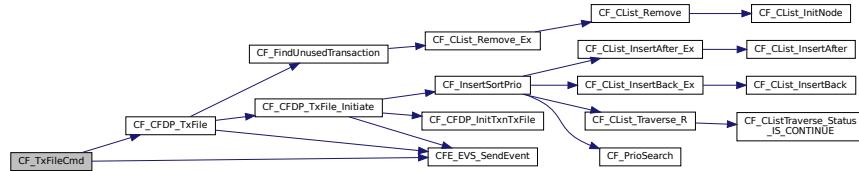
**Parameters**

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 134 of file cf\_cmd.c.

References CF\_AppData, CF\_CFDP\_CLASS\_1, CF\_CFDP\_CLASS\_2, CF\_CFDP\_TxFile(), CF\_CMD\_BAD\_←  
PARAM\_ERR\_EID, CF\_CMD\_TX\_FILE\_ERR\_EID, CF\_CMD\_TX\_FILE\_INF\_EID, CF\_NUM\_CHANNELS, CF←  
TxFile\_Payload::cfdp\_class, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS←  
SendEvent(), CFE\_SUCCESS, CF\_TxFile\_Payload::chan\_num, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload←  
::counters, CF\_TxFile\_Payload::dest\_id, CF\_TxFile\_Payload::dst\_filename, CF\_HkCmdCounters::err, CF\_AppData←  
\_t::hk, CF\_TxFile\_Payload::keep, CF\_HkPacket::Payload, CF\_TxFileCmd::Payload, CF\_TxFile\_Payload::priority, and  
CF\_TxFile\_Payload::src\_filename.

Here is the call graph for this function:



**12.50.2.35 CF\_ValidateChunkSizeCmd()** `CF_ChanAction_Status_t CF_ValidateChunkSizeCmd ( CF_ChunkSize_t val, uint8 chan_num )`

Checks if the value is less than or equal to the max PDU size.

**Assumptions, External Events, and Notes:**

None

#### Parameters

<code>val</code>	Size of chunk to test
<code>chan_num</code>	Ignored by this implementation

#### Returns

status code indicating if check passed

#### Return values

<code>CF_ChanAction_Status_SUCCESS</code>	if successful (val is less than or equal to max PDU)
<code>CF_ChanAction_Status_ERROR</code>	if failed (val is greater than max PDU)

Definition at line 931 of file cf\_cmd.c.

References `CF_ChanAction_Status_ERROR`, and `CF_ChanAction_Status_SUCCESS`.

Referenced by `CF_GetSetParamCmd()`.

**12.50.2.36 CF\_ValidateMaxOutgoingCmd()** `CF_ChanAction_Status_t CF_ValidateMaxOutgoingCmd ( uint32 val, uint8 chan_num )`

Checks if the value is within allowable range as outgoing packets per wakeup.

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>val</i>	Number to test
<i>chan_num</i>	CF channel number

**Returns**

status code indicating if check passed

**Return values**

<i>CF_ChanAction_Status_SUCCESS</i>	if successful (val is allowable as max packets per wakeup)
<i>CF_ChanAction_Status_ERROR</i>	if failed (val is not allowed)

Definition at line 947 of file cf\_cmd.c.

References CF\_AppData, CF\_ChанAction\_Status\_ERROR, CF\_ChанAction\_Status\_SUCCESS, CF\_ConfigTable<::chan, CF\_AppData\_t::config\_table, and CF\_ChannelConfig::sem\_name.

Referenced by CF\_GetSetParamCmd().

**12.50.2.37 CF\_WakeupCmd()** *CFE\_Status\_t* *CF\_WakeupCmd* (

```
const CF_WakeupCmd_t * msg )
```

CF wakeup function.

**Description**

Performs a single engine cycle for each wakeup

**Assumptions, External Events, and Notes:**

None

**Parameters**

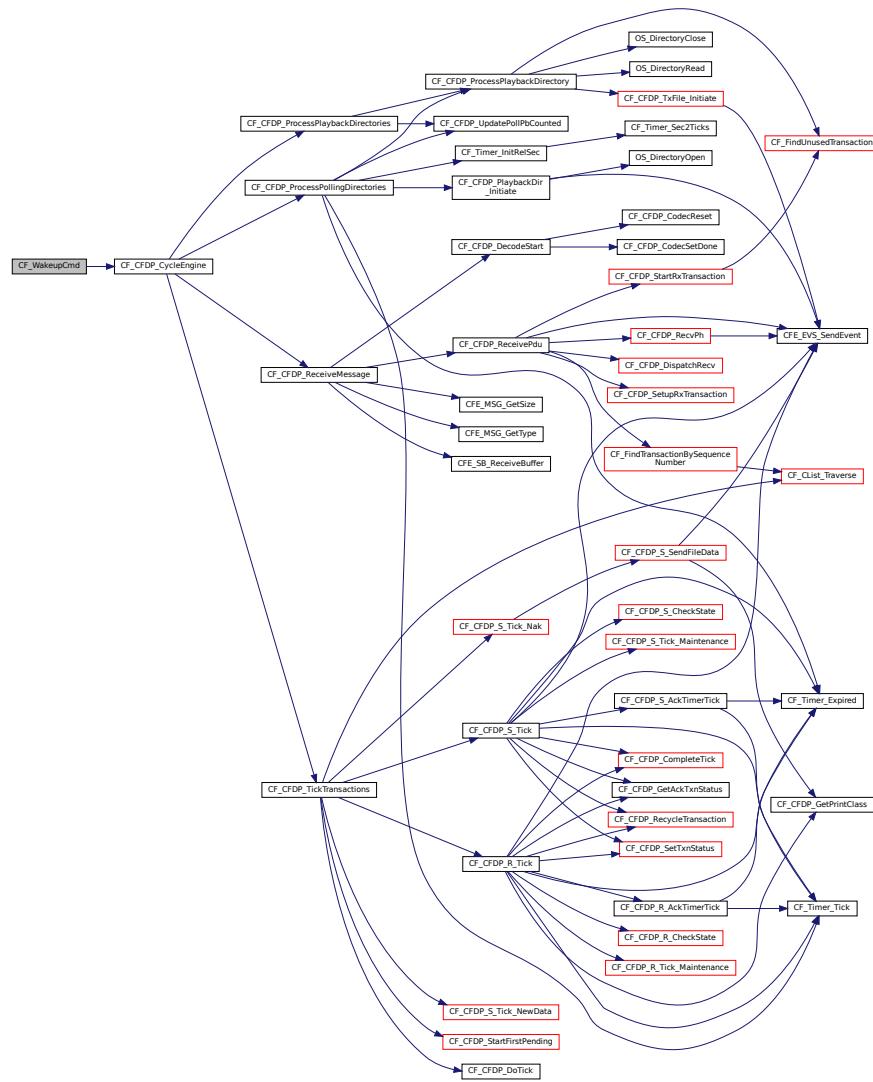
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1224 of file cf\_cmd.c.

References CF\_CFDP\_CycleEngine(), CF\_PERF\_ID\_CYCLE\_ENG, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and CFE\_SUCCESS.

Referenced by CF\_AppPipe().

Here is the call graph for this function:



**12.50.2.38 CF\_WriteQueueCmd()** CFE\_Status\_t CF\_WriteQueueCmd (

```
const CF_WriteQueueCmd_t * msg )
```

Ground command to write a file with queue information.

### **Assumptions, External Events, and Notes:**

msg must not be NULL.

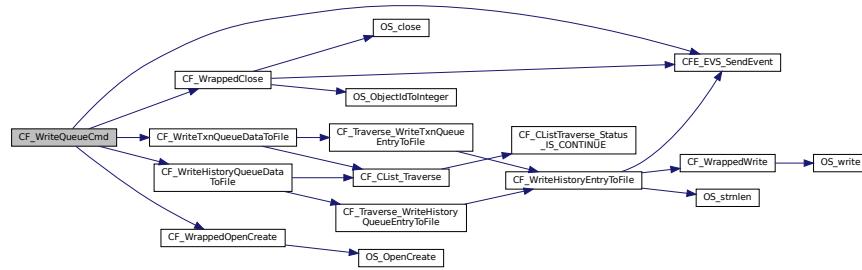
## Parameters

*msg* Pointer to command message

Definition at line 801 of file cf\_cmd.c.

References CF\_AppData, CF\_CMD\_WQ\_ARGS\_ERR\_EID, CF\_CMD\_WQ\_CHAN\_ERR\_EID, CF\_CMD\_WQ\_INF\_EID, CF\_CMD\_WQ\_OPEN\_ERR\_EID, CF\_CMD\_WQ\_WRITEHIST\_RX\_ERR\_EID, CF\_CMD\_WQ\_WRITEHIST\_TX\_ERR\_EID, CF\_CMD\_WQ\_WRITEQ\_PEND\_ERR\_EID, CF\_CMD\_WQ\_WRITEQ\_RX\_ERR\_EID, CF\_CMD\_WQ\_WRITEQ\_TX\_ERR\_EID, CF\_Direction\_RX, CF\_Direction\_TX, CF\_NUM\_CHANNELS, CF\_Queue\_active, CF\_Queue\_all, CF\_Queue\_history, CF\_Queue\_pend, CF\_QueueIdx\_PEND, CF\_QueueIdx\_RX, CF\_QueueIdx\_TX, CF\_Type\_all, CF\_Type\_down, CF\_Type\_up, CF\_WrappedClose(), CF\_WrappedOpenCreate(), CF\_WriteHistoryQueueDataToFile(), CF\_WriteTxnQueueDataToFile(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_WriteQueue\_Payload::chan, CF\_Engine::channels, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_AppData\_t::engine, CF\_HkCmdCounters::err, CF\_WriteQueue\_Payload::filename, CF\_AppData\_t::hk, OS\_FILE\_FLAG\_CREATE, OS\_FILE\_FLAG\_TRUNCATE, OS\_OBJECT\_ID\_UNDEFINED, OS\_WRITE\_ONLY, CF\_HkPacket::Payload, CF\_WriteQueueCmd::Payload, CF\_WriteQueue\_Payload::queue, and CF\_WriteQueue\_Payload::type.

Here is the call graph for this function:



## 12.51 apps/cf/fsw/src(cf\_cmd.h File Reference

```
#include "cfe.h"
#include "cf_app.h"
#include "cf_utils.h"
```

### Data Structures

- struct [CF\\_ChанAction\\_BoolArg](#)  
*An object to use with channel-scope actions requiring only a boolean argument.*
- struct [CF\\_ChанAction\\_SuspResArg](#)  
*An object to use with channel-scope actions for suspend/resume.*
- struct [CF\\_ChанAction\\_BoolMsgArg](#)  
*An object to use with channel-scope actions that require the message value.*
- struct [CF\\_ChанAction\\_MsgArg](#)  
*An object to use with channel-scope actions that require the message value.*

### TypeDefs

- typedef [CF\\_ChанAction\\_Status\\_t](#)(\* [CF\\_ChанActionFn\\_t](#)) (uint8 chan\_num, void \*context)  
*A callback function for use with [CF\\_DoChанAction\(\)](#)*
- typedef struct [CF\\_ChанAction\\_BoolArg](#) [CF\\_ChанAction\\_BoolArg\\_t](#)  
*An object to use with channel-scope actions requiring only a boolean argument.*
- typedef [CF\\_TraverseAllTransactions\\_fn\\_t](#) [CF\\_TsnChанAction\\_fn\\_t](#)

*A callback to use with transaction actions.*

- **typedef struct CF\_ChanAction\_SuspResArg** **CF\_ChanAction\_SuspResArg\_t**  
*An object to use with channel-scope actions for suspend/resume.*
- **typedef struct CF\_ChanAction\_BoolMsgArg** **CF\_ChanAction\_BoolMsgArg\_t**  
*An object to use with channel-scope actions that require the message value.*
- **typedef struct CF\_ChanAction\_MsgArg** **CF\_ChanAction\_MsgArg\_t**  
*An object to use with channel-scope actions that require the message value.*

## Enumerations

- enum **CF\_ChanAction\_Status\_t**{ **CF\_ChanAction\_Status\_SUCCESS** = 0 , **CF\_ChanAction\_Status\_ERROR** = -1 }

## Functions

- static bool **CF\_ChanAction\_Status\_IS\_SUCCESS** (**CF\_ChanAction\_Status\_t** stat)
- **CFE\_Status\_t CF\_SendHkCmd** (const **CF\_SendHkCmd\_t** \*msg)  
*Send CF housekeeping packet.*
- **CFE\_Status\_t CF\_WakeupCmd** (const **CF\_WakeupCmd\_t** \*msg)  
*CF wakeup function.*
- **CFE\_Status\_t CF\_NoopCmd** (const **CF\_NoopCmd\_t** \*msg)  
*The no-operation command.*
- **CFE\_Status\_t CF\_ResetCountersCmd** (const **CF\_ResetCountersCmd\_t** \*msg)  
*The reset counters command.*
- **CFE\_Status\_t CF\_TxFileCmd** (const **CF\_TxFileCmd\_t** \*msg)  
*Ground command to start a file transfer.*
- **CFE\_Status\_t CF\_PlaybackDirCmd** (const **CF\_PlaybackDirCmd\_t** \*msg)  
*Ground command to start directory playback.*
- **CF\_ChanAction\_Status\_t CF\_DoChanAction** (const **CF\_UnionArgs\_Payload\_t** \*data, const char \*errstr, **CF\_ChanActionFn\_t** fn, void \*context)  
*Common logic for all channel-based commands.*
- **CF\_ChanAction\_Status\_t CF\_DoFreezeThaw** (uint8 chan\_num, void \*arg)  
*Channel action to set the frozen bit for a channel.*
- **CFE\_Status\_t CF\_FreezeCmd** (const **CF\_FreezeCmd\_t** \*msg)  
*Freeze a channel.*
- **CFE\_Status\_t CF\_ThawCmd** (const **CF\_ThawCmd\_t** \*msg)  
*Thaw a channel.*
- **CF\_Transaction\_t \* CF\_FindTransactionBySequenceNumberAllChannels** (**CF\_TransactionSeq\_t** ts, **CF\_EntityId\_t** eid)  
*Search for a transaction across all channels.*
- **int32 CF\_TsnChanAction** (const **CF\_Transaction\_Payload\_t** \*data, const char \*cmdstr, **CF\_TsnChanAction\_fn\_t** fn, void \*context)  
*Common logic for all transaction sequence number and channel commands.*
- **void CF\_DoSuspRes\_Txn** (**CF\_Transaction\_t** \*txn, **CF\_ChanAction\_SuspResArg\_t** \*context)  
*Set the suspended bit in a transaction.*
- **void CF\_DoSuspRes** (const **CF\_Transaction\_Payload\_t** \*payload, uint8 action)  
*Handle transaction suspend and resume commands.*
- **CFE\_Status\_t CF\_SuspendCmd** (const **CF\_SuspendCmd\_t** \*msg)  
*Handle transaction suspend command.*

- `CFE_Status_t CF_ResumeCmd (const CF_ResumeCmd_t *msg)`  
*Handle transaction resume command.*
- `void CF_Cancel_TxnCmd (CF_Transaction_t *txn, void *ignored)`  
*tsn chan action to cancel a transaction.*
- `CFE_Status_t CF_CancelCmd (const CF_CancelCmd_t *msg)`  
*Handle a cancel ground command.*
- `void CF_Abandon_TxnCmd (CF_Transaction_t *txn, void *ignored)`  
*tsn chan action to abandon a transaction.*
- `CFE_Status_t CF_AbandonCmd (const CF_AbandonCmd_t *msg)`  
*Handle an abandon ground command.*
- `CF_ChanAction_Status_t CF_DoEnableDisableDequeue (uint8 chan_num, void *arg)`  
*Sets the dequeue enable/disable flag for a channel.*
- `CFE_Status_t CF_EnableDequeueCmd (const CF_EnableDequeueCmd_t *msg)`  
*Handle an enable dequeue ground command.*
- `CFE_Status_t CF_DisableDequeueCmd (const CF_DisableDequeueCmd_t *msg)`  
*Handle a disable dequeue ground command.*
- `CF_ChanAction_Status_t CF_DoEnableDisablePolldir (uint8 chan_num, void *arg)`  
*Sets the enable/disable flag for the specified polling directory.*
- `CFE_Status_t CF_EnableDirPollingCmd (const CF_EnableDirPollingCmd_t *msg)`  
*Enable a polling dir ground command.*
- `CFE_Status_t CF_DisableDirPollingCmd (const CF_DisableDirPollingCmd_t *msg)`  
*Disable a polling dir ground command.*
- `CF_CListTraverse_Status_t CF_PurgeHistory (CF_CListNode_t *node, void *arg)`  
*Purge the history queue for the given channel.*
- `CF_CListTraverse_Status_t CF_PurgeTransaction (CF_CListNode_t *node, void *ignored)`  
*Purge the pending transaction queue.*
- `CF_ChанAction_Status_t CF_DoPurgeQueue (uint8 chan_num, void *arg)`  
*Channel action command to perform purge queue operations.*
- `CFE_Status_t CF_PurgeQueueCmd (const CF_PurgeQueueCmd_t *msg)`  
*Ground command to purge either the history or pending queues.*
- `CFE_Status_t CF_WriteQueueCmd (const CF_WriteQueueCmd_t *msg)`  
*Ground command to write a file with queue information.*
- `CF_ChанAction_Status_t CF_ValidateChunkSizeCmd (CF_ChunkSize_t val, uint8 chan_num)`  
*Checks if the value is less than or equal to the max PDU size.*
- `CF_ChанAction_Status_t CF_ValidateMaxOutgoingCmd (uint32 val, uint8 chan_num)`  
*Checks if the value is within allowable range as outgoing packets per wakeup.*
- `void CF_GetSetParamCmd (bool is_set, CF_GetSet_ValueID_t param_id, uint32 value, uint8 chan_num)`  
*Perform a configuration get/set operation.*
- `CFE_Status_t CF_SetParamCmd (const CF_SetParamCmd_t *msg)`  
*Ground command to set a configuration parameter.*
- `CFE_Status_t CF_GetParamCmd (const CF_GetParamCmd_t *msg)`  
*Ground command to set a configuration parameter.*
- `CFE_Status_t CF_EnableEngineCmd (const CF_EnableEngineCmd_t *msg)`  
*Ground command enable engine.*
- `CFE_Status_t CF_DisableEngineCmd (const CF_DisableEngineCmd_t *msg)`  
*Ground command disable engine.*

### 12.51.1 Detailed Description

CF command processing function declarations

### 12.51.2 Typedef Documentation

**12.51.2.1 CF\_ChanAction\_BoolArg\_t** `typedef struct CF_ChanAction_BoolArg CF_ChanAction_BoolArg_t`  
An object to use with channel-scope actions requiring only a boolean argument.

**12.51.2.2 CF\_ChanAction\_BoolMsgArg\_t** `typedef struct CF_ChanAction_BoolMsgArg CF_ChanAction_BoolMsgArg_t`  
An object to use with channel-scope actions that require the message value.  
This combines a boolean action arg with the command message value

**12.51.2.3 CF\_ChanAction\_MsgArg\_t** `typedef struct CF_ChanAction_MsgArg CF_ChanAction_MsgArg_t`  
An object to use with channel-scope actions that require the message value.  
This combines a boolean action arg with the command message value

**12.51.2.4 CF\_ChanAction\_SuspResArg\_t** `typedef struct CF_ChanAction_SuspResArg CF_ChanAction_SuspResArg_t`  
An object to use with channel-scope actions for suspend/resume.  
This combines a boolean action arg with an output that indicates if it was a change or not.

**12.51.2.5 CF\_ChanActionFn\_t** `typedef CF_ChanAction_Status_t (* CF_ChanActionFn_t) (uint8 chan_num,  
void *context)`  
A callback function for use with `CF_DoChanAction()`

#### Parameters

<code>chan_num</code>	The CF channel number, for statistics purposes
<code>context</code>	Opaque object passed through from initial call

Definition at line 53 of file cf\_cmd.h.

**12.51.2.6 CF\_TsnChanAction\_fn\_t** `typedef CF_TraverseAllTransactions_fn_t CF_TsnChanAction_fn_t`  
A callback to use with transaction actions.  
For now this is the same as `CF_TraverseAllTransactions_fn_t`  
Definition at line 68 of file cf\_cmd.h.

### 12.51.3 Enumeration Type Documentation

**12.51.3.1 CF\_ChanAction\_Status\_t** `enum CF_ChanAction_Status_t`

#### Enumerator

<code>CF_ChanAction_Status_SUCCESS</code>	
<code>CF_ChanAction_Status_ERROR</code>	

Definition at line 33 of file cf\_cmd.h.

#### 12.51.4 Function Documentation

**12.51.4.1 CF\_Abandon\_TxnCmd()** void CF\_Abandon\_TxnCmd (

```
CF_Transaction_t * txn,
void * ignored )
```

tsn chan action to abandon a transaction.

This helper function is used with [CF\\_TsnChanAction\(\)](#) to abandon matched transactions

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

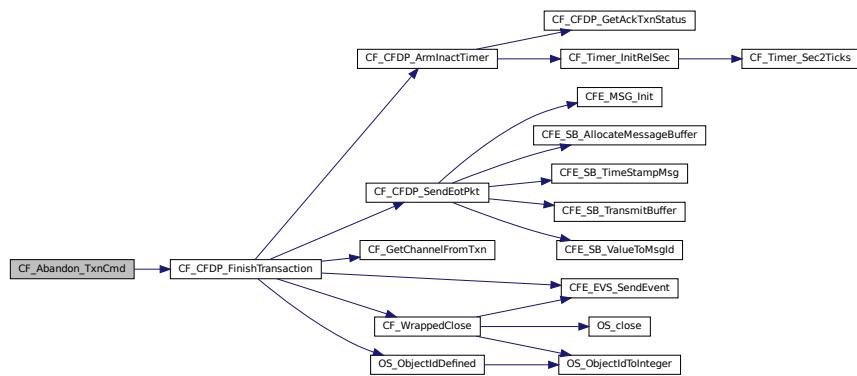
<i>txn</i>	Pointer to transaction object
<i>ignored</i>	Not used by this function

Definition at line 514 of file cf\_cmd.c.

References [CF\\_CFDP\\_FinishTransaction\(\)](#).

Referenced by [CF\\_AbandonCmd\(\)](#).

Here is the call graph for this function:



**12.51.4.2 CF\_AbandonCmd()** CFE\_Status\_t CF\_AbandonCmd (

```
const CFE_AbandonCmd_t * msg )
```

Handle an abandon ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

## Parameters

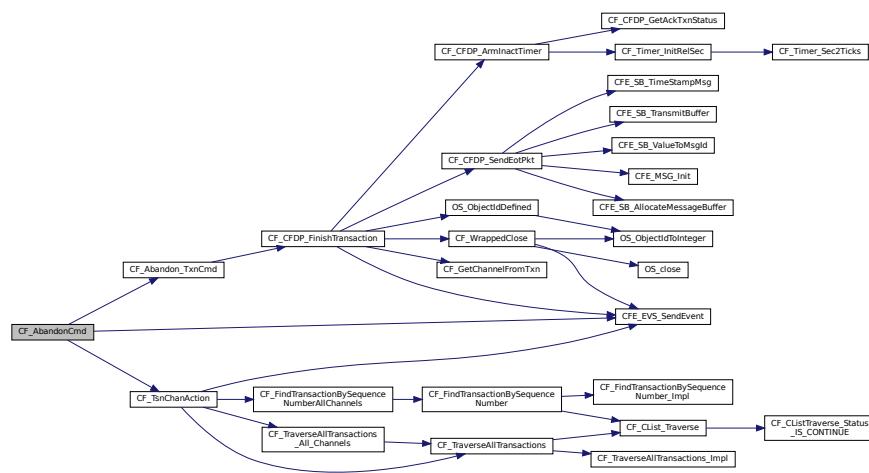
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 525 of file `cf_cmd.c`.

References CF\_Abandon\_TxnCmd(), CF\_AppData, CF\_CMD\_ABANDON\_CHAN\_ERR\_EID, CF\_CMD\_ABANDON\_INF\_EID, CF\_TsnChanAction(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_AbandonCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



#### 12.51.4.3 CF\_Cancel\_TxnCmd() void CF\_Cancel\_TxnCmd (

```
CF_Transaction_t * txn,  
void * ignored )
```

tsn chan action to cancel a transaction.

This helper function is used with `CF_TsnChanAction()` to cancel matched transactions

## **Assumptions, External Events, and Notes:**

txn must not be NULL.

## Parameters

<i>txn</i>	Pointer to transaction object
<i>ignored</i>	Not used by this function

Definition at line 479 of file cf\_cmd.c.

References CF CFDP CancelTransaction().

Referenced by CF CancelCmd().

Here is the call graph for this function:



**12.51.4.4 CF\_CancelCmd()** `CFE_Status_t CF_CancelCmd ( const CF_CancelCmd_t * msg )`

Handle a cancel ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

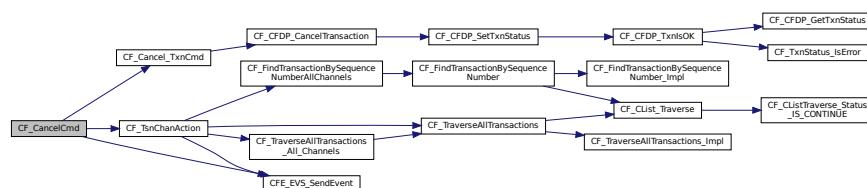
#### Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 490 of file cf\_cmd.c.

References CF\_AppData, CF\_Cancel\_TxnCmd(), CF\_CMD\_CANCEL\_CHAN\_ERR\_EID, CF\_CMD\_CANCEL\_INF\_EID, CF\_TsnChanAction(), CFE\_EVT\_EventType\_ERROR, CFE\_EVT\_EventType\_INFORMATION, CFE\_EVT\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_CancelCmd::Payload.

Here is the call graph for this function:



**12.51.4.5 CF\_ChAction\_Status\_IS\_SUCCESS()** `static bool CF_ChAction_Status_IS_SUCCESS ( CFE_ChAction_Status_t stat ) [inline], [static]`

Checks if the channel action was successful

Definition at line 42 of file cf\_cmd.h.

References CF\_ChAction\_Status\_SUCCESS.

Referenced by CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_FreezeCmd(), CF\_GetSetParamCmd(), CF\_PurgeQueueCmd(), and CF\_ThawCmd().

**12.51.4.6 CF\_DisableDequeueCmd()** `CFE_Status_t CF_DisableDequeueCmd ( const CF_DisableDequeueCmd_t * msg )`

Handle a disable dequeue ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

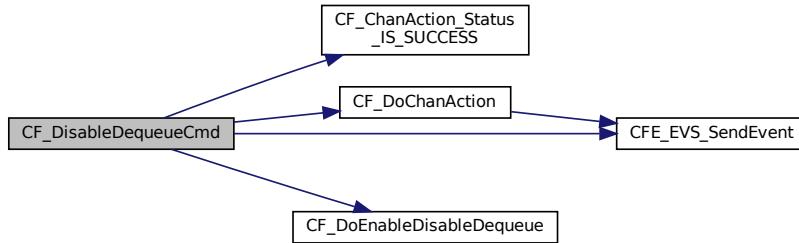
**Parameters**

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 588 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID, CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID, CF\_DoChanAction(), CF\_DoEnableDisableDequeue(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_DisableDequeueCmd::Payload.

Here is the call graph for this function:



**12.51.4.7 CF\_DisableDirPollingCmd()** `CFE_Status_t CF_DisableDirPollingCmd (`  
`const CF_DisableDirPollingCmd_t * msg )`

Disable a polling dir ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

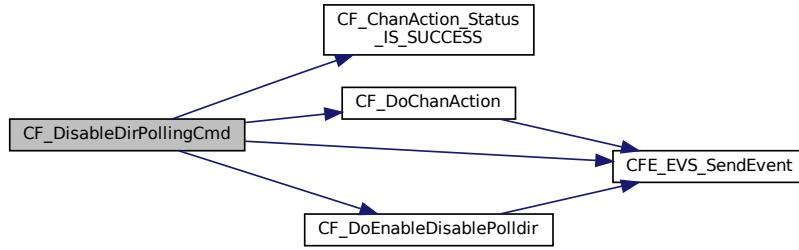
**Parameters**

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 673 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_DISABLE\_POLLDIR\_ERR\_EID, CF\_CMD\_DISABLE\_POLLDIR\_INF\_EID, CF\_DoChanAction(), CF\_DoEnableDisablePolldir(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_DisableDirPollingCmd::Payload.

Here is the call graph for this function:



**12.51.4.8 CF\_DisableEngineCmd()** `CFE_Status_t CF_DisableEngineCmd ( const CF_DisableEngineCmd_t * msg )`

Ground command disable engine.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

#### Parameters

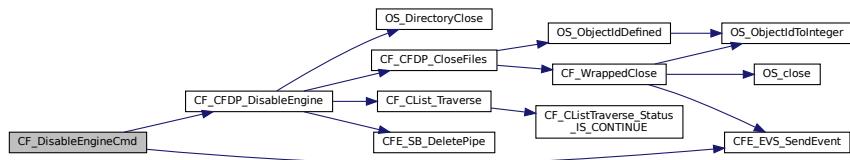
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1183 of file cf\_cmd.c.

References CF\_AppData, CF\_CFDP\_DisableEngine(), CF\_CMD\_DISABLE\_ENGINE\_INF\_EID, CF\_CMD\_ENG\_← ALREADY\_DIS\_INF\_EID, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_Engine::enabled, CF\_AppData\_t::engine, CF\_AppData\_t::hk, and CF\_HkPacket::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.9 CF\_DoChanAction()** `CF_ChанAction_Status_t CF_DoChanAction (`

```

const CF_UnionArgs_Payload_t * data,
const char * errstr,
```

```
CF_ChанActionFn_t fn,
void * context )
```

Common logic for all channel-based commands.

#### Description

All the commands that act on channels or have the special "all channels" parameter come through this function. This puts all common logic in one place. It does not handle the command accept or reject counters.

#### Assumptions, External Events, and Notes:

cmd must not be NULL, errstr must not be NULL, fn must be a valid function, context may be NULL.

#### Parameters

<i>data</i>	Pointer to payload being processed
<i>errstr</i>	String to be included in the EVS event if command should fail
<i>fn</i>	Callback action function to invoke for each affected channel
<i>context</i>	Opaque pointer to pass through to callback (not used in this function)

#### Returns

The return value from the given action function.

#### Return values

<i>CF_ChанAction_Status_ERROR</i>	on error
-----------------------------------	----------

Definition at line 221 of file cf\_cmd.c.

References CF\_UnionArgs\_Payload::byte, CF\_ALL\_CHANNELS, CF\_ChанAction\_Status\_ERROR, CF\_ChанAction\_Status\_SUCCESS, CF\_CMD\_CHAN\_PARAM\_ERR\_EID, CF\_NUM\_CHANNELS, CFE\_EVS\_EventType\_ERROR, and CFE\_EVS\_SendEvent().

Referenced by CF\_DisableDequeueCmd(), CF\_DisableDirPollingCmd(), CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_FreezeCmd(), CF\_PurgeQueueCmd(), and CF\_ThawCmd().

Here is the call graph for this function:



**12.51.4.10 CF\_DoEnableDisableDequeue()** *CF\_ChанAction\_Status\_t* CF\_DoEnableDisableDequeue (  
    *uint8 chan\_num,*  
    *void \* arg*)

Sets the dequeue enable/disable flag for a channel.

**Assumptions, External Events, and Notes:**

context must not be NULL.

**Parameters**

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be CF_ChanAction_BoolArg_t*

**Returns**

Always succeeds, so returns CF\_ChanAction\_Status\_SUCCESS.

Definition at line 549 of file cf\_cmd.c.

References CF\_ChanAction\_BoolArg::barg, CF\_AppData, CF\_ChanAction\_Status\_SUCCESS, CF\_ConfigTable::chan, CF\_AppData\_t::config\_table, and CF\_ChannelConfig::dequeue\_enabled.

Referenced by CF\_DisableDequeueCmd(), and CF\_EnableDequeueCmd().

**12.51.4.11 CF\_DoEnableDisablePolldir() CF\_ChanAction\_Status\_t CF\_DoEnableDisablePolldir (**

```
    uint8 chan_num,
    void * arg )
```

Sets the enable/disable flag for the specified polling directory.

**Assumptions, External Events, and Notes:**

context must not be NULL.

**Parameters**

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be CF_ChanAction_BoolMsgArg_t*

**Returns**

success/fail status code

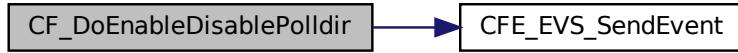
**Return values**

<i>CF_ChanAction_Status_SUCCESS</i>	if successful
<i>CF_ChanAction_Status_ERROR</i>	if failed

Definition at line 613 of file cf\_cmd.c.

References CF\_ChanAction\_BoolMsgArg::barg, CF\_UnionArgs\_Payload::byte, CF\_ALL\_POLLDIRS, CF\_AppData, CF\_ChanAction\_Status\_ERROR, CF\_ChanAction\_Status\_SUCCESS, CF\_CMD\_POLLDIR\_INVALID\_ERR\_EID, CF\_MAX\_POLLING\_DIR\_PER\_CHAN, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_ConfigTable::chan, CF\_AppData\_t::config\_table, CF\_ChanAction\_BoolMsgArg::data, CF\_PollDir::enabled, and CF\_ChannelConfig::polldir. Referenced by CF\_DisableDirPollingCmd(), and CF\_EnableDirPollingCmd().

Here is the call graph for this function:



**12.51.4.12 CF\_DoFreezeThaw()** `CF_ChanAction_Status_t CF_DoFreezeThaw (`  
 `uint8 chan_num,`  
 `void * arg )`

Channel action to set the frozen bit for a channel.

**Assumptions, External Events, and Notes:**

context must not be NULL.

#### Parameters

<i>chan_num</i>	channel number
<i>arg</i>	context pointer to object, must be <code>CF_ChanAction_BoolArg_t*</code>

#### Returns

Always succeeds, so returns `CF_ChanAction_Status_SUCCESS`.

Definition at line 257 of file `cf_cmd.c`.

References `CF_ChanAction_BoolArg::barg`, `CF_AppData`, `CF_ChanAction_Status_SUCCESS`, `CF_HkPacket::Payload::channel_hk`, `CF_HkChannel_Data::frozen`, `CF_AppData_t::hk`, and `CF_HkPacket::Payload`.  
Referenced by `CF_FreezeCmd()`, and `CF_ThawCmd()`.

**12.51.4.13 CF\_DoPurgeQueue()** `CF_ChanAction_Status_t CF_DoPurgeQueue (`  
 `uint8 chan_num,`  
 `void * arg )`

Channel action command to perform purge queue operations.

#### Description

Determines from the command parameters which queues to traverse and purge state.

**Assumptions, External Events, and Notes:**

None

#### Parameters

<i>chan_num</i>	CF channel number
<i>arg</i>	Pointer to purge queue command

**Returns**

integer status code indicating success or failure

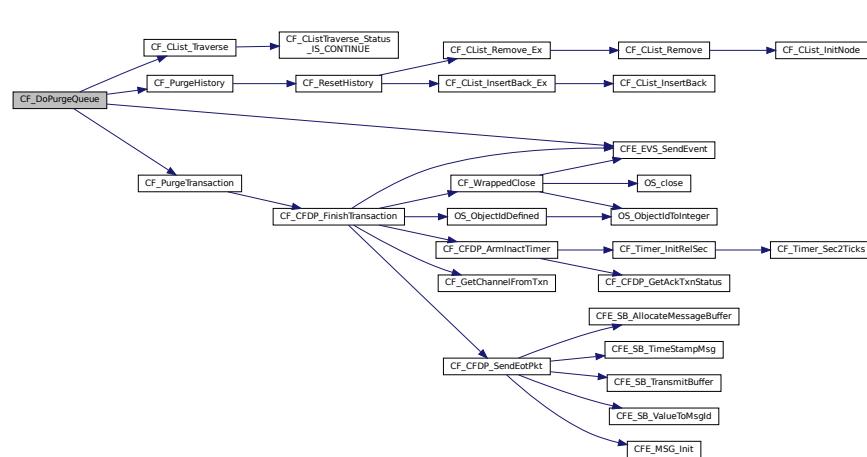
**Return values**

<i>CF_ChanAction_Status_SUCCESS</i>	if successful
<i>CF_ChanAction_Status_ERROR</i>	on error

Definition at line 727 of file cf\_cmd.c.

References CF\_UnionArgs\_Payload::byte, CF\_AppData, CF\_ChanAction\_Status\_ERROR, CF\_ChanAction\_Status\_SUCCESS, CF\_CList\_Traverse(), CF\_CMD\_PURGE\_ARG\_ERR\_EID, CF\_PurgeHistory(), CF\_PurgeTransaction(), CF\_QueueIdx\_HIST, CF\_QueueIdx\_PEND, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CF\_Engine::channels, CF\_AppData\_t::engine, and CF\_Channel::qs.  
Referenced by CF\_PurgeQueueCmd().

Here is the call graph for this function:



**12.51.4.14 CF\_DoSuspRes()** void CF\_DoSuspRes (   
 const CF\_Transaction\_Payload\_t \* payload,  
 uint8 action )

Handle transaction suspend and resume commands.

**Description**

This is called for both suspend and resume ground commands. It uses the [CF\\_TsnChanAction\(\)](#) function to perform the command.

**Assumptions, External Events, and Notes:**

payload must not be NULL.

**Parameters**

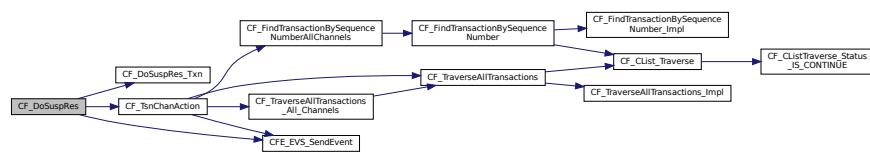
<i>payload</i>	Pointer to the command message
<i>action</i>	Action to take (suspend or resume)

Definition at line 414 of file cf\_cmd.c.

References CF\_AppData, CF\_CMD\_SUSPRES\_CHAN\_ERR\_EID, CF\_CMD\_SUSPRES\_INF\_EID, CF\_CMD\_SUSPRES\_SAME\_INF\_EID, CF\_DoSuspRes\_Txn(), CF\_TsnChanAction(), CFE\_EVT\_EventType\_ERROR, CFE\_EVT\_EventType\_INFORMATION, CFE\_EVT\_SendEvent(), CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_ChAction\_SuspResArg::same.

Referenced by CF\_ResumeCmd(), and CF\_SuspendCmd().

Here is the call graph for this function:



**12.51.4.15 CF\_DoSuspRes\_Txn()** void CF\_DoSuspRes\_Txn (   
`CF_Transaction_t * txn,`  
`CF_ChAction_SuspResArg_t * context )`

Set the suspended bit in a transaction.

**Assumptions, External Events, and Notes:**

txn must not be NULL. context must not be NULL.

**Parameters**

<i>txn</i>	Pointer to the transaction object
<i>context</i>	Pointer to CF_ChAction_SuspResArg_t structure from initial call

Definition at line 395 of file cf\_cmd.c.

References CF\_ChAction\_SuspResArg::action, CF\_ASSERT, CF\_StateFlags::com, CF\_Transaction::flags, CF\_ChAction\_SuspResArg::same, and CF\_Flags\_Common::suspended.

Referenced by CF\_DoSuspRes().

**12.51.4.16 CF\_EnableDequeueCmd()** CFE\_Status\_t CF\_EnableDequeueCmd (   
`const CF_EnableDequeueCmd_t * msg` )

Handle an enable dequeue ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

**Parameters**

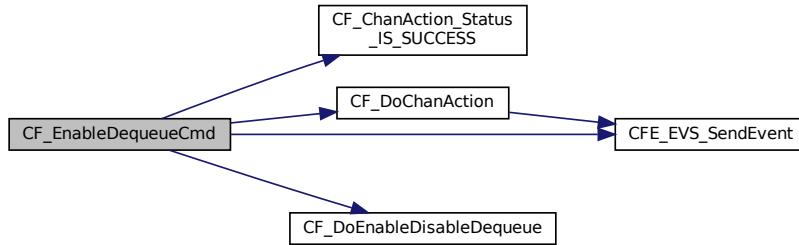
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 563 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID, CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID, CF\_DoChanAction(), CF\_DoEnableDisableDequeue(), CFE\_EVS\_EventType\_Error, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_EnableDequeueCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



#### 12.51.4.17 CF\_EnableDirPollingCmd() [CFE\\_Status\\_t CF\\_EnableDirPollingCmd \(const CF\\_EnableDirPollingCmd\\_t \\* msg \)](#)

Enable a polling dir ground command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

#### Parameters

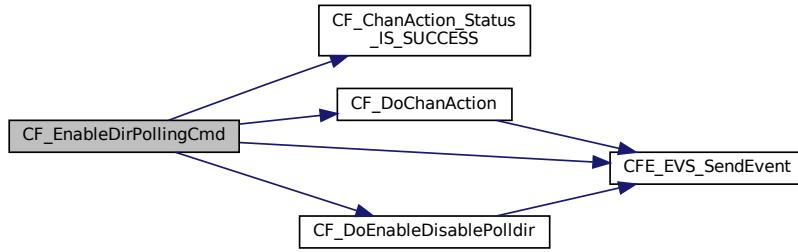
msg	Pointer to command message
-----	----------------------------

Definition at line 646 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID, CF\_CMD\_ENABLE\_POLLDIR\_INF\_EID, CF\_DoChanAction(), CF\_DoEnableDisablePolldir(), CFE\_EVS\_EventType\_Error, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_EnableDirPollingCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.18 `CF_EnableEngineCmd()`** `CFE_Status_t CF_EnableEngineCmd (`  
`const CF_EnableEngineCmd_t * msg )`

Ground command enable engine.

**Assumptions, External Events, and Notes:**

`msg` must not be NULL.

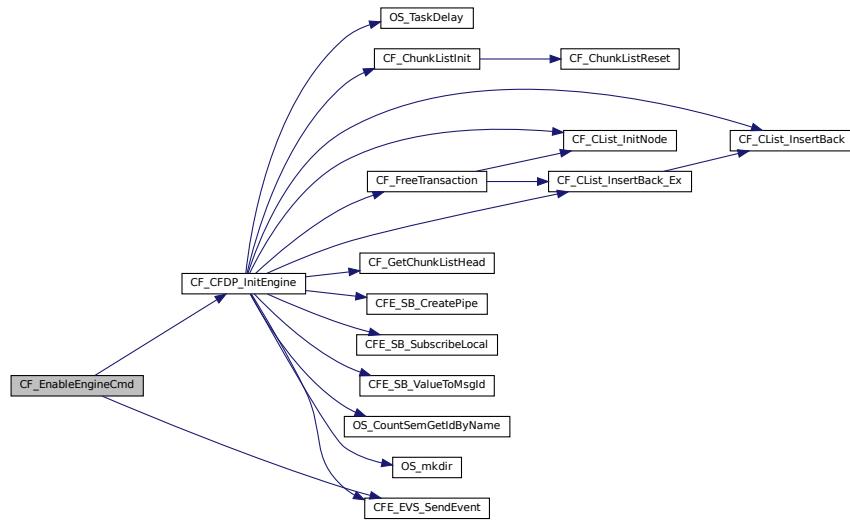
#### Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 1151 of file cf\_cmd.c.

References `CF_AppData`, `CF_CFDP_InitEngine()`, `CF_CMD_ENABLE_ENGINE_ERR_EID`, `CF_CMD_ENABLE_`  
`ENGINE_INF_EID`, `CF_CMD_ENG_ALREADY_ENA_INF_EID`, `CFE_EVS_EventType_ERROR`, `CFE_EVS_Event`  
`Type_INFORMATION`, `CFE_EVS_SendEvent()`, `CFE_SUCCESS`, `CF_HkCmdCounters::cmd`, `CF_HkPacket`  
`Payload::counters`, `CF_Engine::enabled`, `CF_AppData_t::engine`, `CF_HkCmdCounters::err`, `CF_AppData_t::hk`, and  
`CF_HkPacket::Payload`.

Here is the call graph for this function:



**12.51.4.19 CF\_FindTransactionBySequenceNumberAllChannels()** `CF_Transaction_t* CF_FindTransactionBySequenceNumberAllChannels ( CF_TransactionSeq_t ts,  
CF_EntityId_t eid )`

Search for a transaction across all channels.

**Assumptions, External Events, and Notes:**

None

#### Parameters

<code>ts</code>	Transaction sequence number to find
<code>eid</code>	Entity ID of the transaction

#### Returns

Pointer to the transaction, if found

#### Return values

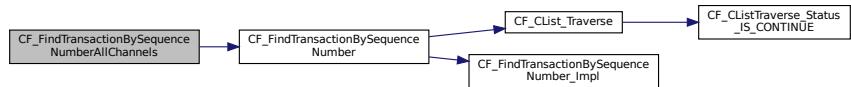
<code>NULL</code>	if transaction not found
-------------------	--------------------------

Definition at line 319 of file cf\_cmd.c.

References `CF_AppData`, `CF_FindTransactionBySequenceNumber()`, `CF_NUM_CHANNELS`, `CF_Engine::channels`, and `CF_AppData_t::engine`.

Referenced by `CF_TsnChanAction()`.

Here is the call graph for this function:



#### 12.51.4.20 CF\_FreezeCmd() CFE\_Status\_t CF\_FreezeCmd (

const CF\_FreezeCmd\_t \* msg )

Freeze a channel.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

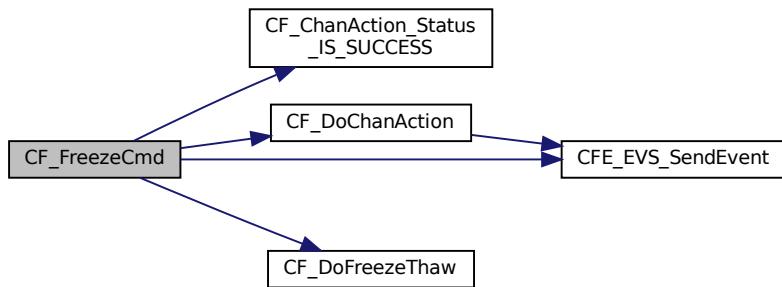
**Parameters**

msg	Pointer to command message
-----	----------------------------

Definition at line 271 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_FREEZE\_ERR\_EID, CF\_CMD\_FREEZE\_INF\_EID, CF\_DoChanAction(), CF\_DoFreezeThaw(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket::Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_FreezeCmd::Payload.

Here is the call graph for this function:



#### 12.51.4.21 CF\_GetParamCmd() CFE\_Status\_t CF\_GetParamCmd (

const CF\_GetParamCmd\_t \* msg )

Ground command to set a configuration parameter.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

**Parameters**

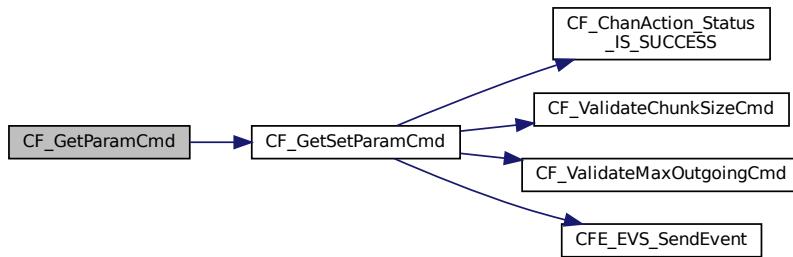
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1136 of file cf\_cmd.c.

References CF\_SetParamCmd(), CFE\_SUCCESS, CF\_GetParam\_Payload::chan\_num, CF\_GetParam\_Payload::key, and CF\_SetParamCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.22 CF\_SetParamCmd()** void CF\_SetParamCmd (

bool <i>is_set</i> ,	
CF_SetParam_ValueID_t <i>param_id</i> ,	
uint32 <i>value</i> ,	
uint8 <i>chan_num</i> )	

Perform a configuration get/set operation.

For a set, this sets the value within the CF configuration For a get, this generates an EVS event with the requested information

**Description**

Combine get and set in one function with common logic.

**Assumptions, External Events, and Notes:**

None

**Parameters**

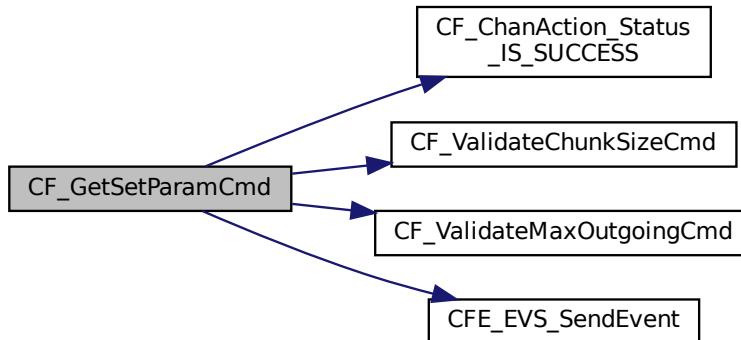
<i>is_set</i>	Whether to get (false) or set (true)
<i>param_id</i>	Parameter ID
<i>value</i>	Value to get/set
<i>chan_num</i>	Channel number to operate on

Definition at line 966 of file cf\_cmd.c.

References CF\_ChannelConfig::ack\_limit, CF\_ChannelConfig::ack\_timer\_s, CF\_AppData, CF\_ChанAction\_Status\_IS\_SUCCESS(), CF\_CMD\_GETSET1\_INF\_EID, CF\_CMD\_GETSET2\_INF\_EID, CF\_CMD\_GETSET\_CHAN\_ERR\_EID, CF\_CMD\_GETSET\_PARAM\_ERR\_EID, CF\_CMD\_GETSET\_VALIDATE\_ERR\_EID, CF\_ERROR, CF\_GetSet\_ValueID\_ack\_limit, CF\_GetSet\_ValueID\_ack\_timer\_s, CF\_GetSet\_ValueID\_chan\_max\_outgoing\_messages\_per\_wakeup, CF\_GetSet\_ValueID\_inactivity\_timer\_s, CF\_GetSet\_ValueID\_local\_eid, CF\_GetSet\_ValueID\_nak\_limit, CF\_GetSet\_ValueID\_nak\_timer\_s, CF\_GetSet\_ValueID\_outgoing\_file\_chunk\_size, CF\_GetSet\_ValueID\_rx\_crc\_calc\_bytes\_per\_wakeup, CF\_GetSet\_ValueID\_ticks\_per\_second, CF\_NUM\_CHANNELS, CF\_ValidateChunkSizeCmd(), CF\_ValidateMaxOutgoingCmd(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_ConfigTable::chan, CF\_HkCmdCounters::cmd, CF\_AppData\_t::config\_table, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_ChannelConfig::inactivity\_timer\_s, CF\_ConfigTable::local\_eid, CF\_ChannelConfig::max\_outgoing\_messages\_per\_wakeup, CF\_ChannelConfig::nak\_limit, CF\_ChannelConfig::nak\_timer\_s, CF\_ConfigTable::outgoing\_file\_chunk\_size, CF\_HkPacket::Payload, CF\_ConfigTable::rx\_crc\_calc\_bytes\_per\_wakeup, and CF\_ConfigTable::ticks\_per\_second.

Referenced by CF\_GetParamCmd(), and CF\_SetParamCmd().

Here is the call graph for this function:



#### 12.51.4.23 CF\_NoopCmd() [CFE\\_Status\\_t](#) CF\_NoopCmd (

```
const CF_NoopCmd_t * msg )
```

The no-operation command.

##### Description

This function has a signature the same of all cmd\_ functions. This function simply prints an event message. Increments the command accept counter. The msg parameter is ignored in this one.

##### Assumptions, External Events, and Notes:

None

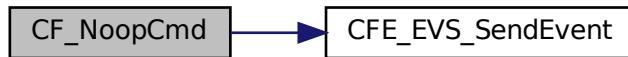
##### Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 48 of file cf\_cmd.c.

References CF\_AppData, CF\_MAJOR\_VERSION, CF\_MINOR\_VERSION, CF\_MISSION\_REV, CF\_NOOP\_INF\_EID, CF\_REVISION, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_AppData\_t::hk, and CF\_HkPacket::Payload.

Here is the call graph for this function:



**12.51.4.24 CF\_PlaybackDirCmd()** `CF_Status_t CF_PlaybackDirCmd ( const CF_PlaybackDirCmd_t * msg )`

Ground command to start directory playback.

#### Description

This function has a signature the same of all cmd\_ functions. Increments the command accept or reject counter.

#### Assumptions, External Events, and Notes:

msg must not be NULL.

#### Parameters

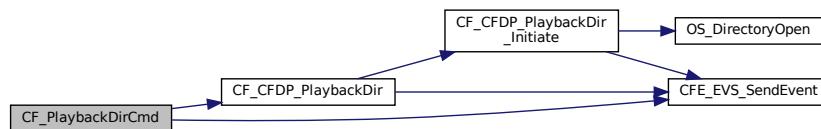
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 177 of file cf\_cmd.c.

References CF\_AppData, CF\_CFDP\_CLASS\_1, CF\_CFDP\_CLASS\_2, CF\_CFDP\_PlaybackDir(), CF\_CMD\_BAD\_PARAM\_ERR\_EID, CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID, CF\_CMD\_PLAYBACK\_DIR\_INF\_EID, CF\_NUM\_CHANNELS, CF\_TxFile\_Payload::cfdp\_class, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_TxFile\_Payload::chan\_num, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_TxFile\_Payload::dest\_id, CF\_TxFile\_Payload::dst\_filename, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_TxFile\_Payload::keep, CF\_HkPacket::Payload, CF\_PlaybackDirCmd::Payload, CF\_TxFile\_Payload::priority, and CF\_TxFile\_Payload::src\_filename.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.25 CF\_PurgeHistory()** *CF\_CListTraverse\_Status\_t* CF\_PurgeHistory (

```
    CF_CListNode_t * node,
    void * arg )
```

Purge the history queue for the given channel.

This helper function is used in conjunction with [CF\\_CList\\_Traverse\(\)](#)

**Assumptions, External Events, and Notes:**

node must not be NULL. chan must not be NULL.

**Parameters**

<i>node</i>	Current list node being traversed
<i>arg</i>	Channel pointer passed through as opaque object, must be <i>CF_Channel_t*</i>

**Returns**

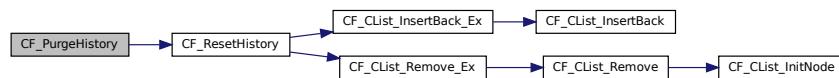
Always [CF\\_CLIST\\_CONT](#) to process all entries

Definition at line 700 of file cf\_cmd.c.

References [CF\\_CLIST\\_CONT](#), [CF\\_ResetHistory\(\)](#), and [container\\_of](#).

Referenced by [CF\\_DoPurgeQueue\(\)](#).

Here is the call graph for this function:

**12.51.4.26 CF\_PurgeQueueCmd()** *CFE\_Status\_t* CF\_PurgeQueueCmd (

```
    const CF_PurgeQueueCmd_t * msg )
```

Ground command to purge either the history or pending queues.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

**Parameters**

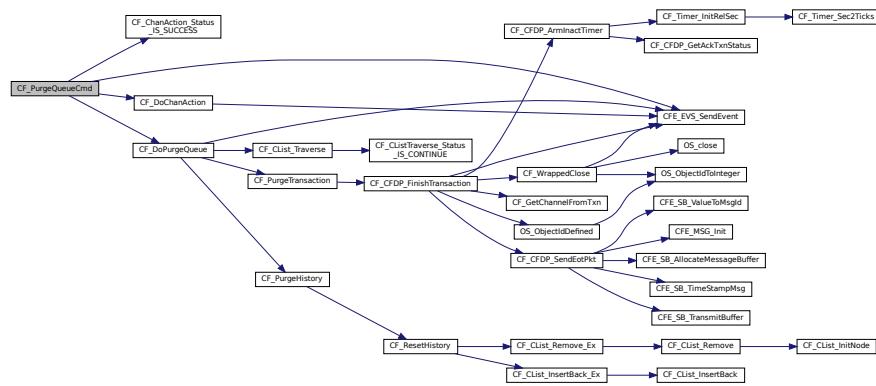
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 778 of file cf\_cmd.c.

References [CF\\_AppData](#), [CF\\_ChAction\\_Status\\_IS\\_SUCCESS\(\)](#), [CF\\_CMD\\_PURGE\\_QUEUE\\_ERR\\_EID](#), [CF\\_CMD\\_PURGE\\_QUEUE\\_INF\\_EID](#), [CF\\_DoChanAction\(\)](#), [CF\\_DoPurgeQueue\(\)](#), [CFE\\_EVS\\_EventType\\_ERROR](#), [CFE\\_EVS\\_EventType\\_INFORMATION](#), [CFE\\_EVS\\_SendEvent\(\)](#), [CFE\\_SUCCESS](#), [CF\\_HkCmdCounters::cmd](#), [CF\\_HkPacket::Payload](#), and [CF\\_PurgeQueueCmd::Payload](#).

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.27 CF\_PurgeTransaction()** [CF\\_CListTraverse\\_Status\\_t](#) CF\_PurgeTransaction (   
`CF_CListNode_t * node,`  
`void * ignored )`

Purge the pending transaction queue.

This helper function is used in conjunction with [CF\\_CList\\_Traverse\(\)](#)

**Assumptions, External Events, and Notes:**

node must not be NULL.

#### Parameters

<i>node</i>	Current list node being traversed
<i>ignored</i>	Not used by this implementation

**Returns**

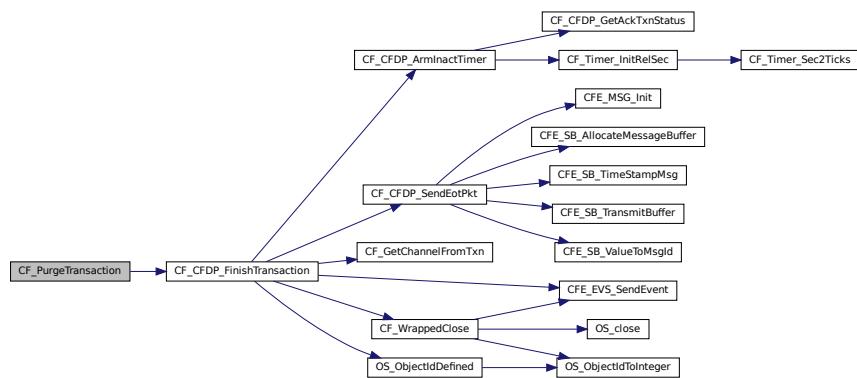
Always [CF\\_CLIST\\_CONT](#) to process all entries

Definition at line 714 of file cf\_cmd.c.

References CF\_CFDP\_FinishTransaction(), CF\_CLIST\_CONT, and container\_of.

Referenced by CF\_DoPurgeQueue().

Here is the call graph for this function:



**12.51.4.28 CF\_ResetCountersCmd()** [CFE\\_Status\\_t](#) CF\_ResetCountersCmd (   
     const [CF\\_ResetCountersCmd\\_t](#) \* msg )

The reset counters command.

**Description**

This function has a signature the same of all cmd\_ functions. Resets the given counter, or all. Increments the command accept or reject counter. If the command counters are reset, then there is no increment.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

**Parameters**

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 64 of file cf\_cmd.c.

References CF\_UnionArgs\_Payload::byte, CF\_AppData, CF\_CMD\_RESET\_INVALID\_ERR\_EID, CF\_NUM\_CHANNELS, CF\_Reset\_all, CF\_Reset\_command, CF\_Reset\_down, CF\_Reset\_fault, CF\_RESET\_INF\_EID, CF\_Reset\_up, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkPacket\_Payload::channel\_hk, CF\_HkCmdCounters::cmd, CF\_HkChannel\_Data::counters, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_HkCounters::fault, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_ResetCountersCmd::Payload, CF\_HkCounters::recv, and CF\_HkCounters::sent.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.29 CF\_ResumeCmd()** `CFE_Status_t CF_ResumeCmd ( const CF_ResumeCmd_t * msg )`

Handle transaction resume command.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

#### Parameters

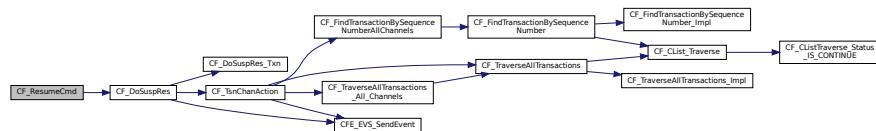
<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 467 of file cf\_cmd.c.

References CF\_DoSuspRes(), CFE\_SUCCESS, and CF\_ResumeCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.30 CF\_SendHkCmd()** `CFE_Status_t CF_SendHkCmd ( const CF_SendHkCmd_t * msg )`

Send CF housekeeping packet.

#### Description

The command to send the CF housekeeping packet

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>msg</i>	Pointer to command message
------------	----------------------------

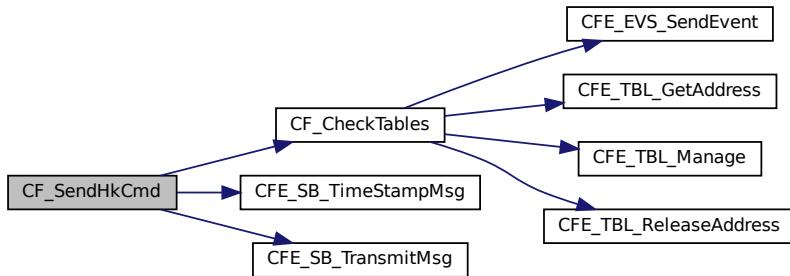
Definition at line 1207 of file cf\_cmd.c.

References CFE\_AppData, CFE\_CheckTables(), CFE\_SB\_TimeStampMsg(), CFE\_SB\_TransmitMsg(), CFE\_SUCCESS,

CF\_AppData\_t::hk, and CF\_HkPacket::TelemetryHeader.

Referenced by CF\_AppPipe().

Here is the call graph for this function:



**12.51.4.31 CF\_SetParamCmd()** `CFE_Status_t CF_SetParamCmd ( const CFE_SetParamCmd_t * msg )`

Ground command to set a configuration parameter.

**Assumptions, External Events, and Notes:**

`msg` must not be NULL.

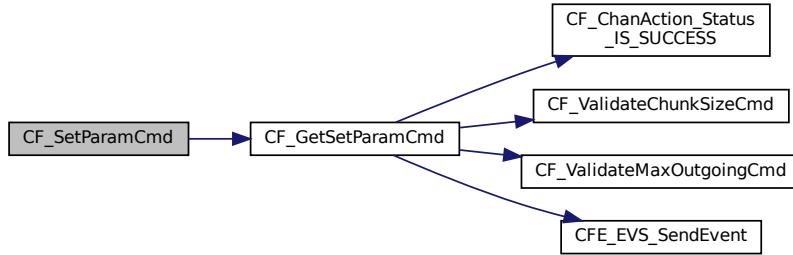
**Parameters**

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1121 of file cf\_cmd.c.

References CFE\_GetSetParamCmd(), CFE\_SUCCESS, CF\_SetParam\_Payload::chan\_num, CF\_SetParam\_Payload::key, CF\_SetParamCmd::Payload, and CF\_SetParam\_Payload::value.

Here is the call graph for this function:



**12.51.4.32 CF\_SuspendCmd()** `CFE_Status_t CF_SuspendCmd ( const CF_SuspendCmd_t * msg )`

Handle transaction suspend command.

Assumptions, External Events, and Notes:

msg must not be NULL.

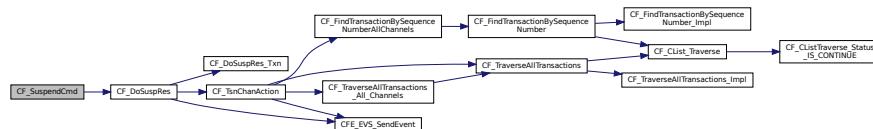
#### Parameters

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 455 of file cf\_cmd.c.

References CF\_DoSuspRes(), CFE\_SUCCESS, and CF\_SuspendCmd::Payload.

Here is the call graph for this function:



**12.51.4.33 CF\_ThawCmd()** `CFE_Status_t CF_ThawCmd ( const CF_ThawCmd_t * msg )`

Thaw a channel.

Assumptions, External Events, and Notes:

msg must not be NULL.

**Parameters**

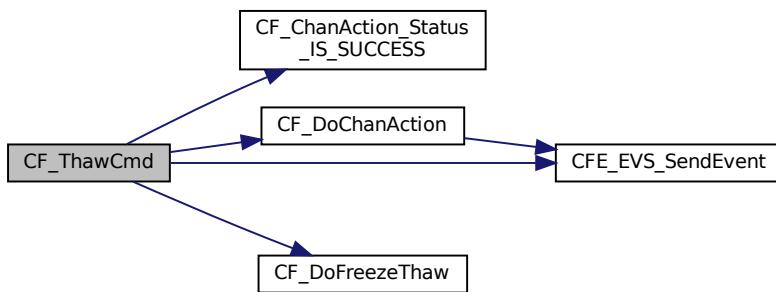
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 295 of file cf\_cmd.c.

References CF\_AppData, CF\_ChanAction\_Status\_IS\_SUCCESS(), CF\_CMD\_THAW\_ERR\_EID, CF\_CMD\_THAW\_INF\_EID, CF\_DoChanAction(), CF\_DoFreezeThaw(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_HkPacket::Payload, and CF\_ThawCmd::Payload.

Referenced by CF\_ProcessGroundCommand().

Here is the call graph for this function:



**12.51.4.34 CF\_TsnChanAction()** `int32 CF_TsnChanAction (`  
 `const CF_Transaction_Payload_t * data,`  
 `const char * cmdstr,`  
 `CF_TsnChanAction_fn_t fn,`  
 `void * context )`

Common logic for all transaction sequence number and channel commands.

**Description**

All the commands that on a transaction on a particular channel come through this function. This puts all common logic in one place. It does handle the command accept or reject counters.

**Assumptions, External Events, and Notes:**

cmd must not be NULL, fn must be a valid function, context may be NULL.

**Parameters**

<i>data</i>	Pointer to payload being processed
<i>cmdstr</i>	String to include in any generated EVS events
<i>fn</i>	Callback function to invoke for each matched transaction
<i>context</i>	Opaque object to pass through to the callback

**Returns**

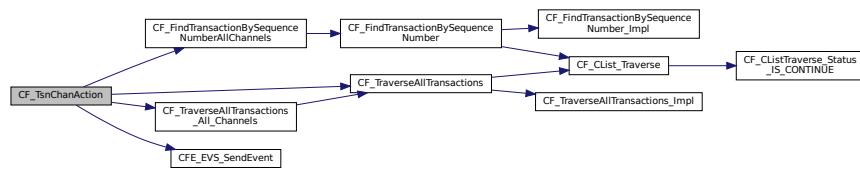
returns the number of transactions acted upon

Definition at line 347 of file cf\_cmd.c.

References CF\_ALL\_CHANNELS, CF\_AppData, CF\_CMD\_TRANS\_NOT\_FOUND\_ERR\_EID, CF\_CMD\_TSN\_←  
CHAN\_INVALID\_ERR\_EID, CF\_COMPOUND\_KEY, CF\_FindTransactionBySequenceNumberAllChannels(), CF←  
NUM\_CHANNELS, CF\_TraverseAllTransactions(), CF\_TraverseAllTransactions\_All\_Channels(), CFE\_EVS\_Event←  
Type\_ERROR, CFE\_EVS\_SendEvent(), CF\_Transaction\_Payload::chan, CF\_Engine::channels, CF\_Transaction←  
Payload::eid, CF\_AppData\_t::engine, and CF\_Transaction\_Payload::ts.

Referenced by CF\_AbandonCmd(), CF\_CancelCmd(), and CF\_DoSuspRes().

Here is the call graph for this function:

**12.51.4.35 CF\_TxFileCmd()** `CFE_Status_t CF_TxFileCmd(`

`const CF_TxFileCmd_t * msg )`

Ground command to start a file transfer.

**Description**

This function has a signature the same of all cmd\_ functions. Increments the command accept or reject counter.

**Assumptions, External Events, and Notes:**

msg must not be NULL.

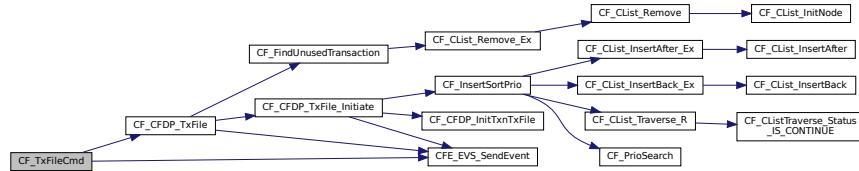
**Parameters**

<code>msg</code>	Pointer to command message
------------------	----------------------------

Definition at line 134 of file cf\_cmd.c.

References CF\_AppData, CF\_CFDP\_CLASS\_1, CF\_CFDP\_CLASS\_2, CF\_CFDP\_TxFile(), CF\_CMD\_BAD\_←  
PARAM\_ERR\_EID, CF\_CMD\_TX\_FILE\_ERR\_EID, CF\_CMD\_TX\_FILE\_INF\_EID, CF\_NUM\_CHANNELS, CF←  
TxFile\_Payload::cfdp\_class, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS←  
SendEvent(), CFE\_SUCCESS, CF\_TxFile\_Payload::chan\_num, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload←  
::counters, CF\_TxFile\_Payload::dest\_id, CF\_TxFile\_Payload::dst\_filename, CF\_HkCmdCounters::err, CF\_AppData←  
\_t::hk, CF\_TxFile\_Payload::keep, CF\_HkPacket::Payload, CF\_TxFileCmd::Payload, CF\_TxFile\_Payload::priority, and  
CF\_TxFile\_Payload::src\_filename.

Here is the call graph for this function:



**12.51.4.36 CF\_ValidateChunkSizeCmd()** `CF_ChanAction_Status_t CF_ValidateChunkSizeCmd ( CF_ChunkSize_t val, uint8 chan_num )`

Checks if the value is less than or equal to the max PDU size.

**Assumptions, External Events, and Notes:**

None

#### Parameters

<code>val</code>	Size of chunk to test
<code>chan_num</code>	Ignored by this implementation

#### Returns

status code indicating if check passed

#### Return values

<code>CF_ChanAction_Status_SUCCESS</code>	if successful (val is less than or equal to max PDU)
<code>CF_ChanAction_Status_ERROR</code>	if failed (val is greater than max PDU)

Definition at line 931 of file cf\_cmd.c.

References `CF_ChanAction_Status_ERROR`, and `CF_ChanAction_Status_SUCCESS`.

Referenced by `CF_GetSetParamCmd()`.

**12.51.4.37 CF\_ValidateMaxOutgoingCmd()** `CF_ChanAction_Status_t CF_ValidateMaxOutgoingCmd ( uint32 val, uint8 chan_num )`

Checks if the value is within allowable range as outgoing packets per wakeup.

**Assumptions, External Events, and Notes:**

None

**Parameters**

<i>val</i>	Number to test
<i>chan_num</i>	CF channel number

**Returns**

status code indicating if check passed

**Return values**

<i>CF_ChanAction_Status_SUCCESS</i>	if successful (val is allowable as max packets per wakeup)
<i>CF_ChanAction_Status_ERROR</i>	if failed (val is not allowed)

Definition at line 947 of file cf\_cmd.c.

References CF\_AppData, CF\_ChанAction\_Status\_ERROR, CF\_ChанAction\_Status\_SUCCESS, CF\_ConfigTable<::chan, CF\_AppData\_t::config\_table, and CF\_ChannelConfig::sem\_name.

Referenced by CF\_GetSetParamCmd().

**12.51.4.38 CF\_WakeupCmd()** *CFE\_Status\_t* *CF\_WakeupCmd* (

```
const CF_WakeupCmd_t * msg )
```

CF wakeup function.

**Description**

Performs a single engine cycle for each wakeup

**Assumptions, External Events, and Notes:**

None

**Parameters**

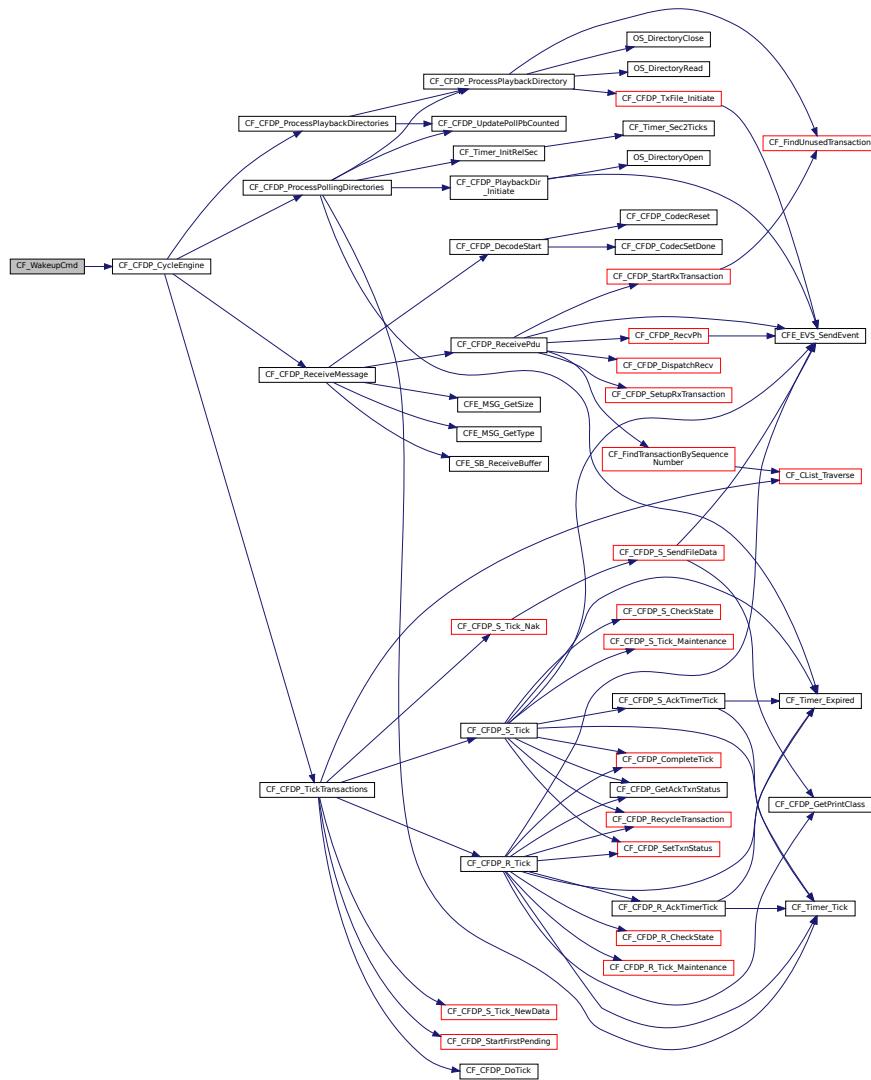
<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 1224 of file cf\_cmd.c.

References CF\_CFDP\_CycleEngine(), CF\_PERF\_ID\_CYCLE\_ENG, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and CFE\_SUCCESS.

Referenced by CF\_AppPipe().

Here is the call graph for this function:



**12.51.4.39 CF\_WriteQueueCmd()** CFE\_Status\_t CF\_WriteQueueCmd (

```
const CF_WriteQueueCmd_t * msg )
```

Ground command to write a file with queue information.

### **Assumptions, External Events, and Notes:**

msg must not be NULL.

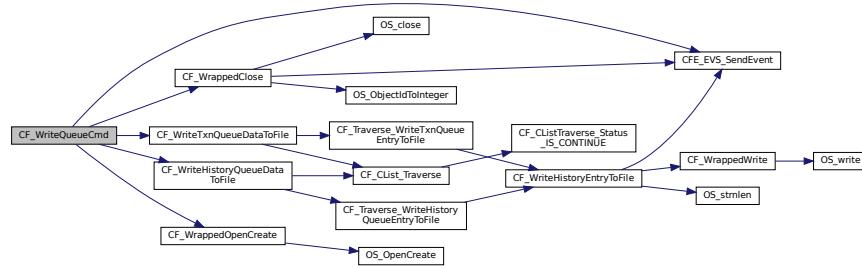
## Parameters

<i>msg</i>	Pointer to command message
------------	----------------------------

Definition at line 801 of file cf\_cmd.c.

References CF\_AppData, CF\_CMD\_WQ\_ARGS\_ERR\_EID, CF\_CMD\_WQ\_CHAN\_ERR\_EID, CF\_CMD\_WQ\_INF\_EID, CF\_CMD\_WQ\_OPEN\_ERR\_EID, CF\_CMD\_WQ\_WRITEHIST\_RX\_ERR\_EID, CF\_CMD\_WQ\_WRITEHIST\_TX\_ERR\_EID, CF\_CMD\_WQ\_WRITEQ\_PEND\_ERR\_EID, CF\_CMD\_WQ\_WRITEQ\_RX\_ERR\_EID, CF\_CMD\_WQ\_WRITEQ\_TX\_ERR\_EID, CF\_Direction\_RX, CF\_Direction\_TX, CF\_NUM\_CHANNELS, CF\_Queue\_active, CF\_Queue\_all, CF\_Queue\_history, CF\_Queue\_pend, CF\_QueueIdx\_PEND, CF\_QueueIdx\_RX, CF\_QueueIdx\_TX, CF\_Type\_all, CF\_Type\_down, CF\_Type\_up, CF\_WrappedClose(), CF\_WrappedOpenCreate(), CF\_WriteHistoryQueueDataToFile(), CF\_WriteTxnQueueDataToFile(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_EventType\_INFORMATION, CFE\_EVS\_SendEvent(), CFE\_SUCCESS, CF\_WriteQueue\_Payload::chan, CF\_Engine::channels, CF\_HkCmdCounters::cmd, CF\_HkPacket\_Payload::counters, CF\_AppData\_t::engine, CF\_HkCmdCounters::err, CF\_WriteQueue\_Payload::filename, CF\_AppData\_t::hk, OS\_FILE\_FLAG\_CREATE, OS\_FILE\_FLAG\_TRUNCATE, OS\_OBJECT\_ID\_UNDEFINED, OS\_WRITE\_ONLY, CF\_HkPacket::Payload, CF\_WriteQueueCmd::Payload, CF\_WriteQueue\_Payload::queue, and CF\_WriteQueue\_Payload::type.

Here is the call graph for this function:



## 12.52 apps/cf/fsw/src/cf\_codec.c File Reference

```
#include "cf_app.h"
#include "cf_cfdp_pdu.h"
#include "cf_codec.h"
#include "cf_eventids.h"
#include <stdint.h>
```

### Data Structures

- struct [CF\\_Codec\\_BitField](#)

### Macros

- #define [CF\\_INIT\\_FIELD](#)(NBITS, SHIFT)
- #define [FGV](#)(SRC, NAME) ([CF\\_FieldGetVal](#)((SRC).octets, (NAME).shift, (NAME).mask))
- #define [FSV](#)(DEST, NAME, VAL) ([CF\\_FieldSetVal](#)((DEST).octets, (NAME).shift, (NAME).mask, VAL))

### TypeDefs

- typedef struct [CF\\_Codec\\_BitField](#) [CF\\_Codec\\_BitField\\_t](#)

### Functions

- static [uint8 CF\\_FieldGetVal](#) (const [uint8](#) \*src, [uint8](#) shift, [uint8](#) mask)
- static void [CF\\_FieldSetVal](#) ([uint8](#) \*dest, [uint8](#) shift, [uint8](#) mask, [uint8](#) val)

- static void `CF_Codec_Store_uint8` (`CF_CFDP_uint8_t` \*`pdst`, `uint8` `val`)
- static void `CF_Codec_Store_uint16` (`CF_CFDP_uint16_t` \*`pdst`, `uint16` `val`)
- static void `CF_Codec_Store_uint32` (`CF_CFDP_uint32_t` \*`pdst`, `uint32` `val`)
- static void `CF_Codec_Store_uint64` (`CF_CFDP_uint64_t` \*`pdst`, `uint64` `val`)
- static void `CF_Codec_Load_uint8` (`uint8` \*`pdst`, const `CF_CFDP_uint8_t` \*`psrc`)
- static void `CF_Codec_Load_uint16` (`uint16` \*`pdst`, const `CF_CFDP_uint16_t` \*`psrc`)
- static void `CF_Codec_Load_uint32` (`uint32` \*`pdst`, const `CF_CFDP_uint32_t` \*`psrc`)
- static void `CF_Codec_Load_uint64` (`uint64` \*`pdst`, const `CF_CFDP_uint64_t` \*`psrc`)
- bool `CF_CFDP_CodecCheckSize` (`CF_CodecState_t` \*`state`, `size_t` `chunkszie`)  
*Advances the position by the indicated size, confirming the block will fit into the PDU.*
- void \* `CF_CFDP_DoEncodeChunk` (`CF_EncoderState_t` \*`state`, `size_t` `chunkszie`)  
*Encode a block of data into the PDU.*
- const void \* `CF_CFDP_DoDecodeChunk` (`CF_DecoderState_t` \*`state`, `size_t` `chunkszie`)  
*Decode a block of data from the PDU.*
- `uint8 CF_CFDP_GetValueEncodedSize (uint64 Value)`  
*Gets the minimum number of octets that the given integer may be encoded in.*
- void `CF_EncodeIntegerInSize (CF_EncoderState_t` \*`state`, `uint64 value`, `uint8 encode_size`)  
*Encodes the given integer value in the given number of octets.*
- void `CF_CFDP_EncodeHeaderWithoutSize (CF_EncoderState_t` \*`state`, `CF_Logical_PduHeader_t` \*`plh`)  
*Encodes a CFDP PDU base header block, bypassing the size field.*
- void `CF_CFDP_EncodeHeaderFinalSize (CF_EncoderState_t` \*`state`, `CF_Logical_PduHeader_t` \*`plh`)  
*Updates an already-encoded PDU base header block with the final PDU size.*
- void `CF_CFDP_EncodeFileDirectiveHeader (CF_EncoderState_t` \*`state`, `CF_Logical_PduFileDirectiveHeader_t` \*`pfdhir`)  
*Encodes a CFDP file directive header block.*
- void `CF_CFDP_EncodeLV (CF_EncoderState_t` \*`state`, `CF_Logical_Lv_t` \*`pllv`)  
*Encodes a single CFDP Length+Value (LV) pair.*
- void `CF_CFDP_EncodeTLV (CF_EncoderState_t` \*`state`, `CF_Logical_Tlv_t` \*`pltlv`)  
*Encodes a single CFDP Type+Length+Value (TLV) tuple.*
- void `CF_CFDP_EncodeSegmentRequest (CF_EncoderState_t` \*`state`, `CF_Logical_SegmentRequest_t` \*`plsseg`)  
*Encodes a single CFDP Segment Request block.*
- void `CF_CFDP_EncodeAllTlv (CF_EncoderState_t` \*`state`, `CF_Logical_TlvList_t` \*`pltlv`)  
*Encodes a list of CFDP Type+Length+Value tuples.*
- void `CF_CFDP_EncodeAllSegments (CF_EncoderState_t` \*`state`, `CF_Logical_SegmentList_t` \*`plseg`)  
*Encodes a list of CFDP Segment Request blocks.*
- void `CF_CFDP_EncodeMd (CF_EncoderState_t` \*`state`, `CF_Logical_PduMd_t` \*`plmd`)  
*Encodes a CFDP Metadata (MD) header block.*
- void `CF_CFDP_EncodeFileDataHeader (CF_EncoderState_t` \*`state`, `bool with_meta`, `CF_Logical_PduFileDataHeader_t` \*`plfdh`)  
*Encodes a CFDP File Data (FD) header block.*
- void `CF_CFDP_EncodeEof (CF_EncoderState_t` \*`state`, `CF_Logical_PduEof_t` \*`pleof`)  
*Encodes a CFDP End-of-File (EOF) header block.*
- void `CF_CFDP_EncodeFin (CF_EncoderState_t` \*`state`, `CF_Logical_PduFin_t` \*`plfin`)  
*Encodes a CFDP Final (FIN) header block.*
- void `CF_CFDP_EncodeAck (CF_EncoderState_t` \*`state`, `CF_Logical_PduAck_t` \*`plack`)  
*Encodes a CFDP Acknowledge (ACK) header block.*
- void `CF_CFDP_EncodeNak (CF_EncoderState_t` \*`state`, `CF_Logical_PduNak_t` \*`plnak`)  
*Encodes a CFDP Non-Acknowledge (NAK) header block.*

- void `CF_CFDP_EncodeCrc (CF_EncoderState_t *state, uint32 *plcrc)`  
*Encodes a CFDP CRC/Checksum.*
- `uint64 CF_DecodeIntegerInSize (CF_DecoderState_t *state, uint8 decode_size)`  
*Decodes an integer value from the specified number of octets.*
- `CFE_Status_t CF_CFDP_DecodeHeader (CF_DecoderState_t *state, CF_Logical_PduHeader_t *plh)`  
*Decodes a CFDP base PDU header.*
- `void CF_CFDP_DecodeFileDirectiveHeader (CF_DecoderState_t *state, CF_Logical_PduFileDirectiveHeader_t *pfdir)`  
*Decodes a CFDP file directive header block.*
- `void CF_CFDP_DecodeLV (CF_DecoderState_t *state, CF_Logical_Lv_t *pllv)`  
*Decodes a single CFDP Length+Value (LV) pair.*
- `void CF_CFDP_DecodeTLV (CF_DecoderState_t *state, CF_Logical_Tlv_t *pltlv)`  
*Decodes a single CFDP Type+Length+Value (TLV) tuple.*
- `void CF_CFDP_DecodeSegmentRequest (CF_DecoderState_t *state, CF_Logical_SegmentRequest_t *plseg)`  
*Decodes a single CFDP Segment Request block.*
- `void CF_CFDP_DecodeMd (CF_DecoderState_t *state, CF_Logical_PduMd_t *plmd)`  
*Decodes a CFDP Metadata (MD) header block.*
- `void CF_CFDP_DecodeFileDataHeader (CF_DecoderState_t *state, bool with_meta, CF_Logical_PduFileDataHeader_t *plfd)`  
*Decodes a CFDP File Data (FD) header block.*
- void `CF_CFDP_DecodeCrc (CF_DecoderState_t *state, uint32 *plcrc)`  
*Decodes a CFDP CRC/Checksum.*
- `void CF_CFDP_DecodeEof (CF_DecoderState_t *state, CF_Logical_PduEof_t *pleof)`  
*Decodes a CFDP End-of-File (EOF) header block.*
- `void CF_CFDP_DecodeFin (CF_DecoderState_t *state, CF_Logical_PduFin_t *plfin)`  
*Decodes a CFDP Final (FIN) header block.*
- `void CF_CFDP_DecodeAck (CF_DecoderState_t *state, CF_Logical_PduAck_t *plack)`  
*Decodes a CFDP Acknowledge (ACK) header block.*
- `void CF_CFDP_DecodeNak (CF_DecoderState_t *state, CF_Logical_PduNak_t *plnak)`  
*Decodes a CFDP Non-Acknowledge (NAK) header block.*
- `void CF_CFDP_DecodeAllTlv (CF_DecoderState_t *state, CF_Logical_TlvList_t *pltlv, uint8 limit)`  
*Decodes a list of CFDP Type+Length+Value tuples.*
- `void CF_CFDP_DecodeAllSegments (CF_DecoderState_t *state, CF_Logical_SegmentList_t *plseg, uint8 limit)`  
*Decodes a list of CFDP Segment Request blocks.*

## Variables

- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_VERSION = CF_INIT_FIELD(3, 5)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_TYPE = CF_INIT_FIELD(1, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_DIR = CF_INIT_FIELD(1, 3)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_MODE = CF_INIT_FIELD(1, 2)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_CRC = CF_INIT_FIELD(1, 1)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_FLAGS_LARGEFILE = CF_INIT_FIELD(1, 0)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_SEGMENTATION_CONTROL = CF_INIT_FIELD(1, 7)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_LENGTHS_ENTITY = CF_INIT_FIELD(3, 4)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_SEGMENT_METADATA = CF_INIT_FIELD(1, 3)`
- static const `CF_Codec_BitField_t CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE = CF_INIT_FIELD(3, 0)`
- static const `CF_Codec_BitField_t CF_CFDP_PduEof_FLAGS_CC = CF_INIT_FIELD(4, 4)`

- static const CF\_Codec\_BitField\_t CF\_CFDP\_PduFin\_FLAGS\_CC = CF\_INIT\_FIELD(4, 4)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduFin\_FLAGS\_DELIVERY\_CODE = CF\_INIT\_FIELD(1, 2)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS = CF\_INIT\_FIELD(2, 0)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduAck\_DIR\_CODE = CF\_INIT\_FIELD(4, 4)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE = CF\_INIT\_FIELD(4, 0)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduAck\_CC = CF\_INIT\_FIELD(4, 4)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduAck\_TRANSACTION\_STATUS = CF\_INIT\_FIELD(2, 0)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduMd\_CLOSURE\_REQUESTED = CF\_INIT\_FIELD(1, 7)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduMd\_CHECKSUM\_TYPE = CF\_INIT\_FIELD(4, 0)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduFileData\_RECORD\_CONTINUATION\_STATE = CF\_INIT\_FIELD(2, 6)
  - static const CF\_Codec\_BitField\_t CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH = CF\_INIT\_FIELD(6, 0)

### **12.52.1 Detailed Description**

CFDP protocol data structure encode/decode implementation

## 12.52.2 Macro Definition Documentation

```
12.52.2.1 CF_INIT_FIELD #define CF_INIT_FIELD(
```

```
Value:
{
    .shift = (SHIFT), .mask = ((1 << NBITS) - 1) \
}
```

Definition at line 40 of file cf\_codec.c.

```
12.52.2.2 FGV #define FGV(  
    SRC,  
    NAME )  (CF_FieldGetVal((SRC).octets, (NAME).shift, (NAME).mask))
```

Definition at line 65 of file cf\_codec.c.

### 12.52.2.3 FSV #define FSV(

```
    DEST,  
    NAME,  
    VAL )  (CF_FieldSetVal((DEST).octets, (NAME).shift, (NAME).mask, VAL))
```

Definition at line 66 of file cf\_codec.c.

### 12.52.3 Typedef Documentation

**12.52.3.1 CF\_Codec\_BitField\_t** `typedef struct CF_Codec_BitField CF_Codec_BitField_t`

## 12.52.4 Function Documentation

```
12.52.4.1 CF_CFDP_CodecCheckSize() bool CF_CFDP_CodecCheckSize (
    CF_CodecState_t * state,
    size_t chunksize )
```

Advances the position by the indicated size, confirming the block will fit into the PDU.

On encode, this confirms there is enough available space to hold a block of the indicated size. On decode, this confirms that decoding the indicated number of bytes will not read beyond the end of data.

If true, then the current position/offset is advanced by the indicated number of bytes. If false, then the error flag is set, so that future calls to CF\_CFDP\_CodecIsOK will also return false.

#### Note

The error flag is sticky, meaning that if any encode/decode operation fails, all future encode/decode requests on the same state will also fail. Each encode/decode step must check the flag, and skip the operation if it is false. Reporting the error can be deferred to the final stage, and only done once.

#### Parameters

<i>state</i>	Encoder/Decoder common state
<i>chunksize</i>	Size of next block to encode/decode

#### Return values

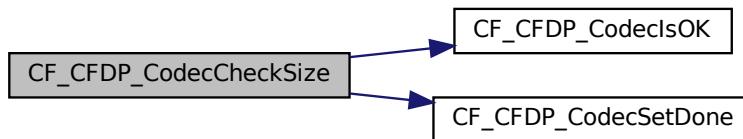
<i>true</i>	If encode/decode is possible, enough space exists
<i>false</i>	If encode/decode is not possible, not enough space or prior error occurred

Definition at line 271 of file cf\_codec.c.

References CF\_CFDP\_CodecIsOK(), CF\_CFDP\_CodecSetDone(), CF\_CodecState::max\_size, and CF\_CodecState::next\_offset.

Referenced by CF\_CFDP\_DoDecodeChunk(), and CF\_CFDP\_DoEncodeChunk().

Here is the call graph for this function:



```
12.52.4.2 CF_CFDP_DecodeAck() void CF_CFDP_DecodeAck (
    CF_DecoderState_t * state,
    CF_Logical_PduAck_t * plack )
```

Decodes a CFDP Acknowledge (ACK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure.

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

**Parameters**

<i>state</i>	Decoder state object
<i>plack</i>	Pointer to logical PDU ACK header data

Definition at line 1049 of file cf\_codec.c.

References CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::ack\_subtype\_code, CF\_Logical\_PduAck::cc, CF\_CFDP\_PduAck::cc\_and\_transaction\_status, CF\_CFDP\_PduAck\_CC, CF\_CFDP\_PduAck\_DIR\_CODE, CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE, CF\_CFDP\_PduAck\_TRANSACTION\_STATUS, CF\_DECODE\_FIXED\_CHUNK, CF\_CFDP\_PduAck::directive\_and\_subtype\_code, FGV, and CF\_Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_RecvAck().

```
12.52.4.3 CF_CFDP_DecodeAllSegments() void CF_CFDP_DecodeAllSegments (
    CF_DecoderState_t * state,
    CF_Logical_SegmentList_t * plseg,
    uint8 limit )
```

Decodes a list of CFDP Segment Request blocks.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

**Parameters**

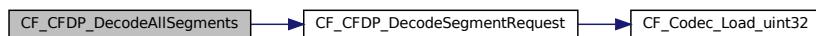
<i>state</i>	Decoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data
<i>limit</i>	Maximum number of Segment Request objects to decode

Definition at line 1125 of file cf\_codec.c.

References CF\_CFDP\_DecodeSegmentRequest(), CF\_CODEC\_GET\_REMAIN, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_PDU\_MAX\_SEGMENTS, CF\_Logical\_SegmentList::num\_segments, and CF\_Logical\_SegmentList::segments.

Referenced by CF\_CFDP\_DecodeNak().

Here is the call graph for this function:



```
12.52.4.4 CF_CFDP_DecodeAllTlv() void CF_CFDP_DecodeAllTlv (
    CF_DecoderState_t * state,
    CF_Logical_TlvList_t * pltlv,
    uint8 limit )
```

Decodes a list of CFDP Type+Length+Value tuples.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

### Parameters

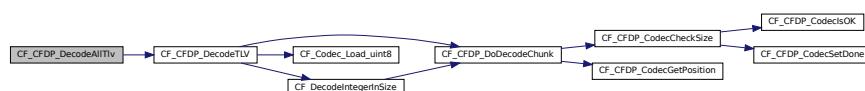
<i>state</i>	Decoder state object
<i>pltlv</i>	Pointer to logical PDU TLV header data
<i>limit</i>	Maximum number of TLV objects to decode

Definition at line 1090 of file cf\_codec.c.

References CF\_CFDP\_DecodeTLV(), CF\_CODEC\_GET\_REMAIN, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_PDU\_MAX\_TLV, CF\_Logical\_TlvList::num\_tlv, and CF\_Logical\_TlvList::tlv.

Referenced by CF\_CFDP\_DecodeEof(), and CF\_CFDP\_DecodeFin().

Here is the call graph for this function:



**12.52.4.5 CF\_CFDP\_DecodeCrc()** void CF\_CFDP\_DecodeCrc (

```

    CF_DecoderState_t * state,
    uint32 * plcrc )
  
```

Decodes a CFDP CRC/Checksum.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

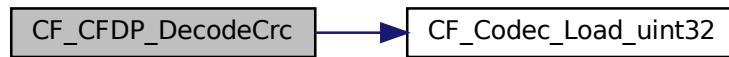
### Parameters

<i>state</i>	Decoder state object
<i>plcrc</i>	Pointer to logical CRC value

Definition at line 990 of file cf\_codec.c.

References CF\_Codec\_Load\_uint32(), and CF\_DECODE\_FIXED\_CHUNK.

Here is the call graph for this function:



**12.52.4.6 CF\_CFDP\_DecodeEof()** void CF\_CFDP\_DecodeEof (

```

    CF_DecoderState_t * state,
    CF_Logical_PduEof_t * pleof )
  
```

Decodes a CFDP End-of-File (EOF) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

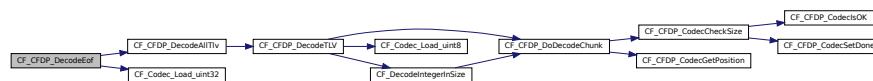
<i>state</i>	Decoder state object
<i>pleof</i>	Pointer to logical PDU EOF header data

Definition at line 1007 of file cf\_codec.c.

References CF\_CFDP\_PduEof::cc, CF\_Logical\_PduEof::cc, CF\_CFDP\_DecodeAllTlv(), CF\_CFDP\_PduEof\_FLAGS←\_CC, CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_PDU\_MAX\_TLV, CF\_CFDP\_PduEof::crc, CF\_Logical\_PduEof::crc, FGV, CF\_CFDP\_PduEof::size, CF\_Logical\_PduEof::size, and CF\_Logical\_PduEof::tlv\_list.

Referenced by CF\_CFDP\_RecvEof().

Here is the call graph for this function:



```

12.52.4.7 CF_CFDP_DecodeFileDataHeader() void CF_CFDP_DecodeFileDataHeader (
    CF_DecoderState_t * state,
    bool with_meta,
    CF_Logical_PduFileDataHeader_t * plfd )

```

Decodes a CFDP File Data (FD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

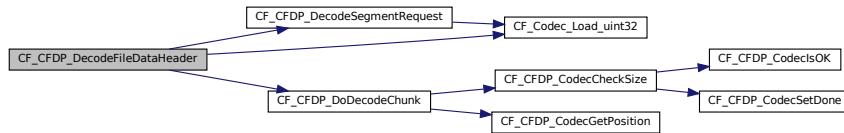
<i>state</i>	Decoder state object
<i>with_meta</i>	Whether to include optional continuation and segment request fields (always false currently)
<i>plfd</i>	Pointer to logical PDU file header data

Definition at line 928 of file cf\_codec.c.

References CF\_CFDP\_DecodeSegmentRequest(), CF\_CFDP\_DoDecodeChunk(), CF\_CFDP\_PduFileData←RECORD\_CONTINUATION\_STATE, CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH, CF\_CODEC\_GET←\_REMAIN, CF\_CODEC\_IS\_OK, CF\_Codec\_Load\_uint32(), CF\_CODEC\_SET\_DONE, CF\_DECODE\_FIXED\_CHUNK, CF\_PDU\_MAX\_SEGMENTS, CF\_Logical\_PduFileDataHeader::continuation\_state, CF\_Logical\_PduFileDataHeader::data\_len, CF\_Logical\_PduFileDataHeader::data\_ptr, FGV, CF\_Logical\_SegmentList::num\_segments, CF\_CFDP\_PduFileDataHeader::offset, CF\_Logical\_PduFileDataHeader::offset, CF\_Logical\_PduFileDataHeader::segment\_list, and CF\_Logical\_SegmentList::segments.

Referenced by CF\_CFDP\_RecvFd().

Here is the call graph for this function:



#### 12.52.4.8 CF\_CFDP\_DecodeFileDirectiveHeader()

```
void CF_CFDP_DecodeFileDirectiveHeader (
    CF_DecoderState_t * state,
    CF_Logical_PduFileDirectiveHeader_t * pfdir )
```

Decodes a CFDP file directive header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

##### Parameters

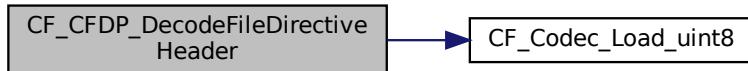
<i>state</i>	Decoder state object
<i>pfdir</i>	Pointer to logical PDU file directive header data

Definition at line 818 of file cf\_codec.c.

References CF\_Codec\_Load\_uint8(), CF\_DECODE\_FIXED\_CHUNK, CF\_CFDP\_PduFileDirectiveHeader::directive\_code, and CF\_Logical\_PduFileDirectiveHeader::directive\_code.

Referenced by CF\_CFDP\_RecvPh().

Here is the call graph for this function:



#### 12.52.4.9 CF\_CFDP\_DecodeFin()

```
void CF_CFDP_DecodeFin (
    CF_DecoderState_t * state,
    CF_Logical_PduFin_t * plfin )
```

Decodes a CFDP Final (FIN) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

### Parameters

<code>state</code>	Decoder state object
<code>plfin</code>	Pointer to logical PDU FIN header data

Definition at line 1028 of file cf\_codec.c.

References CF\_Logical\_PduFin::cc, CF\_CFDP\_DecodeAllTlv(), CF\_CFDP\_PduFin\_FLAGS\_CC, CF\_CFDP\_PduFin→\_FLAGS\_DELIVERY\_CODE, CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS, CF\_DECODE\_FIXED\_CHUNK, CF\_PDU→\_MAX\_TLV, CF\_Logical\_PduFin::delivery\_code, FGV, CF\_Logical\_PduFin::file\_status, CF\_CFDP\_PduFin::flags, and CF\_Logical\_PduFin::tlv\_list.

Referenced by CF\_CFDP\_RecvFin().

Here is the call graph for this function:



### 12.52.4.10 CF\_CFDP\_DecodeHeader() `CFE_Status_t` CF\_CFDP\_DecodeHeader (

```

    CF_DecoderState_t * state,
    CF_Logical_PduHeader_t * plh )

```

Decodes a CFDP base PDU header.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

### Note

On decode the entire base header is decoded in a single call, the size will be decoded like any other field.

### Parameters

<code>state</code>	Decoder state object
<code>plh</code>	Pointer to logical PDU base header data

### Return values

<code>CFE_SUCCESS</code>	Successful execution. Operation was performed successfully
<code>CF_ERROR</code>	on error.

Definition at line 770 of file cf\_codec.c.

References CF\_CFDP\_PduHeader\_FLAGS\_CRC, CF\_CFDP\_PduHeader\_FLAGS\_DIR, CF\_CFDP\_PduHeader→\_FLAGS\_LARGEFILE, CF\_CFDP\_PduHeader\_FLAGS\_MODE, CF\_CFDP\_PduHeader\_FLAGS\_TYPE, CF→\_CFDP\_PduHeader\_FLAGS\_VERSION, CF\_CFDP\_PduHeader\_LENGTHS\_ENTITY, CF\_CFDP\_PduHeader→\_LENGTHS\_TRANSACTION\_SEQUENCE, CF\_CFDP\_PduHeader\_SEGMENT\_METADATA, CF\_CFDP\_PduHeader→\_SEGMENTATION\_CONTROL, CF\_CODEC\_GET\_POSITION, CF\_Codec\_Load\_uint16(), CF\_DECODE\_FIXED→\_CHUNK, CF\_DecodeIntegerInSize(), CF\_ERROR, CFE\_SUCCESS, CF\_Logical\_PduHeader::crc\_flag, CF\_Logical→\_PduHeader::data\_encoded\_length, CF\_Logical\_PduHeader::destination\_eid, CF\_Logical\_PduHeader::direction,

CF\_Logical\_PduHeader::eid\_length, CF\_CFDP\_PduHeader::eid\_tsn\_lengths, FGV, CF\_CFDP\_PduHeader::flags, CF\_Logical\_PduHeader::header\_encoded\_length, CF\_Logical\_PduHeader::large\_flag, CF\_CFDP\_PduHeader::length, CF\_Logical\_PduHeader::pdu\_type, CF\_Logical\_PduHeader::segment\_meta\_flag, CF\_Logical\_PduHeader::segmentation\_control, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical\_PduHeader::source\_eid, CF\_Logical\_PduHeader::txm\_mode, CF\_Logical\_PduHeader::txn\_seq\_length, and CF\_Logical\_PduHeader::version.

Referenced by CF\_CFDP\_RecvPh().

Here is the call graph for this function:



#### 12.52.4.11 CF\_CFDP\_DecodeLV()

```
void CF_CFDP_DecodeLV (
    CF_DecoderState_t * state,
    CF_Logical_Lv_t * pllv )
```

Decodes a single CFDP Length+Value (LV) pair.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

##### Parameters

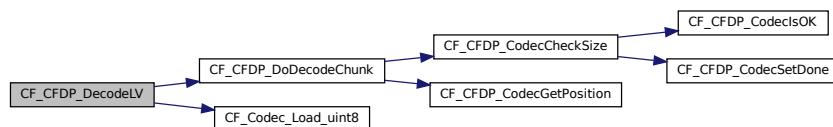
<i>state</i>	Decoder state object
<i>pllv</i>	Pointer to single logical PDU LV data

Definition at line 838 of file cf\_codec.c.

References CF\_CFDP\_DoDecodeChunk(), CF\_Codec\_Load\_uint8(), CF\_DECODE\_FIXED\_CHUNK, CF\_Logical\_Lv::data\_ptr, CF\_CFDP\_Lv::length, and CF\_Logical\_Lv::length.

Referenced by CF\_CFDP\_DecodeMd().

Here is the call graph for this function:



#### 12.52.4.12 CF\_CFDP\_DecodeMd()

```
void CF_CFDP_DecodeMd (
    CF_DecoderState_t * state,
    CF_Logical_PduMd_t * plmd )
```

Decodes a CFDP Metadata (MD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

### Parameters

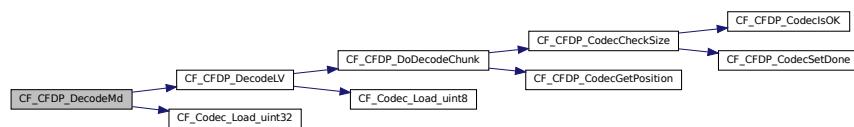
<i>state</i>	Decoder state object
<i>plmd</i>	Pointer to logical PDU metadata header data

Definition at line 905 of file cf\_codec.c.

References CF\_CFDP\_DecodeLV(), CF\_CFDP\_PduMd\_CHECKSUM\_TYPE, CF\_CFDP\_PduMd\_CLOSURE\_Requested, CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_Logical\_PduMd::checksum\_type, CF\_Logical\_PduMd::close\_req, CF\_Logical\_PduMd::dest\_filename, FGV, CF\_CFDP\_PduMd::segmentation\_control, CF\_CFDP\_PduMd::size, CF\_Logical\_PduMd::size, and CF\_Logical\_PduMd::source\_filename.

Referenced by CF\_CFDP\_RecvMd().

Here is the call graph for this function:



### 12.52.4.13 CF\_CFDP\_DecodeNak()

```
void CF_CFDP_DecodeNak (
    CF_DecoderState_t * state,
    CF_Logical_PduNak_t * plnak )
```

Decodes a CFDP Non-Acknowledge (NAK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

### Parameters

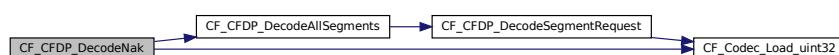
<i>state</i>	Decoder state object
<i>plnak</i>	Pointer to logical PDU NAK header data

Definition at line 1070 of file cf\_codec.c.

References CF\_CFDP\_DecodeAllSegments(), CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_PDU\_MAX\_SEGMENTS, CF\_CFDP\_PduNak::scope\_end, CF\_Logical\_PduNak::scope\_end, CF\_CFDP\_PduNak::scope\_start, CF\_Logical\_PduNak::scope\_start, and CF\_Logical\_PduNak::segment\_list.

Referenced by CF\_CFDP\_RecvNak().

Here is the call graph for this function:



### 12.52.4.14 CF\_CFDP\_DecodeSegmentRequest()

```
void CF_CFDP_DecodeSegmentRequest (
```

```
CF_DecoderState_t * state,
CF_Logical_SegmentRequest_t * plseg )
```

Decodes a single CFDP Segment Request block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

<i>state</i>	Decoder state object
<i>plseg</i>	Pointer to single logical PDU segment request header data

Definition at line 887 of file cf\_codec.c.

References CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_CFDP\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_end, CF\_CFDP\_SegmentRequest::offset\_start, and CF\_Logical\_SegmentRequest::offset\_start.

Referenced by CF\_CFDP\_DecodeAllSegments(), and CF\_CFDP\_DecodeFileDataHeader().

Here is the call graph for this function:



**12.52.4.15 CF\_CFDP\_DecodeTLV()** void CF\_CFDP\_DecodeTLV (

```
CF_DecoderState_t * state,
CF_Logical_Tlv_t * pltlv )
```

Decodes a single CFDP Type+Length+Value (TLV) tuple.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

<i>state</i>	Decoder state object
<i>pltlv</i>	Pointer to single logical PDU TLV data

Definition at line 856 of file cf\_codec.c.

References CF\_CFDP\_DoDecodeChunk(), CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, CF\_Codec\_Load\_uint8(), CF\_DECODE\_FIXED\_CHUNK, CF\_DecodeIntegerInSize(), CF\_Logical\_Tlv::data, CF\_Logical\_TlvData::data\_ptr, CF\_Logical\_TlvData::eid, CF\_CFDP\_tlv::length, CF\_Logical\_Tlv::length, CF\_CFDP\_tlv::type, and CF\_Logical\_Tlv::type. Referenced by CF\_CFDP\_DecodeAllTlv().

Here is the call graph for this function:



**12.52.4.16 CF\_CFDP\_DoDecodeChunk()** `const void* CF_CFDP_DoDecodeChunk ( CF_DecoderState_t * state, size_t chunksize )`

Decode a block of data from the PDU.

Deducts space for a block of the given size from the current PDU

#### Parameters

<code>state</code>	Decoder state object
<code>chunksize</code>	Size of block to decode

#### Returns

Pointer to block, if successful

#### Return values

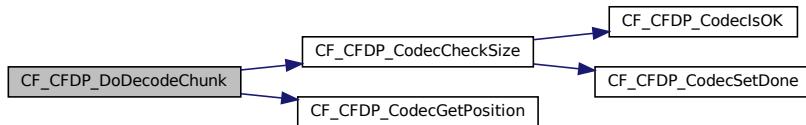
<code>NULL</code>	if not successful (no space or other error).
-------------------	--

Definition at line 311 of file cf\_codec.c.

References `CF_DecoderState::base`, `CF_CFDP_CodecCheckSize()`, `CF_CFDP_CodecGetPosition()`, and `CF_DecoderState::codec_state`.

Referenced by `CF_CFDP_DecodeFileDataHeader()`, `CF_CFDP_DecodeLV()`, `CF_CFDP_DecodeTLV()`, and `CF_DecodeIntegerInSize()`.

Here is the call graph for this function:



**12.52.4.17 CF\_CFDP\_DoEncodeChunk()** `void* CF_CFDP_DoEncodeChunk ( CF_EncoderState_t * state, size_t chunksize )`

Encode a block of data into the PDU.

Adds/Reserves space for a block of the given size in the current PDU

#### Parameters

<i>state</i>	Encoder state object
<i>chunksize</i>	Size of block to encode

#### Returns

Pointer to block, if successful

#### Return values

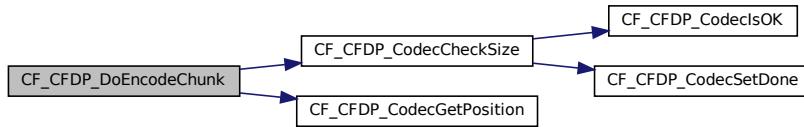
<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 293 of file cf\_codec.c.

References CF\_EncoderState::base, CF\_CFDP\_CodecCheckSize(), CF\_CFDP\_CodecGetPosition(), and CF\_EncoderState::codec\_state.

Referenced by CF\_CFDP\_EncodeLV(), CF\_CFDP\_EncodeTLV(), CF\_CFDP\_S\_SendFileData(), and CF\_EncodeIntegerInSize().

Here is the call graph for this function:



**12.52.4.18 CF\_CFDP\_EncodeAck()** void CF\_CFDP\_EncodeAck (

- CF\_EncoderState\_t \* *state*,
- CF\_Logical\_PduAck\_t \* *plack* )

Encodes a CFDP Acknowledge (ACK) header block.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

<i>state</i>	Encoder state object
<i>plack</i>	Pointer to logical PDU ACK header data

Definition at line 682 of file cf\_codec.c.

References CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::ack\_subtype\_code, CF\_Logical\_PduAck::cc, CF\_CFDP\_PduAck::cc\_and\_transaction\_status, CF\_CFDP\_PduAck\_CC, CF\_CFDP\_PduAck\_DIR\_CODE, CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE, CF\_CFDP\_PduAck\_TRANSACTION\_STATUS, CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_PduAck::directive\_and\_subtype\_code, FSV, and CF\_Logical\_PduAck::

::txn\_status.

Referenced by CF\_CFDP\_SendAck().

Here is the call graph for this function:



**12.52.4.19 CF\_CFDP\_EncodeAllSegments()** void CF\_CFDP\_EncodeAllSegments (

CF_EncoderState_t * state,
CF_Logical_SegmentList_t * plseg )

Encodes a list of CFDP Segment Request blocks.

This invokes [CF\\_CFDP\\_EncodeSegmentRequest\(\)](#) for all segments in the given list.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

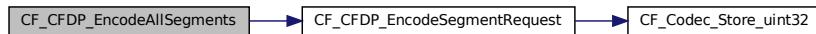
<i>state</i>	Encoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data

Definition at line 561 of file cf\_codec.c.

References [CF\\_CFDP\\_EncodeSegmentRequest\(\)](#), [CF\\_CODEC\\_IS\\_OK](#), [CF\\_Logical\\_SegmentList::num\\_segments](#), and [CF\\_Logical\\_SegmentList::segments](#).

Referenced by [CF\\_CFDP\\_EncodeFileDataHeader\(\)](#), and [CF\\_CFDP\\_EncodeNak\(\)](#).

Here is the call graph for this function:



**12.52.4.20 CF\_CFDP\_EncodeAllTlv()** void CF\_CFDP\_EncodeAllTlv (

CF_EncoderState_t * state,
CF_Logical_TlvList_t * pltlv )

Encodes a list of CFDP Type+Length+Value tuples.

This invokes [CF\\_CFDP\\_EncodeTLV\(\)](#) for all TLV values in the given list.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

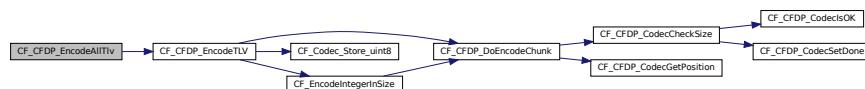
### Parameters

<i>state</i>	Encoder state object
<i>pltlv</i>	Pointer to logical PDU TLV header data

Definition at line 545 of file cf\_codec.c.

References CF\_CFDP\_EncodeTLV(), CF\_CODEC\_IS\_OK, CF\_Logical\_TlvList::num\_tlv, and CF\_Logical\_TlvList::tlv.  
Referenced by CF\_CFDP\_EncodeEof(), and CF\_CFDP\_EncodeFin().

Here is the call graph for this function:



**12.52.4.21 CF\_CFDP\_EncodeCrc()** `void CF_CFDP_EncodeCrc (`  
`CF_EncoderState_t * state,`  
`uint32 * plcrc )`

Encodes a CFDP CRC/Checksum.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

### Parameters

<i>state</i>	Encoder state object
<i>plcrc</i>	Pointer to logical CRC value

Definition at line 725 of file cf\_codec.c.

References CF\_Codec\_Store\_uint32(), and CF\_ENCODE\_FIXED\_CHUNK.

Here is the call graph for this function:



**12.52.4.22 CF\_CFDP\_EncodeEof()** `void CF_CFDP_EncodeEof (`  
`CF_EncoderState_t * state,`  
`CF_Logical_PduEof_t * pleof )`

Encodes a CFDP End-of-File (EOF) header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

**Note**

this encode includes any TLV values which are indicated in the logical data structure

**Parameters**

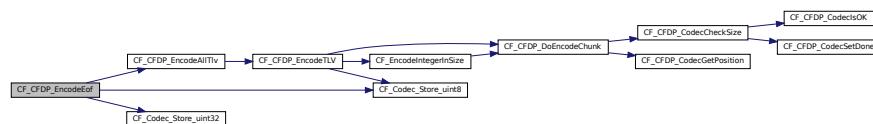
<i>state</i>	Encoder state object
<i>pleof</i>	Pointer to logical PDU EOF header data

Definition at line 638 of file cf\_codec.c.

References CF\_CFDP\_PduEof::cc, CF\_Logical\_PduEof::cc, CF\_CFDP\_EncodeAllTlv(), CF\_CFDP\_PduEof\_FLAGS←\_CC, CF\_Codec\_Store\_uint32(), CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_PduEof::crc, CF\_Logical\_PduEof::crc, FSV, CF\_CFDP\_PduEof::size, CF\_Logical\_PduEof::size, and CF\_Logical\_PduEof::tlv\_list.

Referenced by CF\_CFDP\_SendEof().

Here is the call graph for this function:

**12.52.4.23 CF\_CFDP\_EncodeFileDataHeader()** void CF\_CFDP\_EncodeFileDataHeader (

```

    CF_EncoderState_t * state,
    bool with_meta,
    CF_Logical_PduFileDataHeader_t * plfd )
  
```

Encodes a CFDP File Data (FD) header block.

This only encodes the FD header fields, specifically the data offset (required) and any metadata fields, if indicated. This does *not* encode any actual file data.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

**Parameters**

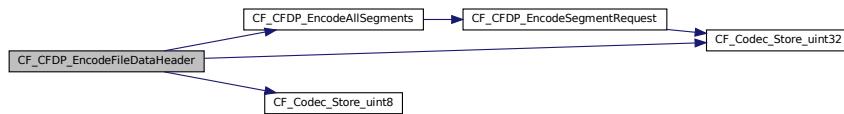
<i>state</i>	Encoder state object
<i>with_meta</i>	Whether to include optional continuation and segment request fields (always false currently)
<i>plfd</i>	Pointer to logical PDU file header data

Definition at line 601 of file cf\_codec.c.

References CF\_CFDP\_EncodeAllSegments(), CF\_CFDP\_PduFileData\_RECORD\_CONTINUATION\_STATE, CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH, CF\_Codec\_Store\_uint32(), CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_Logical\_PduFileDataHeader::continuation\_state, FSV, CF\_Logical\_SegmentList::num\_segments, CF\_CFDP\_PduFileDataHeader::offset, CF\_Logical\_PduFileDataHeader::offset, and CF\_Logical\_PduFileDataHeader::segment\_list.

Referenced by CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:



**12.52.4.24 CF\_CFDP\_EncodeFileDirectiveHeader()** `void CF_CFDP_EncodeFileDirectiveHeader ( CF_EncoderState_t * state,  
CF_Logical_PduFileDirectiveHeader_t * pfdir )`

Encodes a CFDP file directive header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

<code>state</code>	Encoder state object
<code>pfdir</code>	Pointer to logical PDU file directive header data

Definition at line 441 of file cf\_codec.c.

References CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_PduFileDirectiveHeader::directive\_code, and CF\_Logical\_PduFileDirectiveHeader::directive\_code.

Referenced by CF\_CFDP\_ConstructPduHeader().

Here is the call graph for this function:



**12.52.4.25 CF\_CFDP\_EncodeFin()** `void CF_CFDP_EncodeFin ( CF_EncoderState_t * state,  
CF_Logical_PduFin_t * plfin )`

Encodes a CFDP Final (FIN) header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Note

this encode includes any TLV values which are indicated in the logical data structure

### Parameters

<i>state</i>	Encoder state object
<i>pfin</i>	Pointer to logical PDU FIN header data

Definition at line 660 of file cf\_codec.c.

References CF\_Logical\_PduFin::cc, CF\_CFDP\_EncodeAllTlv(), CF\_CFDP\_PduFin\_FLAGS\_CC, CF\_CFDP\_PduFin←\_FLAGS\_DELIVERY\_CODE, CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS, CF\_Codec\_Store\_uint8(), CF\_ENCODE←\_FIXED\_CHUNK, CF\_Logical\_PduFin::delivery\_code, CF\_Logical\_PduFin::file\_status, CF\_CFDP\_PduFin::flags, FSV, and CF\_Logical\_PduFin::tlv\_list.

Referenced by CF\_CFDP\_SendFin().

Here is the call graph for this function:



### 12.52.4.26 CF\_CFDP\_EncodeHeaderFinalSize()

```
void CF_CFDP_EncodeHeaderFinalSize (
    CF_EncoderState_t * state,
    CF_Logical_PduHeader_t * plh )
```

Updates an already-encoded PDU base header block with the final PDU size.

This function encodes the "data\_encoded\_length" field from the logical PDU structure into the encoded header block. The PDU will also be closed (set done) to indicate that no more data should be added.

### Note

Unlike other encode operations, this function does not add any new blocks to the PDU. It only updates the already-encoded block at the beginning of the PDU, which must have been done by a prior call to [CF\\_CFDP\\_EncodeHeaderWithoutSize\(\)](#).

### See also

[CF\\_CFDP\\_EncodeHeaderWithoutSize\(\)](#) for initially encoding the PDU header block

### Parameters

<i>state</i>	Encoder state object
<i>plh</i>	Pointer to logical PDU header data

Definition at line 411 of file cf\_codec.c.

References CF\_EncoderState::base, CF\_CODEC\_GET\_POSITION, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_Codec\_Store\_uint16(), CF\_Logical\_PduHeader::data\_encoded\_length, and CF\_CFDP\_PduHeader::length.

Referenced by CF\_CFDP\_SetPduLength().

Here is the call graph for this function:



**12.52.4.27 CF\_CFDP\_EncodeHeaderWithoutSize()** `void CF_CFDP_EncodeHeaderWithoutSize (`  
 `CF_EncoderState_t * state,`  
 `CF_Logical_PduHeader_t * plh )`

Encodes a CFDP PDU base header block, bypassing the size field.

On transmit side, the common/base header must be encoded in two parts, to deal with the "total\_size" field. The initial encoding of the basic fields is done as soon as it is known that a PDU of this type needs to be sent, but the total size may not be yet known, as it depends on the remainder of encoding and any additional data that might get added to the variable length sections.

This function encodes all base header fields *except* total length. There is a separate function later to update the total\_length to the correct value once the remainder of encoding is done. Luckily, the total\_length is in the first fixed position binary blob so it is easy to update later.

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### See also

[CF\\_CFDP\\_EncodeHeaderFinalSize\(\)](#) for updating the length field once it is known

#### Parameters

<code>state</code>	Encoder state object
<code>plh</code>	Pointer to logical PDU header data

Definition at line 373 of file cf\_codec.c.

References `CF_CFDP_PduHeader_FLAGS_DIR`, `CF_CFDP_PduHeader_FLAGS_MODE`, `CF_CFDP_PduHeader_FLAGS_TYPE`, `CF_CFDP_PduHeader_FLAGS_VERSION`, `CF_CFDP_PduHeader_LENGTHS_ENTITY`, `CF_CFDP_PduHeader_LENGTHS_TRANSACTION_SEQUENCE`, `CF_CFDP_PduHeader_SEGMENT_METADATA`, `CF_CFDP_PduHeader_SEGMENTATION_CONTROL`, `CF_CODEC_GET_POSITION`, `CF_Codec_Store_uint8()`, `CF_ENCODE_FIXED_CHUNK`, `CF_EncodeIntegerInSize()`, `CF_Logical_PduHeader::destination_eid`, `CF_Logical_PduHeader::direction`, `CF_Logical_PduHeader::eid_length`, `CF_CFDP_PduHeader::eid_tsn_lengths`, `CF_CFDP_PduHeader::flags`, `FSV`, `CF_Logical_PduHeader::header_encoded_length`, `CF_Logical_PduHeader::pdu_type`, `CF_Logical_PduHeader::segment_meta_flag`, `CF_Logical_PduHeader::segmentation_control`, `CF_Logical_PduHeader::sequence_num`, `CF_Logical_PduHeader::source_eid`, `CF_Logical_PduHeader::txm_mode`, `CF_Logical_PduHeader::txn_seq_length`, and `CF_Logical_PduHeader::version`.

Referenced by `CF_CFDP_ConstructPduHeader()`.

Here is the call graph for this function:



#### 12.52.4.28 CF\_CFDP\_EncodeLV()

```
void CF_CFDP_EncodeLV (
    CF_EncoderState_t * state,
    CF_Logical_Lv_t * pllv )
```

Encodes a single CFDP Length+Value (LV) pair.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

##### Parameters

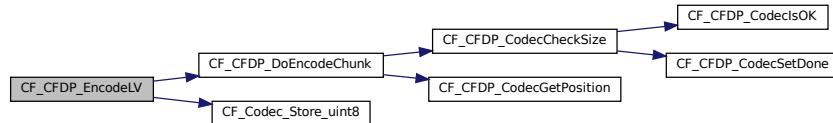
<i>state</i>	Encoder state object
<i>pllv</i>	Pointer to logical PDU LV header data

Definition at line 459 of file cf\_codec.c.

References CF\_CFDP\_DoEncodeChunk(), CF\_CODEC\_SET\_DONE, CF\_Codec\_Store\_uint8(), CF\_ENCODE\_<FIXED\_CHUNK, CF\_Logical\_Lv::data\_ptr, CF\_CFDP\_Lv::length, and CF\_Logical\_Lv::length.

Referenced by CF\_CFDP\_EncodeMd().

Here is the call graph for this function:



#### 12.52.4.29 CF\_CFDP\_EncodeMd()

```
void CF_CFDP_EncodeMd (
    CF_EncoderState_t * state,
    CF_Logical_PduMd_t * plmd )
```

Encodes a CFDP Metadata (MD) header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

##### Note

this encode includes the LV pairs for source and destination file names, which are logically part of the overall MD block.

### Parameters

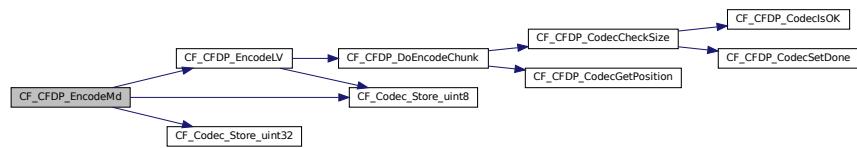
<i>state</i>	Encoder state object
<i>plmd</i>	Pointer to logical PDU metadata header data

Definition at line 577 of file cf\_codec.c.

References CF\_CFDP\_EncodeLV(), CF\_CFDP\_PduMd\_CHECKSUM\_TYPE, CF\_CFDP\_PduMd\_CLOSURE\_ REQUESTED, CF\_Codec\_Store\_uint32(), CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_Logical\_PduMd::checksum\_type, CF\_Logical\_PduMd::close\_req, CF\_Logical\_PduMd::dest\_filename, FSV, CF\_CFDP\_PduMd::segmentation\_control, CF\_CFDP\_PduMd::size, CF\_Logical\_PduMd::size, and CF\_Logical\_PduMd::source\_filename.

Referenced by CF\_CFDP\_SendMd().

Here is the call graph for this function:



**12.52.4.30 CF\_CFDP\_EncodeNak()** void CF\_CFDP\_EncodeNak (

```

    CF_EncoderState_t * state,
    CF_Logical_PduNak_t * plnak )
  
```

Encodes a CFDP Non-Acknowledge (NAK) header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

### Note

this encode includes any Segment Request values which are indicated in the logical data structure

### Parameters

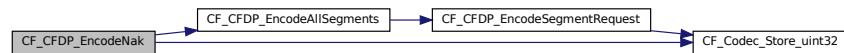
<i>state</i>	Encoder state object
<i>plnak</i>	Pointer to logical PDU NAK header data

Definition at line 705 of file cf\_codec.c.

References CF\_CFDP\_EncodeAllSegments(), CF\_Codec\_Store\_uint32(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_PduNak::scope\_end, CF\_Logical\_PduNak::scope\_end, CF\_CFDP\_PduNak::scope\_start, CF\_Logical\_PduNak::scope\_start, and CF\_Logical\_PduNak::segment\_list.

Referenced by CF\_CFDP\_SendNak().

Here is the call graph for this function:



```
12.52.4.31 CF_CFDP_EncodeSegmentRequest() void CF_CFDP_EncodeSegmentRequest (
    CF_EncoderState_t * state,
    CF_Logical_SegmentRequest_t * plseg )
```

Encodes a single CFDP Segment Request block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

<i>state</i>	Encoder state object
<i>plseg</i>	Pointer to single logical PDU segment request header data

Definition at line 527 of file cf\_codec.c.

References CF\_Codec\_Store\_uint32(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_end, CF\_CFDP\_SegmentRequest::offset\_start, and CF\_Logical\_SegmentRequest::offset\_start.

Referenced by CF\_CFDP\_EncodeAllSegments().

Here is the call graph for this function:



```
12.52.4.32 CF_CFDP_EncodeTLV() void CF_CFDP_EncodeTLV (
    CF_EncoderState_t * state,
    CF_Logical_Tlv_t * pltlv )
```

Encodes a single CFDP Type+Length+Value (TLV) tuple.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Note

Only the CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID TLV type is currently supported by this function, but other TLV types may be added in future versions as needed.

#### Parameters

<i>state</i>	Encoder state object
<i>pltlv</i>	Pointer to single logical PDU TLV header data

Definition at line 489 of file cf\_codec.c.

References CF\_CFDP\_DoEncodeChunk(), CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, CF\_CODEC\_SET\_DONE, CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_EncodeIntegerInSize(), CF\_Logical\_Tlv::data, CF\_Logical\_TlvData::data\_ptr, CF\_Logical\_TlvData::eid, CF\_CFDP\_tlv::length, CF\_Logical\_Tlv::length, CF\_CFDP\_tlv::type, and

CF\_Logical\_Tlv::type.

Referenced by CF\_CFDP\_EncodeAllTlv().

Here is the call graph for this function:



**12.52.4.33 CF\_CFDP\_GetValueEncodedSize()** `uint8 CF_CFDP_GetValueEncodedSize ( uint64 Value )`

Gets the minimum number of octets that the given integer may be encoded in.

Based on the integer value, this computes the minimum number of bytes that must be allocated to that integer within a CFDP PDU. This is typically used for entity IDs and sequence numbers, where CFDP does not specify a specific size for these items. They may be encoded between 1 and 8 bytes, depending on the actual value is.

#### Parameters

<code>Value</code>	Integer value that needs to be encoded
--------------------	--

#### Returns

Minimum number of bytes that the value requires (between 1 and 8, inclusive)

Definition at line 329 of file cf\_codec.c.

Referenced by CF\_CFDP\_AppendTlv(), and CF\_CFDP\_ConstructPduHeader().

**12.52.4.34 CF\_Codec\_Load\_uint16()** `static void CF_Codec_Load_uint16 ( uint16 * pdst, const CF_CFDP_uint16_t * psrc ) [inline], [static]`

Definition at line 206 of file cf\_codec.c.

References CF\_CFDP\_uint16\_t::octets.

Referenced by CF\_CFDP\_DecodeHeader().

**12.52.4.35 CF\_Codec\_Load\_uint32()** `static void CF_Codec_Load_uint32 ( uint32 * pdst, const CF_CFDP_uint32_t * psrc ) [inline], [static]`

Definition at line 222 of file cf\_codec.c.

References CF\_CFDP\_uint32\_t::octets.

Referenced by CF\_CFDP\_DecodeCrc(), CF\_CFDP\_DecodeEof(), CF\_CFDP\_DecodeFileDataHeader(), CF\_CFDP\_DecodeMd(), CF\_CFDP\_DecodeNak(), and CF\_CFDP\_DecodeSegmentRequest().

**12.52.4.36 CF\_Codec\_Load\_uint64()** `static void CF_Codec_Load_uint64 ( uint64 * pdst, const CF_CFDP_uint64_t * psrc ) [inline], [static]`

Definition at line 242 of file cf\_codec.c.

References CF\_CFDP\_uint64\_t::octets.

**12.52.4.37 CF\_Codec\_Load\_uint8()** static void CF\_Codec\_Load\_uint8 (

```
    uint8 * pdst,
    const CF_CFDP_uint8_t * psrc ) [inline], [static]
```

Definition at line 196 of file cf\_codec.c.

References CF\_CFDP\_uint8\_t::octets.

Referenced by CF\_CFDP\_DecodeFileDirectiveHeader(), CF\_CFDP\_DecodeLV(), and CF\_CFDP\_DecodeTLV().

**12.52.4.38 CF\_Codec\_Store\_uint16()** static void CF\_Codec\_Store\_uint16 (

```
    CF_CFDP_uint16_t * pdst,
    uint16 val ) [inline], [static]
```

Definition at line 144 of file cf\_codec.c.

References CF\_CFDP\_uint16\_t::octets.

Referenced by CF\_CFDP\_EncodeHeaderFinalSize().

**12.52.4.39 CF\_Codec\_Store\_uint32()** static void CF\_Codec\_Store\_uint32 (

```
    CF_CFDP_uint32_t * pdst,
    uint32 val ) [inline], [static]
```

Definition at line 156 of file cf\_codec.c.

References CF\_CFDP\_uint32\_t::octets.

Referenced by CF\_CFDP\_EncodeCrc(), CF\_CFDP\_EncodeEof(), CF\_CFDP\_EncodeFileDataHeader(), CF\_CFDP\_EncodeMd(), CF\_CFDP\_EncodeNak(), and CF\_CFDP\_EncodeSegmentRequest().

**12.52.4.40 CF\_Codec\_Store\_uint64()** static void CF\_Codec\_Store\_uint64 (

```
    CF_CFDP_uint64_t * pdst,
    uint64 val ) [inline], [static]
```

Definition at line 172 of file cf\_codec.c.

References CF\_CFDP\_uint64\_t::octets.

**12.52.4.41 CF\_Codec\_Store\_uint8()** static void CF\_Codec\_Store\_uint8 (

```
    CF_CFDP_uint8_t * pdst,
    uint8 val ) [inline], [static]
```

Definition at line 134 of file cf\_codec.c.

References CF\_CFDP\_uint8\_t::octets.

Referenced by CF\_CFDP\_EncodeAck(), CF\_CFDP\_EncodeEof(), CF\_CFDP\_EncodeFileDataHeader(), CF\_CFDP\_EncodeFileDirectiveHeader(), CF\_CFDP\_EncodeFin(), CF\_CFDP\_EncodeHeaderWithoutSize(), CF\_CFDP\_EncodeLV(), CF\_CFDP\_EncodeMd(), and CF\_CFDP\_EncodeTLV().

**12.52.4.42 CF\_DecodeIntegerInSize()** uint64 CF\_DecodeIntegerInSize (

```
    CF_DecoderState_t * state,
    uint8 decode_size )
```

Decodes an integer value from the specified number of octets.

This decodes an integer value in the specified number of octets. The actual number of octets must be determined using another field in the PDU before calling this function.

**Warning**

This function will decode exactly the given number of octets. If this does not match actual encoded size, the return value will be wrong, and it will likely also throw off the decoding of any fields that follow this one.

**See also**

[CF\\_EncodeIntegerInSize\(\)](#) for the inverse operation

**Parameters**

<i>state</i>	Encoder state object
<i>decode_size</i>	Number of octets that the value is encoded in (between 1 and 8, inclusive)

**Returns**

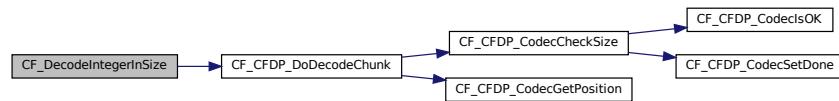
Decoded value

Definition at line 742 of file cf\_codec.c.

References CF\_CFDP\_DoDecodeChunk().

Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_DecodeTLV().

Here is the call graph for this function:

**12.52.4.43 CF\_EncodeIntegerInSize()** `void CF_EncodeIntegerInSize (`

```

    CF_EncoderState_t * state,
    uint64 value,
    uint8 encode_size )

```

Encodes the given integer value in the given number of octets.

This encodes an integer value in the specified number of octets. Use [CF\\_CFDP\\_GetValueEncodedSize\(\)](#) to determine the minimum number of octets required for a given value. Using more than the minimum is OK, but will consume extra bytes.

**Warning**

This function does not error check the encode\_size parameter, and will encode the size given, even if it results in an invalid value. Using fewer octets than the minimum reported by [CF\\_CFDP\\_GetValueEncodedSize\(\)](#) will likely result in incorrect decoding at the receiver.

**See also**

[CF\\_DecodeIntegerInSize\(\)](#) for the inverse operation

### Parameters

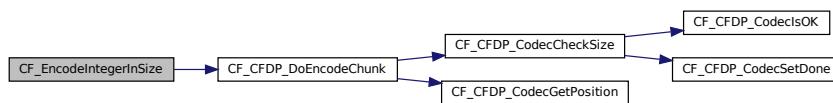
<code>state</code>	Encoder state object
<code>value</code>	Integer value that needs to be encoded
<code>encode_size</code>	Number of octets to encode the value in (between 1 and 8, inclusive)

Definition at line 348 of file cf\_codec.c.

References CF\_CFDP\_DoEncodeChunk().

Referenced by CF\_CFDP\_EncodeHeaderWithoutSize(), and CF\_CFDP\_EncodeTLV().

Here is the call graph for this function:



**12.52.4.44 CF\_FieldGetVal()** static `uint8` CF\_FieldGetVal (   
`const uint8 * src,`  
`uint8 shift,`  
`uint8 mask ) [inline], [static]`

Definition at line 48 of file cf\_codec.c.

**12.52.4.45 CF\_FieldSetVal()** static void CF\_FieldSetVal (   
`uint8 * dest,`  
`uint8 shift,`  
`uint8 mask,`  
`uint8 val ) [inline], [static]`

Definition at line 53 of file cf\_codec.c.

### 12.52.5 Variable Documentation

**12.52.5.1 CF\_CFDP\_PduAck\_CC** const `CF_Codec_BitField_t` CF\_CFDP\_PduAck\_CC = `CF_INIT_FIELD(4, 4)` [static]

Definition at line 104 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeAck(), and CF\_CFDP\_EncodeAck().

**12.52.5.2 CF\_CFDP\_PduAck\_DIR\_CODE** const `CF_Codec_BitField_t` CF\_CFDP\_PduAck\_DIR\_CODE = `CF_INIT_FIELD(4, 4)` [static]

Definition at line 102 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeAck(), and CF\_CFDP\_EncodeAck().

**12.52.5.3 CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE** const `CF_Codec_BitField_t` CF\_CFDP\_PduAck\_DIR\_Subtype\_code = `CF_INIT_FIELD(4, 0)` [static]

Definition at line 103 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeAck(), and CF\_CFDP\_EncodeAck().

**12.52.5.4 CF\_CFDP\_PduAck\_TRANSACTION\_STATUS** const `CF_Codec_BitField_t` CF\_CFDP\_PduAck\_Transaction\_Status = `CF_INIT_FIELD(2, 0)` [static]

Definition at line 105 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeAck(), and CF\_CFDP\_EncodeAck().

**12.52.5.5 CF\_CFDP\_PduEof\_FLAGS\_CC** const `CF_Codec_BitField_t` CF\_CFDP\_PduEof\_FLAGS\_CC = `CF_INIT_FIELD(4, 4)` [static]

Definition at line 89 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeEof(), and CF\_CFDP\_EncodeEof().

**12.52.5.6 CF\_CFDP\_PduFileData\_RECORD\_CONTINUATION\_STATE** const `CF_Codec_BitField_t` CF\_CFDP\_PduFileData\_Record\_Continuation\_State = `CF_INIT_FIELD(2, 6)` [static]

Definition at line 119 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), and CF\_CFDP\_EncodeFileDataHeader().

**12.52.5.7 CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH** const `CF_Codec_BitField_t` CF\_CFDP\_PduFileData\_Segment\_Metadata\_Length = `CF_INIT_FIELD(6, 0)` [static]

Definition at line 120 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), and CF\_CFDP\_EncodeFileDataHeader().

**12.52.5.8 CF\_CFDP\_PduFin\_FLAGS\_CC** const `CF_Codec_BitField_t` CF\_CFDP\_PduFin\_FLAGS\_CC = `CF_INIT_FIELD(4, 4)` [static]

Definition at line 94 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeFin(), and CF\_CFDP\_EncodeFin().

**12.52.5.9 CF\_CFDP\_PduFin\_FLAGS\_DELIVERY\_CODE** const `CF_Codec_BitField_t` CF\_CFDP\_PduFin\_FLAGS\_Delivery\_Code = `CF_INIT_FIELD(1, 2)` [static]

Definition at line 95 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeFin(), and CF\_CFDP\_EncodeFin().

**12.52.5.10 CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS** const `CF_Codec_BitField_t` CF\_CFDP\_PduFin\_FLAGS\_File\_Status = `CF_INIT_FIELD(2, 0)` [static]

Definition at line 96 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeFin(), and CF\_CFDP\_EncodeFin().

**12.52.5.11 CF\_CFDP\_PduHeader\_FLAGS\_CRC** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_FLAGS\_CRC = `CF_INIT_FIELD(1, 1)` [static]

Definition at line 75 of file cf\_codec.c.

Referenced by CF\_CFDP\_DecodeHeader().

**12.52.5.12 CF\_CFDP\_PduHeader\_FLAGS\_DIR** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_FLAGS\_DIR = `CF_INIT_FIELD(1, 3)` [static]  
Definition at line 73 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.13 CF\_CFDP\_PduHeader\_FLAGS\_LARGEFILE** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_FLAGS\_LARGEFILE = `CF_INIT_FIELD(1, 0)` [static]  
Definition at line 76 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader().

**12.52.5.14 CF\_CFDP\_PduHeader\_FLAGS\_MODE** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_FLAGS\_MODE = `CF_INIT_FIELD(1, 2)` [static]  
Definition at line 74 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.15 CF\_CFDP\_PduHeader\_FLAGS\_TYPE** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_FLAGS\_TYPE = `CF_INIT_FIELD(1, 4)` [static]  
Definition at line 72 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.16 CF\_CFDP\_PduHeader\_FLAGS\_VERSION** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_FLAGS\_VERSION = `CF_INIT_FIELD(3, 5)` [static]  
Definition at line 71 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.17 CF\_CFDP\_PduHeader\_LENGTHS\_ENTITY** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_LENGTHS\_ENTITY = `CF_INIT_FIELD(3, 4)` [static]  
Definition at line 82 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.18 CF\_CFDP\_PduHeader\_LENGTHS\_TRANSACTION\_SEQUENCE** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_LENGTHS\_TRANSACTION\_SEQUENCE = `CF_INIT_FIELD(3, 0)` [static]  
Definition at line 84 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.19 CF\_CFDP\_PduHeader\_SEGMENT\_METADATA** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_SEGMENT\_METADATA = `CF_INIT_FIELD(1, 3)` [static]  
Definition at line 83 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.20 CF\_CFDP\_PduHeader\_SEGMENTATION\_CONTROL** const `CF_Codec_BitField_t` CF\_CFDP\_PduHeader\_SEGMENTATION\_CONTROL = `CF_INIT_FIELD(1, 7)` [static]  
Definition at line 81 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeHeader(), and CF\_CFDP\_EncodeHeaderWithoutSize().

**12.52.5.21 CF\_CFDP\_PduMd\_CHECKSUM\_TYPE** const `CF_Codec_BitField_t` CF\_CFDP\_PduMd\_CHECKSUM\_TYPE = `CF_INIT_FIELD(4, 0)` [static]  
Definition at line 112 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeMd(), and CF\_CFDP\_EncodeMd().

**12.52.5.22 CF\_CFDP\_PduMd\_CLOSURE\_REQUESTED** const `CF_Codec_BitField_t` CF\_CFDP\_PduMd\_CLOSURE\_REQUESTED = `CF_INIT_FIELD(1, 7)` [static]  
Definition at line 111 of file cf\_codec.c.  
Referenced by CF\_CFDP\_DecodeMd(), and CF\_CFDP\_EncodeMd().

## 12.53 apps/cf/fsw/src(cf\_codec.h File Reference

```
#include "cfe.h"
#include "cf_cfdp_pdu.h"
#include "cf_logical_pdu.h"
```

### Data Structures

- struct `CF_CodecState`  
*Tracks the current state of an encode or decode operation.*
- struct `CF_EncoderState`  
*Current state of an encode operation.*
- struct `CF_DecoderState`  
*Current state of a decode operation.*

### Macros

- #define `CF_ENCODE_FIXED_CHUNK(state, type)` ((type \*)CF\_CFDP\_DoEncodeChunk(state, sizeof(type)))  
*Macro to encode a block of a given CFDP type into a PDU.*
- #define `CF_DECODE_FIXED_CHUNK(state, type)` ((const type \*)CF\_CFDP\_DoDecodeChunk(state, sizeof(type)))  
*Macro to decode a block of a given CFDP type into a PDU.*
- #define `CF_CODEC_IS_OK(s)` (CF\_CFDP\_CodecIsOK(&((s)->codec\_state)))  
*Macro wrapper around CF\_CFDP\_CodecIsOK()*
- #define `CF_CODEC_SET_DONE(s)` (CF\_CFDP\_CodecSetDone(&((s)->codec\_state)))  
*Macro wrapper around CF\_CFDP\_CodecSetDone()*
- #define `CF_CODEC_GET_POSITION(s)` (CF\_CFDP\_CodecGetPosition(&((s)->codec\_state)))  
*Macro wrapper around CF\_CFDP\_CodecGetPosition()*
- #define `CF_CODEC_GET_REMAIN(s)` (CF\_CFDP\_CodecGetRemain(&((s)->codec\_state)))  
*Macro wrapper around CF\_CFDP\_CodecGetRemain()*
- #define `CF_CODEC_GET_SIZE(s)` (CF\_CFDP\_CodecGetSize(&((s)->codec\_state)))  
*Macro wrapper around CF\_CFDP\_CodecGetSize()*

## Typedefs

- **typedef struct CF\_CodecState CF\_CodecState\_t**  
*Tracks the current state of an encode or decode operation.*
- **typedef struct CF\_EncoderState CF\_EncoderState\_t**  
*Current state of an encode operation.*
- **typedef struct CF\_DecoderState CF\_DecoderState\_t**  
*Current state of a decode operation.*

## Functions

- **static bool CF\_CFDP\_CodeclsOK (const CF\_CodecState\_t \*state)**  
*Checks if the codec is currently valid or not.*
- **static void CF\_CFDP\_CodecSetDone (CF\_CodecState\_t \*state)**  
*Sets a codec to the "done" state.*
- **static size\_t CF\_CFDP\_CodecGetPosition (const CF\_CodecState\_t \*state)**  
*Obtains the current position/offset within the PDU.*
- **static size\_t CF\_CFDP\_CodecGetSize (const CF\_CodecState\_t \*state)**  
*Obtains the maximum size of the PDU being encoded/decoded.*
- **static size\_t CF\_CFDP\_CodecGetRemain (const CF\_CodecState\_t \*state)**  
*Obtains the remaining size of the PDU being encoded/decoded.*
- **static void CF\_CFDP\_CodecReset (CF\_CodecState\_t \*state, size\_t max\_size)**  
*Resets a codec state.*
- **bool CF\_CFDP\_CodecCheckSize (CF\_CodecState\_t \*state, size\_t chunksize)**  
*Advances the position by the indicated size, confirming the block will fit into the PDU.*
- **void \* CF\_CFDP\_DoEncodeChunk (CF\_EncoderState\_t \*state, size\_t chunksize)**  
*Encode a block of data into the PDU.*
- **const void \* CF\_CFDP\_DoDecodeChunk (CF\_DecoderState\_t \*state, size\_t chunksize)**  
*Decode a block of data from the PDU.*
- **uint8 CF\_CFDP\_GetValueEncodedSize (uint64 Value)**  
*Gets the minimum number of octets that the given integer may be encoded in.*
- **void CF\_EncodeIntegerInSize (CF\_EncoderState\_t \*state, uint64 value, uint8 encode\_size)**  
*Encodes the given integer value in the given number of octets.*
- **uint64 CF\_DecodeIntegerInSize (CF\_DecoderState\_t \*state, uint8 decode\_size)**  
*Decodes an integer value from the specified number of octets.*
- **void CF\_CFDP\_EncodeHeaderWithoutSize (CF\_EncoderState\_t \*state, CF\_Logical\_PduHeader\_t \*plh)**  
*Encodes a CFDP PDU base header block, bypassing the size field.*
- **void CF\_CFDP\_EncodeHeaderFinalSize (CF\_EncoderState\_t \*state, CF\_Logical\_PduHeader\_t \*plh)**  
*Updates an already-encoded PDU base header block with the final PDU size.*
- **void CF\_CFDP\_EncodeFileDirectiveHeader (CF\_EncoderState\_t \*state, CF\_Logical\_PduFileDirectiveHeader\_t \*pfdir)**  
*Encodes a CFDP file directive header block.*
- **void CF\_CFDP\_EncodeLV (CF\_EncoderState\_t \*state, CF\_Logical\_Lv\_t \*pllv)**  
*Encodes a single CFDP Length+Value (LV) pair.*
- **void CF\_CFDP\_EncodeTLV (CF\_EncoderState\_t \*state, CF\_Logical\_Tlv\_t \*pltlv)**  
*Encodes a single CFDP Type+Length+Value (TLV) tuple.*
- **void CF\_CFDP\_EncodeSegmentRequest (CF\_EncoderState\_t \*state, CF\_Logical\_SegmentRequest\_t \*plseg)**  
*Encodes a single CFDP Segment Request block.*

- void `CF_CFDP_EncodeAllTlv` (`CF_EncoderState_t` \*state, `CF_Logical_TlvList_t` \*pltlv)  
*Encodes a list of CFDP Type+Length+Value tuples.*
- void `CF_CFDP_EncodeAllSegments` (`CF_EncoderState_t` \*state, `CF_Logical_SegmentList_t` \*plseg)  
*Encodes a list of CFDP Segment Request blocks.*
- void `CF_CFDP_EncodeMd` (`CF_EncoderState_t` \*state, `CF_Logical_PduMd_t` \*plmd)  
*Encodes a CFDP Metadata (MD) header block.*
- void `CF_CFDP_EncodeFileDataHeader` (`CF_EncoderState_t` \*state, bool with\_meta, `CF_Logical_PduFileDataHeader_t` \*plfd)  
*Encodes a CFDP File Data (FD) header block.*
- void `CF_CFDP_EncodeEof` (`CF_EncoderState_t` \*state, `CF_Logical_PduEof_t` \*pleof)  
*Encodes a CFDP End-of-File (EOF) header block.*
- void `CF_CFDP_EncodeFin` (`CF_EncoderState_t` \*state, `CF_Logical_PduFin_t` \*plfin)  
*Encodes a CFDP Final (FIN) header block.*
- void `CF_CFDP_EncodeAck` (`CF_EncoderState_t` \*state, `CF_Logical_PduAck_t` \*plack)  
*Encodes a CFDP Acknowledge (ACK) header block.*
- void `CF_CFDP_EncodeNak` (`CF_EncoderState_t` \*state, `CF_Logical_PduNak_t` \*plnak)  
*Encodes a CFDP Non-Acknowledge (NAK) header block.*
- void `CF_CFDP_EncodeCrc` (`CF_EncoderState_t` \*state, `uint32` \*plcrc)  
*Encodes a CFDP CRC/Checksum.*
- `CFE_Status_t CF_CFDP_DecodeHeader` (`CF_DecoderState_t` \*state, `CF_Logical_PduHeader_t` \*plh)  
*Decodes a CFDP base PDU header.*
- void `CF_CFDP_DecodeFileDirectiveHeader` (`CF_DecoderState_t` \*state, `CF_Logical_PduFileDirectiveHeader_t` \*pfdir)  
*Decodes a CFDP file directive header block.*
- void `CF_CFDP_DecodeLV` (`CF_DecoderState_t` \*state, `CF_Logical_Lv_t` \*pllv)  
*Decodes a single CFDP Length+Value (LV) pair.*
- void `CF_CFDP_DecodeTLV` (`CF_DecoderState_t` \*state, `CF_Logical_Tlv_t` \*pltlv)  
*Decodes a single CFDP Type+Length+Value (TLV) tuple.*
- void `CF_CFDP_DecodeSegmentRequest` (`CF_DecoderState_t` \*state, `CF_Logical_SegmentRequest_t` \*plseg)  
*Decodes a single CFDP Segment Request block.*
- void `CF_CFDP_DecodeAllTlv` (`CF_DecoderState_t` \*state, `CF_Logical_TlvList_t` \*pltlv, `uint8` limit)  
*Decodes a list of CFDP Type+Length+Value tuples.*
- void `CF_CFDP_DecodeAllSegments` (`CF_DecoderState_t` \*state, `CF_Logical_SegmentList_t` \*plseg, `uint8` limit)  
*Decodes a list of CFDP Segment Request blocks.*
- void `CF_CFDP_DecodeMd` (`CF_DecoderState_t` \*state, `CF_Logical_PduMd_t` \*plmd)  
*Decodes a CFDP Metadata (MD) header block.*
- void `CF_CFDP_DecodeFileDataHeader` (`CF_DecoderState_t` \*state, bool with\_meta, `CF_Logical_PduFileDataHeader_t` \*plfd)  
*Decodes a CFDP File Data (FD) header block.*
- void `CF_CFDP_DecodeEof` (`CF_DecoderState_t` \*state, `CF_Logical_PduEof_t` \*pleof)  
*Decodes a CFDP End-of-File (EOF) header block.*
- void `CF_CFDP_DecodeFin` (`CF_DecoderState_t` \*state, `CF_Logical_PduFin_t` \*plfin)  
*Decodes a CFDP Final (FIN) header block.*
- void `CF_CFDP_DecodeAck` (`CF_DecoderState_t` \*state, `CF_Logical_PduAck_t` \*plack)  
*Decodes a CFDP Acknowledge (ACK) header block.*
- void `CF_CFDP_DecodeNak` (`CF_DecoderState_t` \*state, `CF_Logical_PduNak_t` \*plnak)  
*Decodes a CFDP Non-Acknowledge (NAK) header block.*
- void `CF_CFDP_DecodeCrc` (`CF_DecoderState_t` \*state, `uint32` \*plcrc)  
*Decodes a CFDP CRC/Checksum.*

### 12.53.1 Detailed Description

CFDP protocol data structure encode/decode API declarations

### 12.53.2 Macro Definition Documentation

**12.53.2.1 CF\_CODEC\_GET\_POSITION** #define CF\_CODEC\_GET\_POSITION(  
    s ) (CF\_CFDP\_CodecGetPosition(&((s)->codec\_state)))

Macro wrapper around [CF\\_CFDP\\_CodecGetPosition\(\)](#)

Checks the position of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF\_EncoderState\_t or CF\_DecoderState\_t object.

#### Parameters

s	Encoder or Decoder state
---	--------------------------

Definition at line 268 of file cf\_codec.h.

**12.53.2.2 CF\_CODEC\_GET\_REMAIN** #define CF\_CODEC\_GET\_REMAIN(  
    s ) (CF\_CFDP\_CodecGetRemain(&((s)->codec\_state)))

Macro wrapper around [CF\\_CFDP\\_CodecGetRemain\(\)](#)

Checks the remainder of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF\_EncoderState\_t or CF\_DecoderState\_t object.

#### Parameters

s	Encoder or Decoder state
---	--------------------------

Definition at line 280 of file cf\_codec.h.

**12.53.2.3 CF\_CODEC\_GET\_SIZE** #define CF\_CODEC\_GET\_SIZE(  
    s ) (CF\_CFDP\_CodecGetSize(&((s)->codec\_state)))

Macro wrapper around [CF\\_CFDP\\_CodecGetSize\(\)](#)

Checks the size of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF\_EncoderState\_t or CF\_DecoderState\_t object.

#### Parameters

s	Encoder or Decoder state
---	--------------------------

Definition at line 292 of file cf\_codec.h.

**12.53.2.4 CF\_CODEC\_IS\_OK** #define CF\_CODEC\_IS\_OK(  
    s ) (CF\_CFDP\_CodecIsOK(&((s)->codec\_state)))

Macro wrapper around [CF\\_CFDP\\_CodecIsOK\(\)](#)

Checks the state of either an encoder or decoder object This just simplifies the code, as same macro may be used with either an CF\_EncoderState\_t or CF\_DecoderState\_t object.

**Parameters**

<code>s</code>	Encoder or Decoder state
----------------	--------------------------

Definition at line 244 of file cf\_codec.h.

**12.53.2.5 CF\_CODEC\_SET\_DONE** `#define CF_CODEC_SET_DONE(`  
 `s ) (CF_CFDP_CodecSetDone(&((s)->codec_state)))`

Macro wrapper around [CF\\_CFDP\\_CodecSetDone\(\)](#)

Sets the state of either an encoder or decoder object. This just simplifies the code, as same macro may be used with either an CF\_EncoderState\_t or CF\_DecoderState\_t object.

**Parameters**

<code>s</code>	Encoder or Decoder state
----------------	--------------------------

Definition at line 256 of file cf\_codec.h.

**12.53.2.6 CF\_DECODE\_FIXED\_CHUNK** `#define CF_DECODE_FIXED_CHUNK(`  
 `state,`  
 `type ) ((const type *)CF_CFDP_DoDecodeChunk(state, sizeof(type)))`

Macro to decode a block of a given CFDP type into a PDU.

This is a wrapper around [CF\\_CFDP\\_DoDecodeChunk\(\)](#) to encode the given data type, rather than a generic size. The sizeof() the type should reflect the *encoded* size within the PDU. Specifically, this must only be used with the "CFDP" data types which are specifically designed to match the binary layout of the CFDP-defined header structures.

**Parameters**

<code>state</code>	Decoder state object
<code>type</code>	Data type to decode, from <a href="#">cf_cfdp_pdu.h</a>

**Returns**

Pointer to block, if successful

**Return values**

<code>NULL</code>	if not successful (no space or other error).
-------------------	--

Definition at line 232 of file cf\_codec.h.

**12.53.2.7 CF\_ENCODE\_FIXED\_CHUNK** `#define CF_ENCODE_FIXED_CHUNK(`  
 `state,`  
 `type ) ((type *)CF_CFDP_DoEncodeChunk(state, sizeof(type)))`

Macro to encode a block of a given CFDP type into a PDU.

This is a wrapper around [CF\\_CFDP\\_DoEncodeChunk\(\)](#) to encode the given data type, rather than a generic size. The sizeof() the type should reflect the *encoded* size within the PDU. Specifically, this must only be used with the "CFDP" data types which are specifically designed to match the binary layout of the CFDP-defined header structures.

**Parameters**

<i>state</i>	Encoder state object
<i>type</i>	Data type to encode, from <a href="#">cf_cfdp_pdu.h</a>

**Returns**

Pointer to block, if successful

**Return values**

<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 215 of file cf\_codec.h.

**12.53.3 Typedef Documentation****12.53.3.1 CF\_CodecState\_t** `typedef struct CF_CodecState CF_CodecState_t`

Tracks the current state of an encode or decode operation.

This encapsulates the common state between encode and decode

**12.53.3.2 CF\_DecoderState\_t** `typedef struct CF_DecoderState CF_DecoderState_t`

Current state of a decode operation.

State structure for decodes

**12.53.3.3 CF\_EncoderState\_t** `typedef struct CF_EncoderState CF_EncoderState_t`

Current state of an encode operation.

State structure for encodes

**12.53.4 Function Documentation****12.53.4.1 CF\_CFDP\_CodecCheckSize()** `bool CF_CFDP_CodecCheckSize (`

`CF_CodecState_t * state,`

`size_t chunksize )`

Advances the position by the indicated size, confirming the block will fit into the PDU.

On encode, this confirms there is enough available space to hold a block of the indicated size. On decode, this confirms that decoding the indicated number of bytes will not read beyond the end of data.

If true, then the current position/offset is advanced by the indicated number of bytes. If false, then the error flag is set, so that future calls to CF\_CFDP\_CodecIsOK will also return false.

**Note**

The error flag is sticky, meaning that if any encode/decode operation fails, all future encode/decode requests on the same state will also fail. Each encode/decode step must check the flag, and skip the operation if it is false. Reporting the error can be deferred to the final stage, and only done once.

**Parameters**

<i>state</i>	Encoder/Decoder common state
<i>chunksize</i>	Size of next block to encode/decode

### Return values

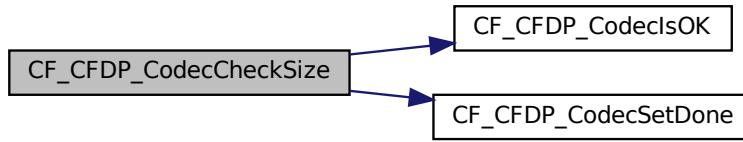
<i>true</i>	If encode/decode is possible, enough space exists
<i>false</i>	If encode/decode is not possible, not enough space or prior error occurred

Definition at line 271 of file cf\_codec.c.

References CF\_CFDP\_CodecIsOK(), CF\_CFDP\_CodecSetDone(), CF\_CodecState::max\_size, and CF\_CodecState::next\_offset.

Referenced by CF\_CFDP\_DoDecodeChunk(), and CF\_CFDP\_DoEncodeChunk().

Here is the call graph for this function:



**12.53.4.2 CF\_CFDP\_CodecGetPosition()** static size\_t CF\_CFDP\_CodecGetPosition (  
    const CF\_CodecState\_t \* state ) [inline], [static]

Obtains the current position/offset within the PDU.

### Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

### Returns

Current offset in PDU

Definition at line 107 of file cf\_codec.h.

References CF\_CodecState::next\_offset.

Referenced by CF\_CFDP\_DoDecodeChunk(), and CF\_CFDP\_DoEncodeChunk().

**12.53.4.3 CF\_CFDP\_CodecGetRemain()** static size\_t CF\_CFDP\_CodecGetRemain (  
    const CF\_CodecState\_t \* state ) [inline], [static]

Obtains the remaining size of the PDU being encoded/decoded.

### Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

**Returns**

Remaining size of PDU

Definition at line 131 of file cf\_codec.h.

References CF\_CodecState::max\_size, and CF\_CodecState::next\_offset.

**12.53.4.4 CF\_CFDP\_CodecGetSize()** static size\_t CF\_CFDP\_CodecGetSize ( const CF\_CodecState\_t \* state ) [inline], [static]

Obtains the maximum size of the PDU being encoded/decoded.

**Parameters**

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

**Returns**

Maximum size of PDU

Definition at line 119 of file cf\_codec.h.

References CF\_CodecState::max\_size.

**12.53.4.5 CF\_CFDP\_CodecIsOK()** static bool CF\_CFDP\_CodecIsOK ( const CF\_CodecState\_t \* state ) [inline], [static]

Checks if the codec is currently valid or not.

**Parameters**

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

**Return values**

<i>true</i>	If encoder/decoder is still valid, has not reached end of PDU
<i>false</i>	If encoder/decoder is not valid, has reached end of PDU or an error occurred

Definition at line 82 of file cf\_codec.h.

References CF\_CodecState::is\_valid.

Referenced by CF\_CFDP\_CodecCheckSize().

**12.53.4.6 CF\_CFDP\_CodecReset()** static void CF\_CFDP\_CodecReset ( CF\_CodecState\_t \* state, size\_t max\_size ) [inline], [static]

Resets a codec state.

**Parameters**

<i>state</i>	Encoder/Decoder common state
<i>max_size</i>	Maximum size of PDU

Definition at line 143 of file cf\_codec.h.

References CF\_CodecState::is\_valid, CF\_CodecState::max\_size, and CF\_CodecState::next\_offset.  
Referenced by CF\_CFDP\_DecodeStart(), and CF\_CFDP\_EncodeStart().

#### **12.53.4.7 CF\_CFDP\_CodecSetDone()** static void CF\_CFDP\_CodecSetDone ( CF\_CodecState\_t \* state ) [inline], [static]

Sets a codec to the "done" state.

This may mean end of PDU data is reached, or that an error occurred

##### Parameters

<i>state</i>	Encoder/Decoder common state
--------------	------------------------------

Definition at line 95 of file cf\_codec.h.

References CF\_CodecState::is\_valid.

Referenced by CF\_CFDP\_CodecCheckSize(), CF\_CFDP\_DecodeStart(), and CF\_CFDP\_EncodeStart().

#### **12.53.4.8 CF\_CFDP\_DecodeAck()** void CF\_CFDP\_DecodeAck ( CF\_DecoderState\_t \* state, CF\_Logical\_PduAck\_t \* plack )

Decodes a CFDP Acknowledge (ACK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

##### Parameters

<i>state</i>	Decoder state object
<i>plack</i>	Pointer to logical PDU ACK header data

Definition at line 1049 of file cf\_codec.c.

References CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::ack\_subtype\_code, CF\_Logical\_PduAck::cc, CF\_CFDP\_PduAck::cc\_and\_transaction\_status, CF\_CFDP\_PduAck\_CC, CF\_CFDP\_PduAck\_DIR\_CODE, CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE, CF\_CFDP\_PduAck\_TRANSACTION\_STATUS, CF\_DECODE\_FIXED\_CHUNK, CF\_CFDP\_PduAck::directive\_and\_subtype\_code, FGV, and CF\_Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_RecvAck().

#### **12.53.4.9 CF\_CFDP\_DecodeAllSegments()** void CF\_CFDP\_DecodeAllSegments ( CF\_DecoderState\_t \* state, CF\_Logical\_SegmentList\_t \* plseg, uint8 limit )

Decodes a list of CFDP Segment Request blocks.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

##### Parameters

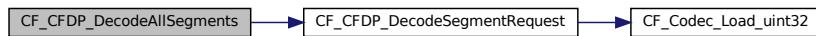
<i>state</i>	Decoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data
<i>limit</i>	Maximum number of Segment Request objects to decode

Definition at line 1125 of file cf\_codec.c.

References CF\_CFDP\_DecodeSegmentRequest(), CF\_CODEC\_GET\_REMAIN, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_PDU\_MAX\_SEGMENTS, CF\_Logical\_SegmentList::num\_segments, and CF\_Logical\_SegmentList::segments.

Referenced by CF\_CFDP\_DecodeNak().

Here is the call graph for this function:



**12.53.4.10 CF\_CFDP\_DecodeAllTlv()** void CF\_CFDP\_DecodeAllTlv (

```

    CF_DecoderState_t * state,
    CF_Logical_TlvList_t * ptlvt,
    uint8 limit )

```

Decodes a list of CFDP Type+Length+Value tuples.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

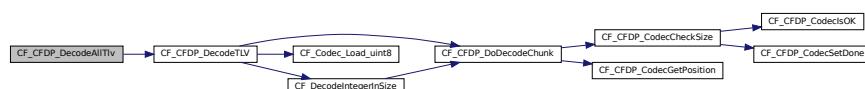
<i>state</i>	Decoder state object
<i>ptlvt</i>	Pointer to logical PDU TLV header data
<i>limit</i>	Maximum number of TLV objects to decode

Definition at line 1090 of file cf\_codec.c.

References CF\_CFDP\_DecodeTLV(), CF\_CODEC\_GET\_REMAIN, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_PDU\_MAX\_TLV, CF\_Logical\_TlvList::num\_tlv, and CF\_Logical\_TlvList::tlv.

Referenced by CF\_CFDP\_DecodeEof(), and CF\_CFDP\_DecodeFin().

Here is the call graph for this function:



**12.53.4.11 CF\_CFDP\_DecodeCrc()** void CF\_CFDP\_DecodeCrc (

```

    CF_DecoderState_t * state,
    uint32 * plcrc )

```

Decodes a CFDP CRC/Checksum.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

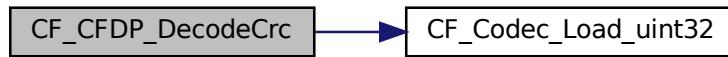
### Parameters

<i>state</i>	Decoder state object
<i>plcrc</i>	Pointer to logical CRC value

Definition at line 990 of file cf\_codec.c.

References CF\_Codec\_Load\_uint32(), and CF\_DECODE\_FIXED\_CHUNK.

Here is the call graph for this function:



```

12.53.4.12 CF_CFDP_DecodeEof() void CF_CFDP_DecodeEof (
    CF_DecoderState_t * state,
    CF_Logical_PduEof_t * pleof )

```

Decodes a CFDP End-of-File (EOF) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

### Parameters

<i>state</i>	Decoder state object
<i>pleof</i>	Pointer to logical PDU EOF header data

Definition at line 1007 of file cf\_codec.c.

References CF\_CFDP\_PduEof::cc, CF\_Logical\_PduEof::cc, CF\_CFDP\_DecodeAllTlv(), CF\_CFDP\_PduEof\_FLAGS←\_CC, CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_PDU\_MAX\_TLV, CF\_CFDP\_PduEof::crc, CF\_Logical\_PduEof::crc, FGV, CF\_CFDP\_PduEof::size, CF\_Logical\_PduEof::size, and CF\_Logical\_PduEof::tlv\_list.

Referenced by CF\_CFDP\_RecvEof().

Here is the call graph for this function:



```

12.53.4.13 CF_CFDP_DecodeFileDataHeader() void CF_CFDP_DecodeFileDataHeader (
    CF_DecoderState_t * state,
    bool with_meta,
    CF_Logical_PduFileDataHeader_t * plfd )

```

Decodes a CFDP File Data (FD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

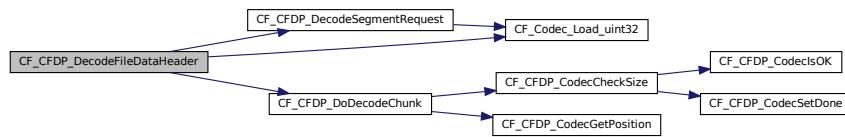
<i>state</i>	Decoder state object
<i>with_meta</i>	Whether to include optional continuation and segment request fields (always false currently)
<i>pfd</i>	Pointer to logical PDU file header data

Definition at line 928 of file cf\_codec.c.

References CF\_CFDP\_DecodeSegmentRequest(), CF\_CFDP\_DoDecodeChunk(), CF\_CFDP\_PduFileData\_<RECORD\_CONTINUE\_STATE, CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH, CF\_CODEC\_GET\_<\_REMAIN, CF\_CODEC\_IS\_OK, CF\_Codec\_Load\_uint32(), CF\_CODEC\_SET\_DONE, CF\_DECODE\_FIXED\_CHUNK, CF\_PDU\_MAX\_SEGMENTS, CF\_Logical\_PduFileDataHeader::continuation\_state, CF\_Logical\_PduFileDataHeader<::data\_len, CF\_Logical\_PduFileDataHeader::data\_ptr, FGV, CF\_Logical\_SegmentList::num\_segments, CF\_CFDP\_<\_PduFileDataHeader::offset, CF\_Logical\_PduFileDataHeader::offset, CF\_Logical\_PduFileDataHeader::segment\_list, and CF\_Logical\_SegmentList::segments.

Referenced by CF\_CFDP\_RecvFd().

Here is the call graph for this function:



#### 12.53.4.14 CF\_CFDP\_DecodeFileDirectiveHeader()

```
void CF_CFDP_DecodeFileDirectiveHeader (
    CF_DecoderState_t * state,
    CF_Logical_PduFileDirectiveHeader_t * pfdir )
```

Decodes a CFDP file directive header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

<i>state</i>	Decoder state object
<i>pfdi</i>	Pointer to logical PDU file directive header data

Definition at line 818 of file cf\_codec.c.

References CF\_Codec\_Load\_uint8(), CF\_DECODE\_FIXED\_CHUNK, CF\_CFDP\_PduFileDirectiveHeader::directive\_<\_code, and CF\_Logical\_PduFileDirectiveHeader::directive\_code.

Referenced by CF\_CFDP\_RecvPh().

Here is the call graph for this function:



**12.53.4.15 CF\_CFDP\_DecodeFin()** `void CF_CFDP_DecodeFin (`  
 `CF_DecoderState_t * state,`  
 `CF_Logical_PduFin_t * plfin )`

Decodes a CFDP Final (FIN) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

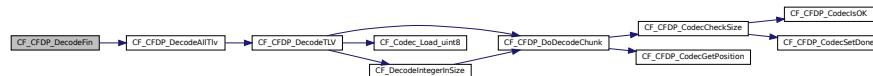
<code>state</code>	Decoder state object
<code>plfin</code>	Pointer to logical PDU FIN header data

Definition at line 1028 of file cf\_codec.c.

References CF\_Logical\_PduFin::cc, CF\_CFDP\_DecodeAllTlv(), CF\_CFDP\_PduFin\_FLAGS\_CC, CF\_CFDP\_PduFin\_FLAGS\_DELIVERY\_CODE, CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS, CF\_DECODE\_FIXED\_CHUNK, CF\_PDU\_MAX\_TLV, CF\_Logical\_PduFin::delivery\_code, FGV, CF\_Logical\_PduFin::file\_status, CF\_CFDP\_PduFin::flags, and CF\_Logical\_PduFin::tlv\_list.

Referenced by CF\_CFDP\_RecvFin().

Here is the call graph for this function:



**12.53.4.16 CF\_CFDP\_DecodeHeader()** `CFE_Status_t CF_CFDP_DecodeHeader (`  
 `CF_DecoderState_t * state,`  
 `CF_Logical_PduHeader_t * plh )`

Decodes a CFDP base PDU header.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

**Note**

On decode the entire base header is decoded in a single call, the size will be decoded like any other field.

**Parameters**

<i>state</i>	Decoder state object
<i>plh</i>	Pointer to logical PDU base header data

**Return values**

<a href="#">CFE_SUCCESS</a>	Successful execution. Operation was performed successfully
<a href="#">CF_ERROR</a>	on error.

Definition at line 770 of file cf\_codec.c.

References CF\_CFDP\_PduHeader\_FLAGS\_CRC, CF\_CFDP\_PduHeader\_FLAGS\_DIR, CF\_CFDP\_PduHeader\_FLAGS\_LARGEFILE, CF\_CFDP\_PduHeader\_FLAGS\_MODE, CF\_CFDP\_PduHeader\_FLAGS\_TYPE, CF\_CFDP\_PduHeader\_FLAGS\_VERSION, CF\_CFDP\_PduHeader\_LENGTHS\_ENTITY, CF\_CFDP\_PduHeader\_LENGTHS\_TRANSACTION\_SEQUENCE, CF\_CFDP\_PduHeader\_SEGMENT\_METADATA, CF\_CFDP\_PduHeader\_SEGMENTATION\_CONTROL, CF\_CODEC\_GET\_POSITION, CF\_Codec\_Load\_uint16(), CF\_DECODE\_FIXED\_CHUNK, CF\_DecodeIntegerInSize(), CF\_ERROR, CFE\_SUCCESS, CF\_Logical\_PduHeader::crc\_flag, CF\_Logical\_PduHeader::data\_encoded\_length, CF\_Logical\_PduHeader::destination\_eid, CF\_Logical\_PduHeader::direction, CF\_Logical\_PduHeader::eid\_length, CF\_CFDP\_PduHeader::eid\_tsn\_lengths, FGV, CF\_CFDP\_PduHeader::flags, CF\_Logical\_PduHeader::header\_encoded\_length, CF\_Logical\_PduHeader::large\_flag, CF\_CFDP\_PduHeader::length, CF\_Logical\_PduHeader::pdu\_type, CF\_Logical\_PduHeader::segment\_meta\_flag, CF\_Logical\_PduHeader::segmentation\_control, CF\_Logical\_PduHeader::sequence\_num, CF\_Logical\_PduHeader::source\_eid, CF\_Logical\_PduHeader::txm\_mode, CF\_Logical\_PduHeader::txn\_seq\_length, and CF\_Logical\_PduHeader::version.

Referenced by CF\_CFDP\_RecvPh().

Here is the call graph for this function:



**12.53.4.17 CF\_CFDP\_DecodeLV()** void CF\_CFDP\_DecodeLV (   
*CF\_DecoderState\_t* \* *state*,   
*CF\_Logical\_Lv\_t* \* *pllv* )

Decodes a single CFDP Length+Value (LV) pair.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

**Parameters**

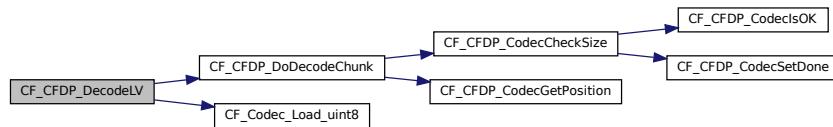
<i>state</i>	Decoder state object
<i>pllv</i>	Pointer to single logical PDU LV data

Definition at line 838 of file cf\_codec.c.

References CF\_CFDP\_DoDecodeChunk(), CF\_Codec\_Load\_uint8(), CF\_DECODE\_FIXED\_CHUNK, CF\_Logical\_Lv::data\_ptr, CF\_CFDP\_Lv::length, and CF\_Logical\_Lv::length.

Referenced by CF\_CFDP\_DecodeMd().

Here is the call graph for this function:



**12.53.4.18 CF\_CFDP\_DecodeMd()** `void CF_CFDP_DecodeMd ( CF_DecoderState_t * state,  
CF_Logical_PduMd_t * plmd )`

Decodes a CFDP Metadata (MD) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

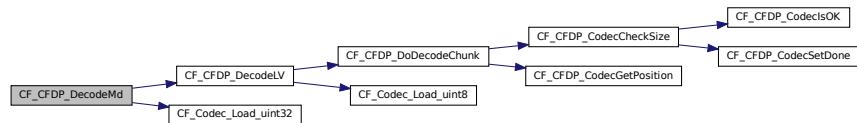
<code>state</code>	Decoder state object
<code>plmd</code>	Pointer to logical PDU metadata header data

Definition at line 905 of file cf\_codec.c.

References CF\_CFDP\_DecodeLV(), CF\_CFDP\_PduMd\_CHECKSUM\_TYPE, CF\_CFDP\_PduMd\_CLOSURE\_REQUESTED, CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_Logical\_PduMd::checksum\_type, CF\_Logical\_PduMd::close\_req, CF\_Logical\_PduMd::dest\_filename, FGV, CF\_CFDP\_PduMd::segmentation\_control, CF\_CFDP\_PduMd::size, CF\_Logical\_PduMd::size, and CF\_Logical\_PduMd::source\_filename.

Referenced by CF\_CFDP\_RecvMd().

Here is the call graph for this function:



**12.53.4.19 CF\_CFDP\_DecodeNak()** `void CF_CFDP_DecodeNak ( CF_DecoderState_t * state,  
CF_Logical_PduNak_t * plnak )`

Decodes a CFDP Non-Acknowledge (NAK) header block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

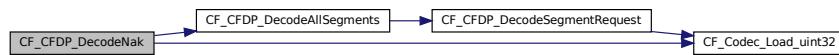
<i>state</i>	Decoder state object
<i>plnak</i>	Pointer to logical PDU NAK header data

Definition at line 1070 of file cf\_codec.c.

References CF\_CFDP\_DecodeAllSegments(), CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_PDU←\_MAX\_SEGMENTS, CF\_CFDP\_PduNak::scope\_end, CF\_Logical\_PduNak::scope\_end, CF\_CFDP\_PduNak::scope←\_start, CF\_Logical\_PduNak::scope\_start, and CF\_Logical\_PduNak::segment\_list.

Referenced by CF\_CFDP\_RecvNak().

Here is the call graph for this function:



**12.53.4.20 CF\_CFDP\_DecodeSegmentRequest()** void CF\_CFDP\_DecodeSegmentRequest ( CF\_DecoderState\_t \* state, CF\_Logical\_SegmentRequest\_t \* plseg )

Decodes a single CFDP Segment Request block.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

<i>state</i>	Decoder state object
<i>plseg</i>	Pointer to single logical PDU segment request header data

Definition at line 887 of file cf\_codec.c.

References CF\_Codec\_Load\_uint32(), CF\_DECODE\_FIXED\_CHUNK, CF\_CFDP\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_end, CF\_CFDP\_SegmentRequest::offset\_start, and CF\_Logical\_SegmentRequest::offset\_start.

Referenced by CF\_CFDP\_DecodeAllSegments(), and CF\_CFDP\_DecodeFileDataHeader().

Here is the call graph for this function:



```
12.53.4.21 CF_CFDP_DecodeTLV() void CF_CFDP_DecodeTLV (
    CF_DecoderState_t * state,
    CF_Logical_Tlv_t * pltlv )
```

Decodes a single CFDP Type+Length+Value (TLV) tuple.

The data will be decoded from the encoded PDU at the current position and the logical fields will be saved to the given data structure

If the encoder is in an error state, nothing is decoded, and the state of the decoder is not changed.

#### Parameters

<i>state</i>	Decoder state object
<i>pltlv</i>	Pointer to single logical PDU TLV data

Definition at line 856 of file cf\_codec.c.

References CF\_CFDP\_DoDecodeChunk(), CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, CF\_Codec\_Load\_uint8(), CF↔ DECODE\_FIXED\_CHUNK, CF\_DecodeIntegerInSize(), CF\_Logical\_Tlv::data, CF\_Logical\_TlvData::data\_ptr, CF↔\_Logical\_TlvData::eid, CF\_CFDP\_tlv::length, CF\_Logical\_Tlv::length, CF\_CFDP\_tlv::type, and CF\_Logical\_Tlv::type.

Referenced by CF\_CFDP\_DecodeAllTlv().

Here is the call graph for this function:



```
12.53.4.22 CF_CFDP_DoDecodeChunk() const void* CF_CFDP_DoDecodeChunk (
    CF_DecoderState_t * state,
    size_t chunksize )
```

Decode a block of data from the PDU.

Deducts space for a block of given size from the current PDU

#### Parameters

<i>state</i>	Decoder state object
<i>chunksize</i>	Size of block to decode

#### Returns

Pointer to block, if successful

#### Return values

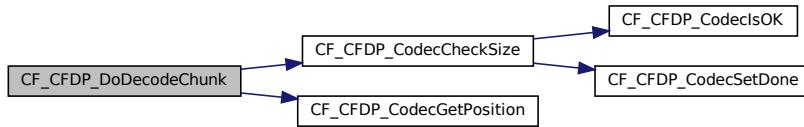
<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 311 of file cf\_codec.c.

References CF\_DecoderState::base, CF\_CFDP\_CodecCheckSize(), CF\_CFDP\_CodecGetPosition(), and CF\_DecoderState::codec\_state.

Referenced by CF\_CFDP\_DecodeFileDataHeader(), CF\_CFDP\_DecodeLV(), CF\_CFDP\_DecodeTLV(), and CF\_DecodeIntegerInSize().

Here is the call graph for this function:



#### 12.53.4.23 CF\_CFDP\_DoEncodeChunk() void\* CF\_CFDP\_DoEncodeChunk (

```

        CF_EncoderState_t * state,
        size_t chunksize )
  
```

Encode a block of data into the PDU.

Adds/Reserves space for a block of the given size in the current PDU

##### Parameters

<i>state</i>	Encoder state object
<i>chunksize</i>	Size of block to encode

##### Returns

Pointer to block, if successful

##### Return values

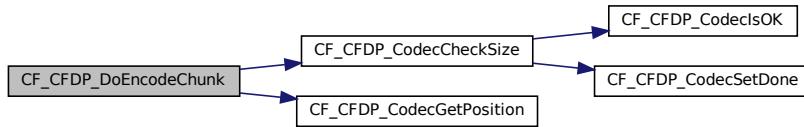
<i>NULL</i>	if not successful (no space or other error).
-------------	--

Definition at line 293 of file cf\_codec.c.

References CF\_EncoderState::base, CF\_CFDP\_CodecCheckSize(), CF\_CFDP\_CodecGetPosition(), and CF\_EncoderState::codec\_state.

Referenced by CF\_CFDP\_EncodeLV(), CF\_CFDP\_EncodeTLV(), CF\_CFDP\_S\_SendFileData(), and CF\_EncodeIntegerInSize().

Here is the call graph for this function:



```
12.53.4.24 CF_CFDP_EncodeAck() void CF_CFDP_EncodeAck (
    CF_EncoderState_t * state,
    CF_Logical_PduAck_t * plack )
```

Encodes a CFDP Acknowledge (ACK) header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

<i>state</i>	Encoder state object
<i>plack</i>	Pointer to logical PDU ACK header data

Definition at line 682 of file cf\_codec.c.

References CF\_Logical\_PduAck::ack\_directive\_code, CF\_Logical\_PduAck::ack\_subtype\_code, CF\_Logical\_PduAck::cc, CF\_CFDP\_PduAck::cc\_and\_transaction\_status, CF\_CFDP\_PduAck\_CC, CF\_CFDP\_PduAck\_DIR\_CODE, CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE, CF\_CFDP\_PduAck\_TRANSACTION\_STATUS, CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_PduAck::directive\_and\_subtype\_code, FSV, and CF\_Logical\_PduAck::txn\_status.

Referenced by CF\_CFDP\_SendAck().

Here is the call graph for this function:



```
12.53.4.25 CF_CFDP_EncodeAllSegments() void CF_CFDP_EncodeAllSegments (
    CF_EncoderState_t * state,
    CF_Logical_SegmentList_t * plseg )
```

Encodes a list of CFDP Segment Request blocks.

This invokes [CF\\_CFDP\\_EncodeSegmentRequest\(\)](#) for all segments in the given list.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

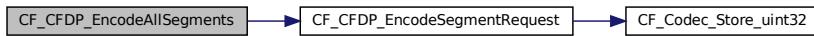
<i>state</i>	Encoder state object
<i>plseg</i>	Pointer to logical PDU segment request header data

Definition at line 561 of file cf\_codec.c.

References CF\_CFDP\_EncodeSegmentRequest(), CF\_CODEC\_IS\_OK, CF\_Logical\_SegmentList::num\_segments, and CF\_Logical\_SegmentList::segments.

Referenced by CF\_CFDP\_EncodeFileDataHeader(), and CF\_CFDP\_EncodeNak().

Here is the call graph for this function:



#### 12.53.4.26 CF\_CFDP\_EncodeAllTlv()

```
void CF_CFDP_EncodeAllTlv (
    CF_EncoderState_t * state,
    CF_Logical_TlvList_t * pltlv )
```

Encodes a list of CFDP Type+Length+Value tuples.

This invokes [CF\\_CFDP\\_EncodeTLV\(\)](#) for all TLV values in the given list.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

##### Parameters

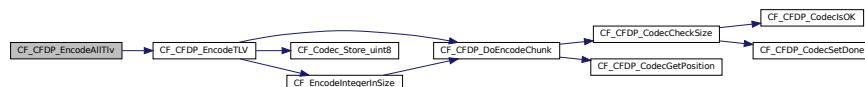
<i>state</i>	Encoder state object
<i>pltlv</i>	Pointer to logical PDU TLV header data

Definition at line 545 of file cf\_codec.c.

References [CF\\_CFDP\\_EncodeTLV\(\)](#), [CF\\_CODEC\\_IS\\_OK](#), [CF\\_Logical\\_TlvList::num\\_tlv](#), and [CF\\_Logical\\_TlvList::tlv](#).

Referenced by [CF\\_CFDP\\_EncodeEof\(\)](#), and [CF\\_CFDP\\_EncodeFin\(\)](#).

Here is the call graph for this function:



#### 12.53.4.27 CF\_CFDP\_EncodeCrc()

```
void CF_CFDP_EncodeCrc (
    CF_EncoderState_t * state,
    uint32 * plcrc )
```

Encodes a CFDP CRC/Checksum.

The data in the logical header will be appended to the encoded PDU at the current position

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

##### Parameters

<i>state</i>	Encoder state object
<i>plcrc</i>	Pointer to logical CRC value

Definition at line 725 of file cf\_codec.c.

References [CF\\_Codec\\_Store\\_uint32\(\)](#), and [CF\\_ENCODE\\_FIXED\\_CHUNK](#).

Here is the call graph for this function:



**12.53.4.28 CF\_CFDP\_EncodeEof()** `void CF_CFDP_EncodeEof (`  
 `CF_EncoderState_t * state,`  
 `CF_Logical_PduEof_t * pleof )`

Encodes a CFDP End-of-File (EOF) header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Note

this encode includes any TLV values which are indicated in the logical data structure

#### Parameters

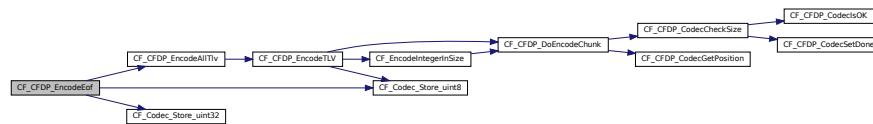
<code>state</code>	Encoder state object
<code>pleof</code>	Pointer to logical PDU EOF header data

Definition at line 638 of file cf\_codec.c.

References CF\_CFDP\_PduEof::cc, CF\_Logical\_PduEof::cc, CF\_CFDP\_EncodeAllTlv(), CF\_CFDP\_PduEof\_FLAGS←\_CC, CF\_Codec\_Store\_uint32(), CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_PduEof::crc, CF\_Logical\_PduEof::crc, FSV, CF\_CFDP\_PduEof::size, CF\_Logical\_PduEof::size, and CF\_Logical\_PduEof::tlv\_list.

Referenced by CF\_CFDP\_SendEof().

Here is the call graph for this function:



**12.53.4.29 CF\_CFDP\_EncodeFileDataHeader()** `void CF_CFDP_EncodeFileDataHeader (`  
 `CF_EncoderState_t * state,`  
 `bool with_meta,`  
 `CF_Logical_PduFileDataHeader_t * plfd )`

Encodes a CFDP File Data (FD) header block.

This only encodes the FD header fields, specifically the data offset (required) and any metadata fields, if indicated. This does *not* encode any actual file data.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

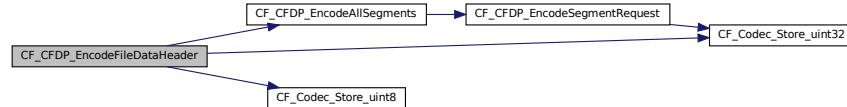
<i>state</i>	Encoder state object
<i>with_meta</i>	Whether to include optional continuation and segment request fields (always false currently)
<i>pfd</i>	Pointer to logical PDU file header data

Definition at line 601 of file cf\_codec.c.

References CF\_CFDP\_EncodeAllSegments(), CF\_CFDP\_PduFileData\_RECORD\_CONTINUATION\_STATE, CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH, CF\_Codec\_Store\_uint32(), CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_Logical\_PduFileDataHeader::continuation\_state, FSV, CF\_Logical\_SegmentList::num\_segments, CF\_CFDP\_PduFileDataHeader::offset, CF\_Logical\_PduFileDataHeader::offset, and CF\_Logical\_PduFileDataHeader::segment\_list.

Referenced by CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:



**12.53.4.30 CF\_CFDP\_EncodeFileDirectiveHeader()** void CF\_CFDP\_EncodeFileDirectiveHeader ( CF\_EncoderState\_t \* state,  
CF\_Logical\_PduFileDirectiveHeader\_t \* pfdir )

Encodes a CFDP file directive header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

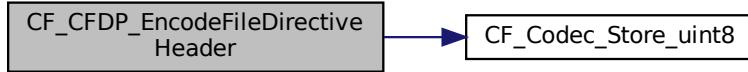
<i>state</i>	Encoder state object
<i>pfdir</i>	Pointer to logical PDU file directive header data

Definition at line 441 of file cf\_codec.c.

References CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_PduFileDirectiveHeader::directive\_code, and CF\_Logical\_PduFileDirectiveHeader::directive\_code.

Referenced by CF\_CFDP\_ConstructPduHeader().

Here is the call graph for this function:



**12.53.4.31 CF\_CFDP\_EncodeFin()** void CF\_CFDP\_EncodeFin ( CF\_EncoderState\_t \* state,  
CF\_Logical\_PduFin\_t \* plfin )

Encodes a CFDP Final (FIN) header block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Note

this encode includes any TLV values which are indicated in the logical data structure

#### Parameters

<i>state</i>	Encoder state object
<i>plfin</i>	Pointer to logical PDU FIN header data

Definition at line 660 of file cf\_codec.c.

References CF\_Logical\_PduFin::cc, CF\_CFDP\_EncodeAllTlv(), CF\_CFDP\_PduFin\_FLAGS\_CC, CF\_CFDP\_PduFin\_FLAGS\_DELIVERY\_CODE, CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS, CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_Logical\_PduFin::delivery\_code, CF\_Logical\_PduFin::file\_status, CF\_CFDP\_PduFin::flags, FSV, and CF\_Logical\_PduFin::tlv\_list.

Referenced by CF\_CFDP\_SendFin().

Here is the call graph for this function:



**12.53.4.32 CF\_CFDP\_EncodeHeaderFinalSize()** void CF\_CFDP\_EncodeHeaderFinalSize ( CF\_EncoderState\_t \* state,  
CF\_Logical\_PduHeader\_t \* plh )

Updates an already-encoded PDU base header block with the final PDU size.

This function encodes the "data\_encoded\_length" field from the logical PDU structure into the encoded header block.  
The PDU will also be closed (set done) to indicate that no more data should be added.

**Note**

Unlike other encode operations, this function does not add any new blocks to the PDU. It only updates the already-encoded block at the beginning of the PDU, which must have been done by a prior call to [CF\\_CFDP\\_EncodeHeaderWithoutSize\(\)](#).

**See also**

[CF\\_CFDP\\_EncodeHeaderWithoutSize\(\)](#) for initially encoding the PDU header block

**Parameters**

<i>state</i>	Encoder state object
<i>plh</i>	Pointer to logical PDU header data

Definition at line 411 of file cf\_codec.c.

References CF\_EncoderState::base, CF\_CODEC\_GET\_POSITION, CF\_CODEC\_IS\_OK, CF\_CODEC\_SET\_DONE, CF\_Codec\_Store\_uint16(), CF\_Logical\_PduHeader::data\_encoded\_length, and CF\_CFDP\_PduHeader::length.

Referenced by CF\_CFDP\_SetPduLength().

Here is the call graph for this function:

**12.53.4.33 CF\_CFDP\_EncodeHeaderWithoutSize()** `void CF_CFDP_EncodeHeaderWithoutSize (`

```
    CF_EncoderState_t * state,
    CF_Logical_PduHeader_t * plh )
```

Encodes a CFDP PDU base header block, bypassing the size field.

On transmit side, the common/base header must be encoded in two parts, to deal with the "total\_size" field. The initial encoding of the basic fields is done as soon as it is known that a PDU of this type needs to be sent, but the total size may not be yet known, as it depends on the remainder of encoding and any additional data that might get added to the variable length sections.

This function encodes all base header fields *except* total length. There is a separate function later to update the total\_length to the correct value once the remainder of encoding is done. Luckily, the total\_length is in the first fixed position binary blob so it is easy to update later.

If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

**See also**

[CF\\_CFDP\\_EncodeHeaderFinalSize\(\)](#) for updating the length field once it is known

**Parameters**

<i>state</i>	Encoder state object
<i>plh</i>	Pointer to logical PDU header data

Definition at line 373 of file cf\_codec.c.

References CF\_CFDP\_PduHeader\_FLAGS\_DIR, CF\_CFDP\_PduHeader\_FLAGS\_MODE, CF\_CFDP\_PduHeader→\_FLAGS\_TYPE, CF\_CFDP\_PduHeader\_FLAGS\_VERSION, CF\_CFDP\_PduHeader\_LENGTHS\_ENTITY, CF→\_CFDP\_PduHeader\_LENGTHS\_TRANSACTION\_SEQUENCE, CF\_CFDP\_PduHeader\_SEGMENT\_METADATA, CF→\_CFDP\_PduHeader\_SEGMENTATION\_CONTROL, CF\_CODEC\_GET\_POSITION, CF\_Codec\_Store\_uint8(), CF→\_ENCODE\_FIXED\_CHUNK, CF\_EncodeIntegerInSize(), CF\_Logical\_PduHeader::destination\_eid, CF\_Logical→\_PduHeader::direction, CF\_Logical\_PduHeader::eid\_length, CF\_CFDP\_PduHeader::eid\_tsn\_lengths, CF\_CFDP→\_PduHeader::flags, FSV, CF\_Logical\_PduHeader::header\_encoded\_length, CF\_Logical\_PduHeader::pdu\_type, CF\_Logical\_PduHeader::segment\_meta\_flag, CF\_Logical\_PduHeader::segmentation\_control, CF\_Logical\_Pdu→\_Header::sequence\_num, CF\_Logical\_PduHeader::source\_eid, CF\_Logical\_PduHeader::txm\_mode, CF\_Logical→\_PduHeader::txn\_seq\_length, and CF\_Logical\_PduHeader::version.

Referenced by CF\_CFDP\_ConstructPduHeader().

Here is the call graph for this function:



**12.53.4.34 CF\_CFDP\_EncodeLV()** void CF\_CFDP\_EncodeLV (   
`CF_EncoderState_t * state,`  
`CF_Logical_Lv_t * pllv )`

Encodes a single CFDP Length+Value (LV) pair.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

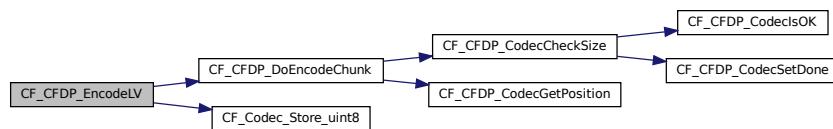
<code>state</code>	Encoder state object
<code>pllv</code>	Pointer to logical PDU LV header data

Definition at line 459 of file cf\_codec.c.

References CF\_CFDP\_DoEncodeChunk(), CF\_CODEC\_SET\_DONE, CF\_Codec\_Store\_uint8(), CF\_ENCODE→\_FIXED\_CHUNK, CF\_Logical\_Lv::data\_ptr, CF\_CFDP\_Lv::length, and CF\_Logical\_Lv::length.

Referenced by CF\_CFDP\_EncodeMd().

Here is the call graph for this function:



**12.53.4.35 CF\_CFDP\_EncodeMd()** void CF\_CFDP\_EncodeMd (   
`CF_EncoderState_t * state,`

```
CF_Logical_PduMd_t * plmd )
```

Encodes a CFDP Metadata (MD) header block.

The data in the logical header will be appended to the encoded PDU at the current position. If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

### Note

this encode includes the LV pairs for source and destination file names, which are logically part of the overall MD block.

## Parameters

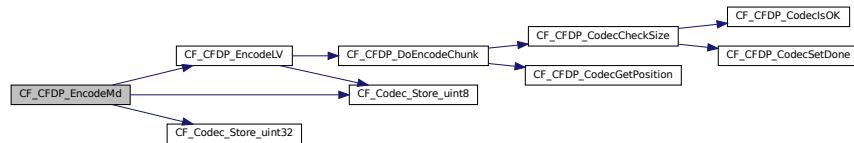
<i>state</i>	Encoder state object
<i>plmd</i>	Pointer to logical PDU metadata header data

Definition at line 577 of file cf\_codec.c.

References CF\_CFDP\_EncodeLV(), CF\_CFDP\_PduMd\_CHECKSUM\_TYPE, CF\_CFDP\_PduMd\_CLOSURE\_REQUESTED, CF\_Codec\_Store\_uint32(), CF\_Codec\_Store\_uint8(), CF\_ENCODE\_FIXED\_CHUNK, CF\_Logical\_PduMd::checksum\_type, CF\_Logical\_PduMd::close\_req, CF\_Logical\_PduMd::dest\_filename, FSV, CF\_CFDP\_PduMd::segmentation\_control, CF\_CFDP\_PduMd::size, CF\_Logical\_PduMd::size, and CF\_Logical\_PduMd::source\_filename.

Referenced by CF CFDP SendMd().

Here is the call graph for this function:



**12.53.4.36 CF\_CFDP\_EncodeNak()** void CF\_CFDP\_EncodeNak (

```
CF_EncoderState_t * state,  
CF_Logical_PduNak_t * plnak )
```

Encodes a CFDP Non-Acknowledge (NAK) header block.

The data in the logical header will be appended to the encoded PDU at the current position. If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

### Note

this encode includes any Segment Request values which are indicated in the logical data structure

## Parameters

<i>state</i>	Encoder state object
<i>plnak</i>	Pointer to logical PDU NAK header data

Definition at line 705 of file cf\_codec.c.

References CF CFDP EncodeAllSegments(), CF Codec Store uint32(), CF ENCODE FIXED CHUNK, CF ↵

CFDP\_PduNak::scope\_end, CF\_Logical\_PduNak::scope\_end, CF\_CFDP\_PduNak::scope\_start, CF\_Logical\_PduNak::scope\_start, and CF\_Logical\_PduNak::segment\_list.

Referenced by CF\_CFDP\_SendNak().

Here is the call graph for this function:



**12.53.4.37 CF\_CFDP\_EncodeSegmentRequest()** `void CF_CFDP_EncodeSegmentRequest (`  
 `CF_EncoderState_t * state,`  
 `CF_Logical_SegmentRequest_t * plseg )`

Encodes a single CFDP Segment Request block.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

#### Parameters

<code>state</code>	Encoder state object
<code>plseg</code>	Pointer to single logical PDU segment request header data

Definition at line 527 of file cf\_codec.c.

References CF\_Codec\_Store\_uint32(), CF\_ENCODE\_FIXED\_CHUNK, CF\_CFDP\_SegmentRequest::offset\_end, CF\_Logical\_SegmentRequest::offset\_end, CF\_CFDP\_SegmentRequest::offset\_start, and CF\_Logical\_SegmentRequest::offset\_start.

Referenced by CF\_CFDP\_EncodeAllSegments().

Here is the call graph for this function:



**12.53.4.38 CF\_CFDP\_EncodeTLV()** `void CF_CFDP_EncodeTLV (`  
 `CF_EncoderState_t * state,`  
 `CF_Logical_Tlv_t * pltlv )`

Encodes a single CFDP Type+Length+Value (TLV) tuple.

The data in the logical header will be appended to the encoded PDU at the current position  
If the encoder is in an error state, nothing is encoded, and the state of the encoder is not changed.

**Note**

Only the CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID TLV type is currently supported by this function, but other TLV types may be added in future versions as needed.

## Parameters

<code>state</code>	Encoder state object
<code>ptlvt</code>	Pointer to single logical PDU TLV header data

Definition at line 489 of file cf\_codec.c.

References `CF_CFDP_DoEncodeChunk()`, `CF_CFDP_TLV_TYPE_ENTITY_ID`, `CF_CODEC_SET_DONE`, `CF_Codec_Store_uint8()`, `CF_ENCODE_FIXED_CHUNK`, `CF_EncodeIntegerInSize()`, `CF_Logical_Tlv::data`, `CF_Logical_TlvData::data_ptr`, `CF_Logical_TlvData::eid`, `CF_CFDP_tlv::length`, `CF_Logical_Tlv::length`, `CF_CFDP_tlv::type`, and `CF_Logical_Tlv::type`.

Referenced by `CF_CFDP_EncodeAllTlv()`.

Here is the call graph for this function:



**12.53.4.39 CF\_CFDP\_GetValueEncodedSize()** `uint8 CF_CFDP_GetValueEncodedSize ( uint64 Value )`

Gets the minimum number of octets that the given integer may be encoded in.

Based on the integer value, this computes the minimum number of bytes that must be allocated to that integer within a CFDP PDU. This is typically used for entity IDs and sequence numbers, where CFDP does not specify a specific size for these items. They may be encoded between 1 and 8 bytes, depending on the actual value is.

## Parameters

<code>Value</code>	Integer value that needs to be encoded
--------------------	--

## Returns

Minimum number of bytes that the value requires (between 1 and 8, inclusive)

Definition at line 329 of file cf\_codec.c.

Referenced by `CF_CFDP_AppendTlv()`, and `CF_CFDP_ConstructPduHeader()`.

**12.53.4.40 CF\_DecodeIntegerInSize()** `uint64 CF_DecodeIntegerInSize ( CF_DecoderState_t * state, uint8 decode_size )`

Decodes an integer value from the specified number of octets.

This decodes an integer value in the specified number of octets. The actual number of octets must be determined using another field in the PDU before calling this function.

## Warning

This function will decode exactly the given number of octets. If this does not match actual encoded size, the return value will be wrong, and it will likely also throw off the decoding of any fields that follow this one.

## See also

[CF\\_EncodeIntegerInSize\(\)](#) for the inverse operation

## Parameters

<i>state</i>	Encoder state object
<i>decode_size</i>	Number of octets that the value is encoded in (between 1 and 8, inclusive)

## Returns

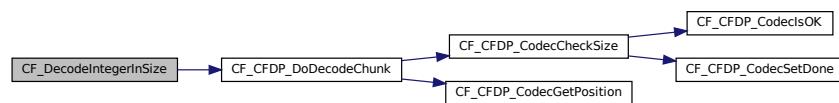
Decoded value

Definition at line 742 of file cf\_codec.c.

References [CF\\_CFDP\\_DoDecodeChunk\(\)](#).

Referenced by [CF\\_CFDP\\_DecodeHeader\(\)](#), and [CF\\_CFDP\\_DecodeTLV\(\)](#).

Here is the call graph for this function:



**12.53.4.41 CF\_EncodeIntegerInSize()** void CF\_EncodeIntegerInSize (   
*CF\_EncoderState\_t* \* *state*,   
*uint64* *value*,   
*uint8* *encode\_size* )

Encodes the given integer value in the given number of octets.

This encodes an integer value in the specified number of octets. Use [CF\\_CFDP\\_GetValueEncodedSize\(\)](#) to determine the minimum number of octets required for a given value. Using more than the minimum is OK, but will consume extra bytes.

## Warning

This function does not error check the encode\_size parameter, and will encode the size given, even if it results in an invalid value. Using fewer octets than the minimum reported by [CF\\_CFDP\\_GetValueEncodedSize\(\)](#) will likely result in incorrect decoding at the receiver.

## See also

[CF\\_DecodeIntegerInSize\(\)](#) for the inverse operation

## Parameters

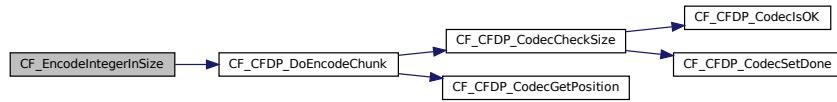
<i>state</i>	Encoder state object
<i>value</i>	Integer value that needs to be encoded
<i>encode_size</i>	Number of octets to encode the value in (between 1 and 8, inclusive)

Definition at line 348 of file cf\_codec.c.

References CF\_CFDP\_DoEncodeChunk().

Referenced by CF\_CFDP\_EncodeHeaderWithoutSize(), and CF\_CFDP\_EncodeTLV().

Here is the call graph for this function:



## 12.54 apps/cf/fsw/src(cf\_crc.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_crc.h"
#include <string.h>
```

### Functions

- void [CF\\_CRC\\_Start \(CF\\_Crc\\_t \\*crc\)](#)  
*Start a CRC streamable digest.*
- void [CF\\_CRC\\_Digest \(CF\\_Crc\\_t \\*crc, const uint8 \\*data, size\\_t len\)](#)  
*Digest a chunk for CRC calculation.*
- void [CF\\_CRC\\_Finalize \(CF\\_Crc\\_t \\*crc\)](#)  
*Finalize a CRC calculation.*

#### 12.54.1 Detailed Description

The CF Application CRC calculation source file

This is a streaming CRC calculator. Data can all be given at once for a result or it can trickle in.

This file is intended to be generic and usable by other apps.

#### 12.54.2 Function Documentation

**12.54.2.1 CF\_CRC\_Digest()** void CF\_CRC\_Digest (

```
    CF_Crc_t * crc,
    const uint8 * data,
    size_t len )
```

Digest a chunk for CRC calculation.

##### Description

Does the CRC calculation, and stores an index into the given 4-byte word in case the input was not evenly divisible for 4.

##### Assumptions, External Events, and Notes:

crc must not be NULL.

**Parameters**

<i>crc</i>	CRC object to operate on
<i>data</i>	Pointer to data to digest
<i>len</i>	Length of data to digest

Definition at line 53 of file cf\_crc.c.

References CF\_Crc::index, CF\_Crc::result, and CF\_Crc::working.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), and CF\_CFDP\_S\_SendFileData().

**12.54.2.2 CF\_CRC\_Finalize()** void CF\_CRC\_Finalize (   
     CF\_Crc\_t \* *crc* )

Finalize a CRC calculation.

**Description**

Checks the index and if it isn't 0, does the final calculations on the bytes in the shift register. After this call is made, the result field of the structure holds the result.

**Assumptions, External Events, and Notes:**

*crc* must not be NULL.

**Parameters**

<i>crc</i>	CRC object to operate on
------------	--------------------------

Definition at line 78 of file cf\_crc.c.

References CF\_Crc::index, CF\_Crc::result, and CF\_Crc::working.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), and CF\_CFDP\_S\_CheckState().

**12.54.2.3 CF\_CRC\_Start()** void CF\_CRC\_Start (   
     CF\_Crc\_t \* *crc* )

Start a CRC streamable digest.

**Assumptions, External Events, and Notes:**

*crc* must not be NULL.

**Parameters**

<i>crc</i>	CRC object to operate on
------------	--------------------------

Definition at line 42 of file cf\_crc.c.

Referenced by CF\_CFDP\_R\_CalcCrcStart(), and CF\_CFDP\_S\_Init().

## 12.55 apps/cf/fsw/src/cf\_crc.h File Reference

```
#include "cfe.h"
```

## Data Structures

- struct [CF\\_Crc](#)  
*CRC state object.*

## Typedefs

- typedef struct [CF\\_Crc CF\\_Crc\\_t](#)  
*CRC state object.*

## Functions

- void [CF\\_CRC\\_Start \(CF\\_Crc\\_t \\*crc\)](#)  
*Start a CRC streamable digest.*
- void [CF\\_CRC\\_Digest \(CF\\_Crc\\_t \\*crc, const uint8 \\*data, size\\_t len\)](#)  
*Digest a chunk for CRC calculation.*
- void [CF\\_CRC\\_Finalize \(CF\\_Crc\\_t \\*crc\)](#)  
*Finalize a CRC calculation.*

### 12.55.1 Detailed Description

The CF Application CRC calculation header file

### 12.55.2 Typedef Documentation

**12.55.2.1 CF\_Crc\_t** [typedef struct CF\\_Crc CF\\_Crc\\_t](#)  
CRC state object.

### 12.55.3 Function Documentation

**12.55.3.1 CF\_CRC\_Digest()** [void CF\\_CRC\\_Digest \(](#)  
[CF\\_Crc\\_t \\* crc,](#)  
[const uint8 \\* data,](#)  
[size\\_t len \)](#)

Digest a chunk for CRC calculation.

#### Description

Does the CRC calculation, and stores an index into the given 4-byte word in case the input was not evenly divisible for 4.

#### Assumptions, External Events, and Notes:

crc must not be NULL.

#### Parameters

<i>crc</i>	CRC object to operate on
<i>data</i>	Pointer to data to digest
<i>len</i>	Length of data to digest

Definition at line 53 of file cf\_crc.c.

References CF\_Crc::index, CF\_Crc::result, and CF\_Crc::working.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), and CF\_CFDP\_S\_SendFileData().

### 12.55.3.2 CF\_CRC\_Finalize() void CF\_CRC\_Finalize (

CF\_Crc\_t \* crc )

Finalize a CRC calculation.

#### Description

Checks the index and if it isn't 0, does the final calculations on the bytes in the shift register. After this call is made, the result field of the structure holds the result.

#### Assumptions, External Events, and Notes:

crc must not be NULL.

#### Parameters

crc	CRC object to operate on
-----	--------------------------

Definition at line 78 of file cf\_crc.c.

References CF\_Crc::index, CF\_Crc::result, and CF\_Crc::working.

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), and CF\_CFDP\_S\_CheckState().

### 12.55.3.3 CF\_CRC\_Start() void CF\_CRC\_Start (

CF\_Crc\_t \* crc )

Start a CRC streamable digest.

#### Assumptions, External Events, and Notes:

crc must not be NULL.

#### Parameters

crc	CRC object to operate on
-----	--------------------------

Definition at line 42 of file cf\_crc.c.

Referenced by CF\_CFDP\_R\_CalcCrcStart(), and CF\_CFDP\_S\_Init().

## 12.56 apps/cf/fsw/src(cf\_dispatch.c File Reference

```
#include "cf_dispatch.h"
#include "cf_app.h"
#include "cf_eventids.h"
#include "cf_cmd.h"
#include "cfe.h"
#include <string.h>
```

## Functions

- void [CF\\_ProcessGroundCommand](#) (const [CFE\\_SB\\_Buffer\\_t](#) \*BufPtr)  
*Process any ground command contained in the given message.*
- void [CF\\_AppPipe](#) (const [CFE\\_SB\\_Buffer\\_t](#) \*BufPtr)  
*CF message processing function.*

### 12.56.1 Detailed Description

The CF Application main application source file

This file contains the functions that initialize the application and link all logic and functionality to the CFS.

### 12.56.2 Function Documentation

**12.56.2.1 CF\_AppPipe()** void CF\_AppPipe (

```
    const CFE_SB_Buffer_t * BufPtr )
```

CF message processing function.

#### Description

Process message packets received via the Software Bus command pipe

#### Assumptions, External Events, and Notes:

BufPtr must not be NULL.

#### Parameters

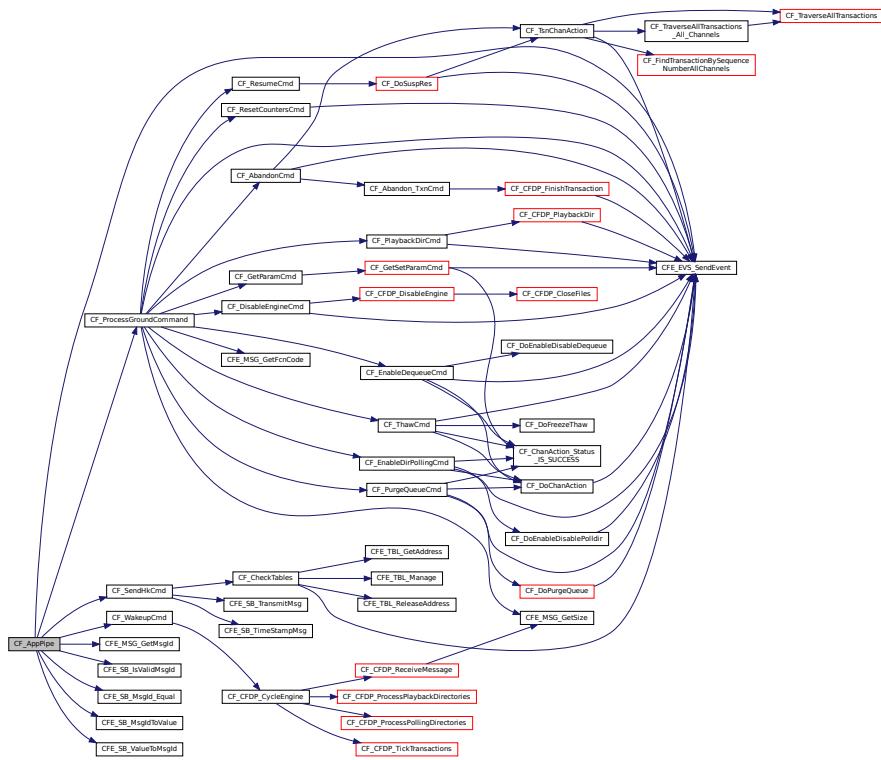
in	<i>BufPtr</i>	Software Bus message pointer
----	---------------	------------------------------

Definition at line 132 of file cf\_dispatch.c.

References CF\_AppData, CF\_CMD\_MID, CF\_MID\_ERR\_EID, CF\_ProcessGroundCommand(), CF\_SEND\_HK\_MID, CF\_SendHkCmd(), CF\_WAKE\_UP\_MID, CF\_WakeupCmd(), CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetMsgId(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_IsValidMsgId(), CFE\_SB\_MsgId\_Equal(), CFE\_SB\_\_MSGID\_RESERVED, CFE\_SB\_MsgIdToValue(), CFE\_SB\_ValueToMsgId(), CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CFE\_SB\_Msg::Msg, and CF\_HkPacket::Payload.

Referenced by CF\_AppMain().

Here is the call graph for this function:



**12.56.2.2 CF\_ProcessGroundCommand()** void CF\_ProcessGroundCommand ( const CFE\_SB\_Buffer\_t \* BufPtr )

Process any ground command contained in the given message.

**Assumptions, External Events, and Notes:**

BuPtr must not be NULL.

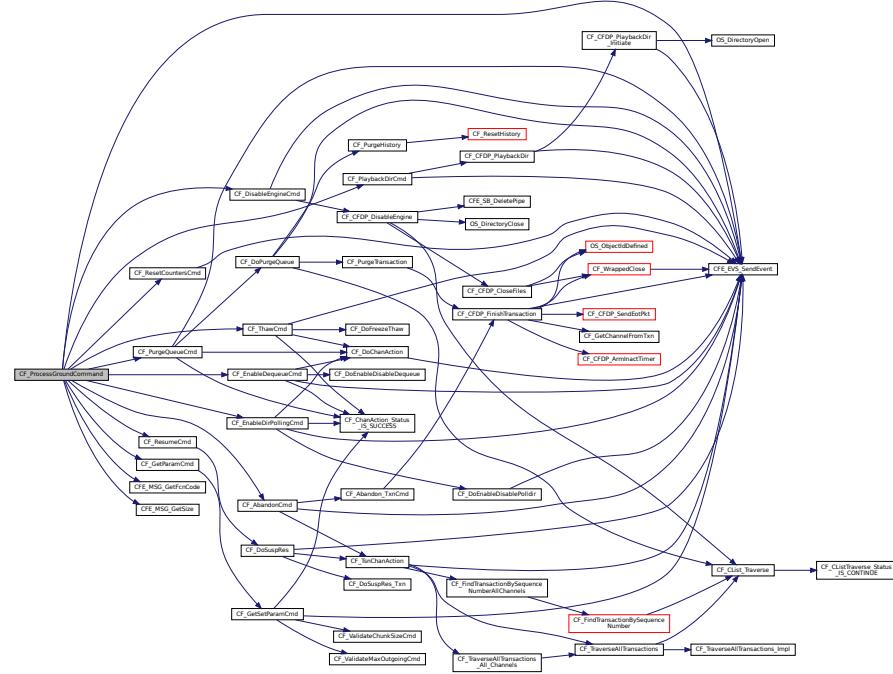
#### Parameters

BufPtr	Pointer to command message
--------	----------------------------

Definition at line 42 of file cf\_dispatch.c.

References CF\_ABANDON\_CC, CF\_AbandonCmd(), CF\_AppData, CF\_CANCEL\_CC, CF\_CC\_ERR\_EID, CF\_CMD\_LEN\_ERR\_EID, CF\_DISABLE\_DEQUEUE\_CC, CF\_DISABLE\_DIR\_POLLING\_CC, CF\_DISABLE\_ENGINE\_CC, CF\_DisableEngineCmd(), CF\_ENABLE\_DEQUEUE\_CC, CF\_ENABLE\_DIR\_POLLING\_CC, CF\_ENABLE\_ENGINE\_CC, CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_FREEZE\_CC, CF\_GET\_PARAM\_CC, CF\_GetParamCmd(), CF\_NOOP\_CC, CF\_PLAYBACK\_DIR\_CC, CF\_PlaybackDirCmd(), CF\_PURGE\_QUEUE\_CC, CF\_PurgeQueueCmd(), CF\_RESET\_CC, CF\_ResetCountersCmd(), CF\_RESUME\_CC, CF\_ResumeCmd(), CF\_SET\_PARAM\_CC, CF\_SUSPEND\_CC, CF\_THAW\_CC, CF\_ThawCmd(), CF\_TX\_FILE\_CC, CF\_WRITE\_QUEUE\_CC, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetSize(), CF\_HkPacket::Payload, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CF\_SB\_Msg::Msg, and CF\_HkPacket::Payload.

Referenced by CF\_AppPipe().  
Here is the call graph for this function:



## 12.57 apps/cf/fsw/src(cf\_dispatch.h File Reference

```
#include "cfe.h"
```

### Functions

- void [CF\\_ProcessGroundCommand](#) (const [CFE\\_SB\\_Buffer\\_t](#) \*BufPtr)  
*Process any ground command contained in the given message.*
- void [CF\\_AppPipe](#) (const [CFE\\_SB\\_Buffer\\_t](#) \*BufPtr)  
*CF message processing function.*

#### 12.57.1 Detailed Description

The CF Application main application header file

#### 12.57.2 Function Documentation

**12.57.2.1 CF\_AppPipe()** void CF\_AppPipe (   
   const [CFE\\_SB\\_Buffer\\_t](#) \* BufPtr )  
CF message processing function.

## Description

Process message packets received via the Software Bus command pipe

## **Assumptions, External Events, and Notes:**

BufPtr must not be NULL.

## Parameters

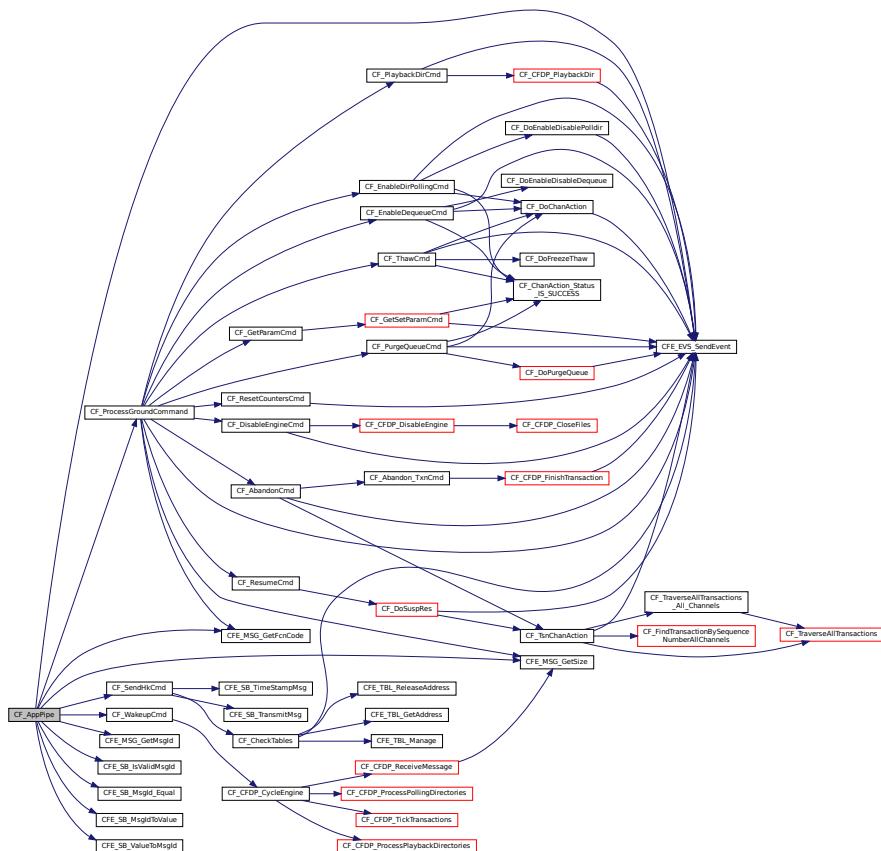
in *BufPtr* Software Bus message pointer

Definition at line 132 of file cf\_dispatch.c.

References CF\_AppData, CF\_CC\_ERR\_EID, CF\_CMD\_LEN\_ERR\_EID, CF\_CMD\_MID, CF\_MID\_ERR\_EID, CF\_ProcessGroundCommand(), CF\_SEND\_HK\_MID, CF\_SendHkCmd(), CF\_TC\_DISPATCH\_TABLE, CF\_WAKE\_UP\_MID, CF\_WakeupCmd(), CFE\_EVT\_EventType\_ERROR, CFE\_EVT\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_MSG.GetSize(), CFE\_SB\_INVALID\_MSG\_ID, CFE\_SB\_IsValidMsgId(), CFE\_SB\_MsgId\_Equal(), CFE\_SB\_MSGID\_RESERVED, CFE\_SB\_MsgIdToValue(), CFE\_SB\_ValueToMsgId(), CFE\_STATUS\_UNKNOWN\_MSG\_ID, CFE\_STATUS\_WRONG\_MSG\_LENGTH, CFE\_SUCCESS, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CFE\_SB\_Msg::Msg, and CF\_HkPacket::Payload.

Referenced by CF\_AppMain().

Here is the call graph for this function:



**12.57.2.2 CF\_ProcessGroundCommand()** void CF\_ProcessGroundCommand (

```
const CFE_SB_Buffer_t * BufPtr )
```

Process any ground command contained in the given message.

## **Assumptions, External Events, and Notes:**

BufPtr must not be NULL.

## Parameters

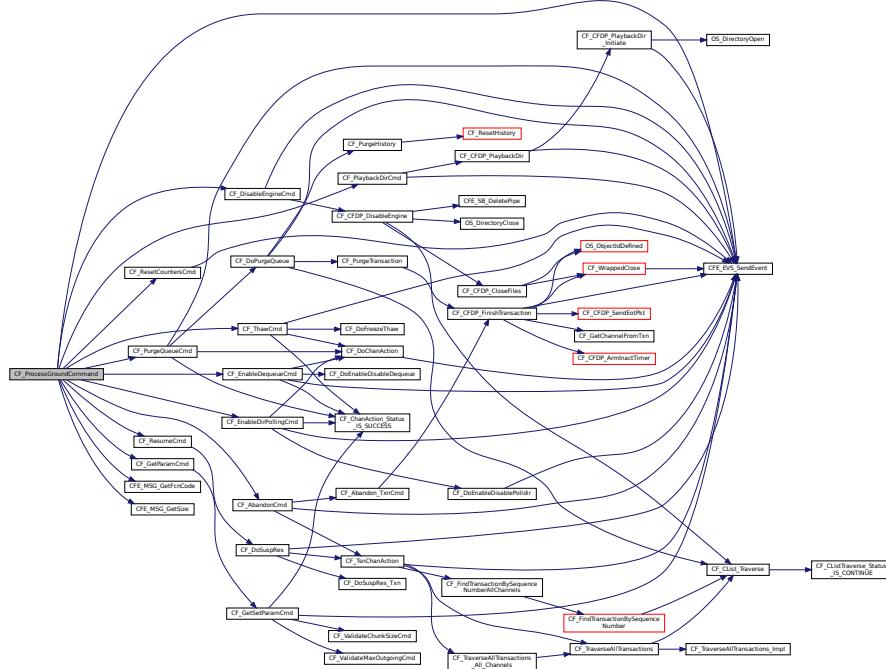
*BufPtr*      Pointer to command message

Definition at line 42 of file cf\_dispatch.c.

References CF\_ABANDON\_CC, CF\_AbandonCmd(), CF\_AppData, CF\_CANCEL\_CC, CF\_CC\_ERR\_EID, CF\_CMD\_LEN\_ERR\_EID, CF\_DISABLE\_DEQUEUE\_CC, CF\_DISABLE\_DIR\_POLLING\_CC, CF\_DISABLE\_ENGINE\_CC, CF\_DisableEngineCmd(), CF\_ENABLE\_DEQUEUE\_CC, CF\_ENABLE\_DIR\_POLLING\_CC, CF\_ENABLE\_ENGINE\_CC, CF\_EnableDequeueCmd(), CF\_EnableDirPollingCmd(), CF\_FREEZE\_CC, CF\_GET\_PARAM\_CC, CF\_GetParamCmd(), CF\_NOOP\_CC, CF\_PLAYBACK\_DIR\_CC, CF\_PlaybackDirCmd(), CF\_PURGE\_QUEUE\_CC, CF\_PurgeQueueCmd(), CF\_RESET\_CC, CF\_ResetCountersCmd(), CF\_RESUME\_CC, CF\_ResumeCmd(), CF\_SET\_PARAM\_CC, CF\_SUSPEND\_CC, CF\_THAW\_CC, CF\_ThawCmd(), CF\_TX\_FILE\_CC, CF\_WRITE\_QUEUE\_CC, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetSize(), CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CFE\_SB\_Msg::Msg, and CF\_HkPacket::Payload.

Referenced by CF\_AppPipe().

Here is the call graph for this function:



## 12.58 apps/cf/fsw/src/cf\_eds\_dispatch.c File Reference

```
#include "cf_app.h"  
#include "cf_eventids.h"
```

```
#include "cf_dispatch.h"
#include "cf_cmd.h"
#include "cf_eds_dictionary.h"
#include "cf_eds_dispatcher.h"
#include "cfe_msg.h"
```

## Functions

- void **CF\_AppPipe** (const **CFE\_SB\_Buffer\_t** \*BufPtr)  
*CF message processing function.*

## Variables

- static const EdsDispatchTable\_EdsComponent\_CF\_Application\_CFE\_SB\_Telecommand\_t **CF\_TC\_DISPATCH\_TABLE**

### 12.58.1 Function Documentation

**12.58.1.1 CF\_AppPipe()** void CF\_AppPipe (
 const **CFE\_SB\_Buffer\_t** \* BufPtr )

CF message processing function.

#### Description

Process message packets received via the Software Bus command pipe

#### Assumptions, External Events, and Notes:

BufPtr must not be NULL.

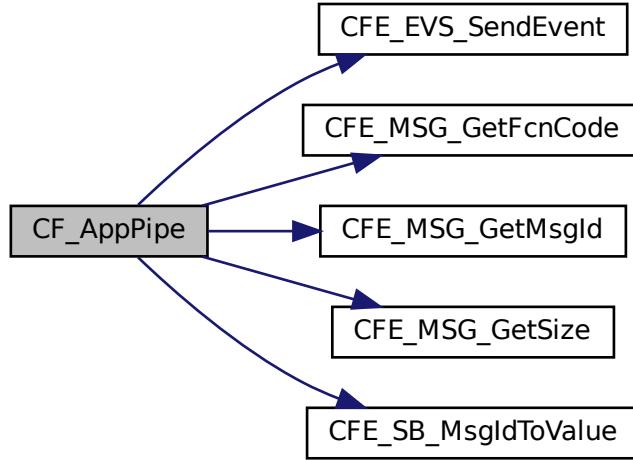
#### Parameters

in	<b>BufPtr</b>	Software Bus message pointer
----	---------------	------------------------------

Definition at line 44 of file cf\_eds\_dispatch.c.

References CF\_AppData, CF\_CC\_ERR\_EID, CF\_CMD\_LEN\_ERR\_EID, CF\_MID\_ERR\_EID, CF\_TC\_DISPATCH\_TABLE, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), CFE\_MSG\_GetFcnCode(), CFE\_MSG\_GetMsgId(), CFE\_MSG\_GetSize(), CFE\_SB\_MsgIdToValue(), CFE\_STATUS\_UNKNOWN\_MSG\_ID, CFE\_STATUS\_WRONG\_MSG\_LENGTH, CFE\_SUCCESS, CF\_HkPacket\_Payload::counters, CF\_HkCmdCounters::err, CF\_AppData\_t::hk, CFE\_SB\_Msg::Msg, and CF\_HkPacket::Payload.

Here is the call graph for this function:



## 12.58.2 Variable Documentation

**12.58.2.1 CF\_TC\_DISPATCH\_TABLE** const EdsDispatchTable\_EdsComponent\_CF\_Application\_CFE\_SB\_<  
Telecommand\_t CF\_TC\_DISPATCH\_TABLE [static]

**Initial value:**

```

= {
    .CMD =
    {
        .AbandonCmd_indication      = CF_AbandonCmd,
        .CancelCmd_indication       = CF_CancelCmd,
        .DisableDequeueCmd_indication = CF_DisableDequeueCmd,
        .DisableDirPollingCmd_indication = CF_DisableDirPollingCmd,
        .DisableEngineCmd_indication = CF_DisableEngineCmd,
        .EnableDequeueCmd_indication = CF_EnableDequeueCmd,
        .EnableDirPollingCmd_indication = CF_EnableDirPollingCmd,
        .EnableEngineCmd_indication = CF_EnableEngineCmd,
        .FreezeCmd_indication       = CF_FreezeCmd,
        .GetParamCmd_indication     = CF_GetParamCmd,
        .NoopCmd_indication         = CF_NoopCmd,
        .PlaybackDirCmd_indication = CF_PlaybackDirCmd,
        .PurgeQueueCmd_indication   = CF_PurgeQueueCmd,
        .ResetCountersCmd_indication = CF_ResetCountersCmd,
        .ResumeCmd_indication       = CF_ResumeCmd,
        .SetParamCmd_indication     = CF_SetParamCmd,
        .SuspendCmd_indication      = CF_SuspendCmd,
        .ThawCmd_indication         = CF_ThawCmd,
        .TxFileCmd_indication       = CF_TxFileCmd,
        .WriteQueueCmd_indication   = CF_WriteQueueCmd,
    },
    .SEND_HK = {.indication = CF_SendHkCmd},
    .WAKE_UP = {.indication = CF_WakeupCmd}}
  
```

Definition at line 11 of file cf\_eds\_dispatch.c.

Referenced by `CF_AppPipe()`.

## 12.59 apps/cf/fsw/src(cf\_logical\_pdu.h File Reference

```
#include "common_types.h"
#include "cf_extern_typedefs.h"
#include "cf_cfdp_pdu.h"
```

### Data Structures

- struct [CF\\_Logical\\_PduHeader](#)  
*Structure representing base CFDP PDU header.*
- struct [CF\\_Logical\\_PduFileDirectiveHeader](#)  
*Structure representing logical File Directive header.*
- struct [CF\\_Logical\\_Lv](#)  
*Structure representing logical LV Object format.*
- union [CF\\_Logical\\_TlvData](#)  
*Union of various data items that may occur in a TLV item.*
- struct [CF\\_Logical\\_Tlv](#)  
*Structure representing logical TLV Object format.*
- struct [CF\\_Logical\\_SegmentRequest](#)  
*Structure representing logical Segment Request data.*
- struct [CF\\_Logical\\_SegmentList](#)
- struct [CF\\_Logical\\_TlvList](#)
- struct [CF\\_Logical\\_PduEof](#)  
*Structure representing logical End of file PDU.*
- struct [CF\\_Logical\\_PduFin](#)  
*Structure representing logical Finished PDU.*
- struct [CF\\_Logical\\_PduAck](#)  
*Structure representing CFDP Acknowledge PDU.*
- struct [CF\\_Logical\\_PduMd](#)  
*Structure representing CFDP Metadata PDU.*
- struct [CF\\_Logical\\_PduNak](#)  
*Structure representing logical Non-Acknowledge PDU.*
- struct [CF\\_Logical\\_PduFileDataHeader](#)
- union [CF\\_Logical\\_IntHeader](#)  
*A union of all possible internal header types in a PDU.*
- struct [CF\\_Logical\\_PduBuffer](#)  
*Encapsulates the entire PDU information.*

### Macros

- #define [CF\\_PDU\\_MAX\\_TLV](#) (4)  
*Maximum number of TLV values in a single PDU.*
- #define [CF\\_PDU\\_MAX\\_SEGMENTS](#) ([CF\\_NAK\\_MAX\\_SEGMENTS](#))  
*Maximum number of segment requests in a single PDU.*

## Typedefs

- `typedef uint32 CF_FileSize_t`  
*Type for logical file size/offset value.*
- `typedef struct CF_Logical_PduHeader CF_Logical_PduHeader_t`  
*Structure representing base CFDP PDU header.*
- `typedef struct CF_Logical_PduFileDirectiveHeader CF_Logical_PduFileDirectiveHeader_t`  
*Structure representing logical File Directive header.*
- `typedef struct CF_Logical_Lv CF_Logical_Lv_t`  
*Structure representing logical LV Object format.*
- `typedef union CF_Logical_TlvData CF_Logical_TlvData_t`  
*Union of various data items that may occur in a TLV item.*
- `typedef struct CF_Logical_Tlv CF_Logical_Tlv_t`  
*Structure representing logical TLV Object format.*
- `typedef struct CF_Logical_SegmentRequest CF_Logical_SegmentRequest_t`  
*Structure representing logical Segment Request data.*
- `typedef struct CF_Logical_SegmentList CF_Logical_SegmentList_t`
- `typedef struct CF_Logical_TlvList CF_Logical_TlvList_t`
- `typedef struct CF_Logical_PduEof CF_Logical_PduEof_t`  
*Structure representing logical End of file PDU.*
- `typedef struct CF_Logical_PduFin CF_Logical_PduFin_t`  
*Structure representing logical Finished PDU.*
- `typedef struct CF_Logical_PduAck CF_Logical_PduAck_t`  
*Structure representing CFDP Acknowledge PDU.*
- `typedef struct CF_Logical_PduMd CF_Logical_PduMd_t`  
*Structure representing CFDP Metadata PDU.*
- `typedef struct CF_Logical_PduNak CF_Logical_PduNak_t`  
*Structure representing logical Non-Acknowledge PDU.*
- `typedef struct CF_Logical_PduFileDataHeader CF_Logical_PduFileDataHeader_t`
- `typedef union CF_Logical_IntHeader CF_Logical_IntHeader_t`  
*A union of all possible internal header types in a PDU.*
- `typedef struct CF_Logical_PduBuffer CF_Logical_PduBuffer_t`  
*Encapsulates the entire PDU information.*

### 12.59.1 Detailed Description

Structures defining logical CFDP PDUs

These are CF-specific data structures that reflect the logical content of the CFDP PDUs defined in [cf\\_cfdp\\_pdu.h](#). Note these are *NOT* intended to reflect the bitwise structures defined in the CCSDS blue book, but rather the values contained within those structures, in a form that can be used by software.

Specifically, this intent differs in the following ways:

- All numeric fields are in native byte order
- All structures are padded/aligned according to native CPU (i.e. not packed)
- All bit-fields are exploded, where each field/group is a separate member
- Variable-size content is normalized, allocated as the maximum possible size

### 12.59.2 Macro Definition Documentation

**12.59.2.1 CF\_PDU\_MAX\_SEGMENTS** #define CF\_PDU\_MAX\_SEGMENTS (CF\_NAK\_MAX\_SEGMENTS)

Maximum number of segment requests in a single PDU.

Sets an upper bound on the logical structures for the most possible segment structures in a single PDU.

Definition at line 66 of file cf\_logical\_pdu.h.

**12.59.2.2 CF\_PDU\_MAX\_TLV** #define CF\_PDU\_MAX\_TLV (4)

Maximum number of TLV values in a single PDU.

This just serves to set an upper bound on the logical structures, to keep things simple. The real limit varies depending on the specific PDU type being processed. This caps the amount of storage memory for the worst case, the actual number present is always part of the run-time state.

Without filestore requests, use of TLV is pretty limited.

Definition at line 58 of file cf\_logical\_pdu.h.

### 12.59.3 Typedef Documentation

**12.59.3.1 CF\_FileSize\_t** typedef uint32 CF\_FileSize\_t

Type for logical file size/offset value.

The CFDP protocol permits use of 64-bit values for file size/offsets Although the CF application only supports 32-bit legacy file size type at this point, the logical structures should use this type in case future support for large files is added.

Definition at line 76 of file cf\_logical\_pdu.h.

**12.59.3.2 CF\_Logical\_IntHeader\_t** typedef union CF\_Logical\_IntHeader CF\_Logical\_IntHeader\_t

A union of all possible internal header types in a PDU.

The specific entry which applies depends on the combination of pdu type and directive code.

**12.59.3.3 CF\_Logical\_Lv\_t** typedef struct CF\_Logical\_Lv CF\_Logical\_Lv\_t

Structure representing logical LV Object format.

These Length + Value pairs used in several CFDP PDU types, typically for storage of strings such as file names.

These are only used for string data (mostly filenames) so the data can refer directly to the encoded bits, it does not necessarily need to be duplicated here.

**12.59.3.4 CF\_Logical\_PduAck\_t** typedef struct CF\_Logical\_PduAck CF\_Logical\_PduAck\_t

Structure representing CFDP Acknowledge PDU.

Defined per section 5.2.4 / table 5-8 of CCSDS 727.0-B-5

**12.59.3.5 CF\_Logical\_PduBuffer\_t** typedef struct CF\_Logical\_PduBuffer CF\_Logical\_PduBuffer\_t

Encapsulates the entire PDU information.

**12.59.3.6 CF\_Logical\_PduEof\_t** typedef struct CF\_Logical\_PduEof CF\_Logical\_PduEof\_t

Structure representing logical End of file PDU.

See also

[CF\\_CFDP\\_PduEof\\_t](#) for encoded form

**12.59.3.7 CF\_Logical\_PduFileDataHeader\_t** typedef struct CF\_Logical\_PduFileDataHeader CF\_Logical\_PduFileDataHeader\_t

**12.59.3.8 CF\_Logical\_PduFileDirectiveHeader\_t** `typedef struct CF_Logical_PduFileDirectiveHeader CF_Logical_PduFileDirectiveHeader_t`

Structure representing logical File Directive header.

This contains the file directive code from the PDUs for which it applies. The codes are mapped directly to the CFDP protocol values, but converted to a native value (enum) for direct use by software.

**12.59.3.9 CF\_Logical\_PduFin\_t** `typedef struct CF_Logical_PduFin CF_Logical_PduFin_t`

Structure representing logical Finished PDU.

See also

[CF\\_CFDP\\_PduFin\\_t](#) for encoded form

**12.59.3.10 CF\_Logical\_PduHeader\_t** `typedef struct CF_Logical_PduHeader CF_Logical_PduHeader_t`

Structure representing base CFDP PDU header.

Reflects the common content at the beginning of all CFDP PDUs, of all types.

See also

[CF\\_CFDP\\_PduHeader\\_t](#) for encoded form

**12.59.3.11 CF\_Logical\_PduMd\_t** `typedef struct CF_Logical_PduMd CF_Logical_PduMd_t`

Structure representing CFDP Metadata PDU.

Defined per section 5.2.5 / table 5-9 of CCSDS 727.0-B-5

**12.59.3.12 CF\_Logical\_PduNak\_t** `typedef struct CF_Logical_PduNak CF_Logical_PduNak_t`

Structure representing logical Non-Acknowledge PDU.

**12.59.3.13 CF\_Logical\_SegmentList\_t** `typedef struct CF_Logical_SegmentList CF_Logical_SegmentList_t`**12.59.3.14 CF\_Logical\_SegmentRequest\_t** `typedef struct CF_Logical_SegmentRequest CF_Logical_SegmentRequest_t`  
Structure representing logical Segment Request data.**12.59.3.15 CF\_Logical\_Tlv\_t** `typedef struct CF_Logical_Tlv CF_Logical_Tlv_t`

Structure representing logical TLV Object format.

In the current implementation of CF, only entity IDs are currently encoded in this form where indicated in the spec. This may change in a future version.

See also

[CF\\_CFDP\\_tlv\\_t](#) for encoded form

**12.59.3.16 CF\_Logical\_TlvData\_t** `typedef union CF_Logical_TlvData CF_Logical_TlvData_t`

Union of various data items that may occur in a TLV item.

The actual type is identified by the "type" field in the enclosing TLV

Currently filestore requests are not implemented in CF, so the TLV use is limited. This may change in the future.

Numeric data needs to actually be copied to this buffer, because it needs to be normalized in length and byte-order. But string data (e.g. filenames, messages) can reside in the original encoded form.

### 12.59.3.17 CF\_Logical\_TlvList\_t `typedef struct CF_Logical_TlvList CF_Logical_TlvList_t`

## 12.60 apps/cf/fsw/src(cf\_timer.c File Reference

```
#include "cfe.h"
#include "cf_verify.h"
#include "cf_timer.h"
#include "cf_app.h"
#include "cf_assert.h"
```

### Functions

- `uint32 CF_Timer_Sec2Ticks (CF_Timer_Seconds_t sec)`  
*Converts seconds into scheduler ticks.*
- `void CF_Timer_InitRelSec (CF_Timer_t *txn, uint32 rel_sec)`  
*Initialize a timer with a relative number of seconds.*
- `bool CF_Timer_Expired (const CF_Timer_t *txn)`  
*Check if a timer has expired.*
- `void CF_Timer_Tick (CF_Timer_t *txn)`  
*Notify a timer object a tick has occurred.*

### 12.60.1 Detailed Description

The CF Application timer source file

A timer in CF is really just a structure that holds a counter that indicates the timer expired when it reaches 0. The goal is that any timer is driven by the scheduler ticks. There is no reason we need any finer grained resolution than this for CF.

### 12.60.2 Function Documentation

#### 12.60.2.1 CF\_Timer\_Expired() `bool CF_Timer_Expired (`

`const CF_Timer_t * txn )`

Check if a timer has expired.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<code>txn</code>	Timer object to check
------------------	-----------------------

#### Returns

status code indicating whether timer has expired

#### Return values

<code>1</code>	if expired
<code>0</code>	if not expired

Definition at line 65 of file cf\_timer.c.

References CF\_Timer::tick.

Referenced by CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S\_AckTimerTick(), and CF\_CFDP\_S\_Tick().

**12.60.2.2 CF\_Timer\_InitRelSec()** void CF\_Timer\_InitRelSec (

```
CF_Timer_t * txn,
CF_Timer_Seconds_t rel_sec )
```

Initialize a timer with a relative number of seconds.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

<i>txn</i>	Timer object to initialize
<i>rel_sec</i>	Relative number of seconds

Definition at line 54 of file cf\_timer.c.

References CF\_Timer\_Sec2Ticks(), and CF\_Timer::tick.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_ArmInactTimer(), and CF\_CFDP\_ProcessPollingDirectories().

Here is the call graph for this function:



**12.60.2.3 CF\_Timer\_Sec2Ticks()** uint32 CF\_Timer\_Sec2Ticks (

```
CF_Timer_Seconds_t sec )
```

Converts seconds into scheduler ticks.

**Assumptions, External Events, and Notes:**

sub-second resolution is not required

**Parameters**

<i>sec</i>	Number of seconds
------------	-------------------

**Returns**

Number of ticks for the given seconds.

Definition at line 43 of file cf\_timer.c.

References CF\_AppData, CF\_AppData\_t::config\_table, and CF\_ConfigTable::ticks\_per\_second.

Referenced by CF\_Timer\_InitRelSec().

**12.60.2.4 CF\_Timer\_Tick()** `void CF_Timer_Tick (`  
`CF_Timer_t * txn )`

Notify a timer object a tick has occurred.

**Assumptions, External Events, and Notes:**

`txn` must not be NULL.

**Parameters**

<code>txn</code>	Timer object to tick
------------------	----------------------

Definition at line 76 of file cf\_timer.c.

References CF\_ASSERT, and CF\_Timer::tick.

Referenced by CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S\_AckTimerTick(), and CF\_CFDP\_S\_Tick().

**12.61 apps/cf/fsw/src(cf\_timer.h File Reference**

```
#include "cfe.h"
```

**Data Structures**

- struct [CF\\_Timer](#)

*Basic CF timer object.*

**Typedefs**

- `typedef uint32 CF_Timer_Ticks_t`  
*Type for a timer tick count.*
- `typedef uint32 CF_Timer_Seconds_t`  
*Type for a timer number of seconds.*
- `typedef struct CF_Timer CF_Timer_t`  
*Basic CF timer object.*

**Functions**

- `void CF_Timer_InitRelSec (CF_Timer_t *txn, CF_Timer_Seconds_t rel_sec)`  
*Initialize a timer with a relative number of seconds.*
- `bool CF_Timer_Expired (const CF_Timer_t *txn)`  
*Check if a timer has expired.*
- `void CF_Timer_Tick (CF_Timer_t *txn)`  
*Notify a timer object a tick has occurred.*

- `uint32 CF_Timer_Sec2Ticks (CF_Timer_Seconds_t sec)`  
*Converts seconds into scheduler ticks.*

### 12.61.1 Detailed Description

The CF Application timer header file

### 12.61.2 Typedef Documentation

#### 12.61.2.1 `CF_Timer_Seconds_t` `typedef uint32 CF_Timer_Seconds_t`

Type for a timer number of seconds.

Definition at line 43 of file cf\_timer.h.

#### 12.61.2.2 `CF_Timer_t` `typedef struct CF_Timer CF_Timer_t`

Basic CF timer object.

#### 12.61.2.3 `CF_Timer_Ticks_t` `typedef uint32 CF_Timer_Ticks_t`

Type for a timer tick count.

#### Note

We expect ticks to be 100/sec, so using uint32 for sec could have a bounds condition with uint32. But, we don't expect to use more than 400,000,000 seconds for any reason so let's just live with it.

Definition at line 38 of file cf\_timer.h.

### 12.61.3 Function Documentation

#### 12.61.3.1 `CF_Timer_Expired()` `bool CF_Timer_Expired (` `const CF_Timer_t * txn )`

Check if a timer has expired.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

#### Parameters

<code>txn</code>	Timer object to check
------------------	-----------------------

#### Returns

status code indicating whether timer has expired

#### Return values

<code>1</code>	if expired
<code>0</code>	if not expired

Definition at line 65 of file cf\_timer.c.

References CF\_Timer::tick.

Referenced by CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S\_AckTimerTick(), and CF\_CFDP\_S\_Tick().

**12.61.3.2 CF\_Timer\_InitRelSec()** void CF\_Timer\_InitRelSec (

<code>CF_Timer_t * txn,</code>
<code>CF_Timer_Seconds_t rel_sec )</code>

Initialize a timer with a relative number of seconds.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

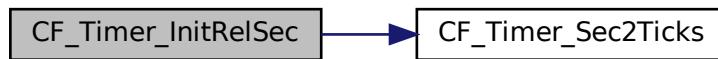
<code>txn</code>	Timer object to initialize
<code>rel_sec</code>	Relative number of seconds

Definition at line 54 of file cf\_timer.c.

References CF\_Timer\_Sec2Ticks(), and CF\_Timer::tick.

Referenced by CF\_CFDP\_ArmAckTimer(), CF\_CFDP\_ArmInactTimer(), and CF\_CFDP\_ProcessPollingDirectories().

Here is the call graph for this function:



**12.61.3.3 CF\_Timer\_Sec2Ticks()** uint32 CF\_Timer\_Sec2Ticks (

<code>CF_Timer_Seconds_t sec )</code>
---------------------------------------

Converts seconds into scheduler ticks.

**Assumptions, External Events, and Notes:**

sub-second resolution is not required

**Parameters**

<code>sec</code>	Number of seconds
------------------	-------------------

**Returns**

Number of ticks for the given seconds.

Definition at line 43 of file cf\_timer.c.

References CF\_AppData, CF\_AppData\_t::config\_table, and CF\_ConfigTable::ticks\_per\_second.

Referenced by CF\_Timer\_InitRelSec().

**12.61.3.4 CF\_Timer\_Tick()** `void CF_Timer_Tick (`  
`CF_Timer_t * txn )`

Notify a timer object a tick has occurred.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

<code>txn</code>	Timer object to tick
------------------	----------------------

Definition at line 76 of file cf\_timer.c.

References CF\_ASSERT, and CF\_Timer::tick.

Referenced by CF\_CFDP\_ProcessPollingDirectories(), CF\_CFDP\_R\_AckTimerTick(), CF\_CFDP\_R\_Tick(), CF\_CFDP\_S\_AckTimerTick(), and CF\_CFDP\_S\_Tick().

## 12.62 apps(cf/fsw/src/cf\_utils.c File Reference

```
#include "cf_app.h"
#include "cf_verify.h"
#include "cf_cfdp.h"
#include "cf_utils.h"
#include "cf_eventids.h"
#include "cf_perfids.h"
#include "cf_assert.h"
```

**Functions**

- **CF\_Channel\_t \* CF\_GetChannelFromTxn (CF\_Transaction\_t \*txn)**  
*Gets the associated channel struct from a transaction.*
- **CF\_CListNode\_t \*\* CF\_GetChunkListHead (CF\_Channel\_t \*chan, uint8 direction)**  
*Gets the head of the chunk list for the given channel + direction.*
- **CF\_CFDP\_AckTxnStatus\_t CF\_CFDP\_GetAckTxnStatus (CF\_Transaction\_t \*txn)**  
*Gets the status of this transaction.*
- **CF\_Transaction\_t \* CF\_FindUnusedTransaction (CF\_Channel\_t \*chan, CF\_Direction\_t direction)**  
*Find an unused transaction on a channel.*
- **void CF\_ResetHistory (CF\_Channel\_t \*chan, CF\_History\_t \*history)**  
*Returns a history structure back to its unused state.*
- **void CF\_FreeTransaction (CF\_Transaction\_t \*txn, uint8 chan)**  
*Frees and resets a transaction and returns it for later use.*
- **CFE\_Status\_t CF\_FindTransactionBySequenceNumber\_Impl (CF\_CListNode\_t \*node, CF\_Traverse\_TransSeqArg\_t \*context)**

*List traversal function to check if the desired sequence number matches.*

- `CF_Transaction_t * CF_FindTransactionBySequenceNumber (CF_Channel_t *chan, CF_TransactionSeq_t transaction_sequence_number, CF_EntityId_t src_eid)`

*Finds an active transaction by sequence number.*
- `CFE_Status_t CF_WriteHistoryEntryToFile (osal_id_t fd, const CF_History_t *history)`

*Write a single history to a file.*
- `CF_CListTraverse_Status_t CF_Traverse_WriteHistoryQueueEntryToFile (CF_CListNode_t *node, void *arg)`

*Writes a human readable representation of a history queue entry to a file.*
- `CF_CListTraverse_Status_t CF_Traverse_WriteTxnQueueEntryToFile (CF_CListNode_t *node, void *arg)`

*Writes a human readable representation of a transaction history entry to a file.*
- `CFE_Status_t CF_WriteTxnQueueDataToFile (osal_id_t fd, CF_Channel_t *chan, CF_QueueIdx_t queue)`

*Write a transaction-based queue's transaction history to a file.*
- `CFE_Status_t CF_WriteHistoryQueueDataToFile (osal_id_t fd, CF_Channel_t *chan, CF_Direction_t dir)`

*Write a history-based queue's entries to a file.*
- `CF_CListTraverse_Status_t CF_PrioSearch (CF_CListNode_t *node, void *context)`

*Searches for the first transaction with a lower priority than given.*
- `void CF_InsertSortPrio (CF_Transaction_t *tx, CF_QueueIdx_t queue)`

*Insert a transaction into a priority sorted transaction queue.*
- `CF_CListTraverse_Status_t CF_TraverseAllTransactions_Impl (CF_CListNode_t *node, void *arg)`

*List traversal function performs operation on every active transaction.*
- `int32 CF_TraverseAllTransactions (CF_Channel_t *chan, CF_TraverseAllTransactions_fn_t fn, void *context)`

*Traverses all transactions on all active queues and performs an operation on them.*
- `int32 CF_TraverseAllTransactions_All_Channels (CF_TraverseAllTransactions_fn_t fn, void *context)`

*Traverses all transactions on all channels and performs an operation on them.*
- `CFE_Status_t CF_WrappedOpenCreate (osal_id_t *fd, const char *fname, int32 flags, int32 access)`

*Wrap the filesystem open call with a perf counter.*
- `void CF_WrappedClose (osal_id_t fd)`

*Wrap the filesystem close call with a perf counter.*
- `CFE_Status_t CF_WrappedRead (osal_id_t fd, void *buf, size_t read_size)`

*Wrap the filesystem read call with a perf counter.*
- `CFE_Status_t CF_WrappedWrite (osal_id_t fd, const void *buf, size_t write_size)`

*Wrap the filesystem write call with a perf counter.*
- `CFE_Status_t CF_WrappedLseek (osal_id_t fd, off_t offset, int mode)`

*Wrap the filesystem lseek call with a perf counter.*
- `CF_CFDP_ConditionCode_t CF_TxnStatus_To_ConditionCode (CF_TxnStatus_t txn_stat)`

*Converts the internal transaction status to a CFDP condition code.*
- `CF_TxnStatus_t CF_TxnStatus_From_ConditionCode (CF_CFDP_ConditionCode_t cc)`

*Converts a CFDP condition code to an internal transaction status.*

### 12.62.1 Detailed Description

The CF Application general utility functions source file  
 Various odds and ends are put here.

### 12.62.2 Function Documentation

**12.62.2.1 CF\_CFDP\_GetAckTxnStatus()** `CF_CFDP_AckTxnStatus_t` CF\_CFDP\_GetAckTxnStatus ( `CF_Transaction_t * txn` )

Gets the status of this transaction.

Determines if the transaction is ACTIVE or TERMINATED. (By definition if it has a txn object then it is not UNRECOGNIZED)

#### Parameters

<code>txn</code>	Transaction
------------------	-------------

#### Returns

`CF_CFDP_AckTxnStatus_t` value corresponding to transaction

Definition at line 87 of file cf\_utils.c.

References `CF_CFDP_AckTxnStatus_ACTIVE`, `CF_CFDP_AckTxnStatus_INVALID`, `CF_CFDP_AckTxnStatus_TERMINATED`, `CF_CFDP_AckTxnStatus_UNRECOGNIZED`, `CF_TxnState_DROP`, `CF_TxnState_HOLD`, `CF_TxnState_R1`, `CF_TxnState_R2`, `CF_TxnState_S1`, `CF_TxnState_S2`, and `CF_Transaction::state`.

Referenced by `CF_CFDP_ArmlnactTimer()`, `CF_CFDP_R_Tick()`, `CF_CFDP_S_Tick()`, and `CF_CFDP_SendAck()`.

**12.62.2.2 CF\_FindTransactionBySequenceNumber()** `CF_Transaction_t*` CF\_FindTransactionBySequenceNumber (

```
    CF_Channel_t * chan,
    CF_TransactionSeq_t transaction_sequence_number,
    CF_EntityId_t src_eid )
```

Finds an active transaction by sequence number.

#### Description

This function traverses the active rx, pending, txa, and txw transaction and looks for the requested transaction.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

#### Parameters

<code>chan</code>	Pointer to the CF channel
<code>transaction_sequence_number</code>	Sequence number to find
<code>src_eid</code>	Entity ID associated with sequence number

#### Returns

Pointer to the given transaction if found

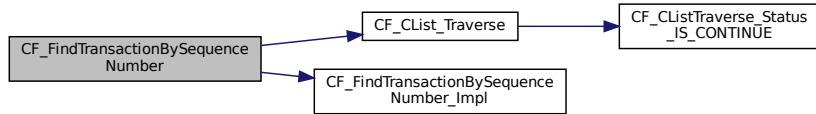
#### Return values

<code>NULL</code>	if the transaction is not found
-------------------	---------------------------------

Definition at line 223 of file cf\_utils.c.

References `CF_CList_Traverse()`, `CF_FindTransactionBySequenceNumber_Impl()`, `CF_QueueIdx_PEND`, `CF_`

Queueldx\_RX, CF\_Queueldx\_TX, CF\_Channel::qs, and CF\_Traverse\_TransSeqArg::txn.  
Referenced by CF\_CFDP\_ReceivePdu(), and CF\_FindTransactionBySequenceNumberAllChannels().  
Here is the call graph for this function:



### 12.62.2.3 CF\_FindTransactionBySequenceNumber\_Impl()

`CFE_Status_t CF_FindTransactionBySequenceNumber_Impl (`

`CF_CListNode_t * node,`  
`CF_Traverse_TransSeqArg_t * context )`

List traversal function to check if the desired sequence number matches.

#### Assumptions, External Events, and Notes:

context must not be NULL. node must not be NULL.

#### Parameters

<code>node</code>	Pointer to node currently being traversed
<code>context</code>	Pointer to state object passed through from initial call

#### Return values

<code>1</code>	when it's found, which terminates list traversal
<code>0</code>	when it isn't found, which causes list traversal to continue

Definition at line 203 of file cf\_utils.c.

References container\_of, CF\_Transaction::history, CF\_History::seq\_num, CF\_History::src\_eid, CF\_Traverse\_TransSeqArg::src\_eid, CF\_Traverse\_TransSeqArg::transaction\_sequence\_number, and CF\_Traverse\_TransSeqArg::txn.  
Referenced by CF\_FindTransactionBySequenceNumber().

### 12.62.2.4 CF\_FindUnusedTransaction()

`CF_Transaction_t* CF_FindUnusedTransaction (`

`CF_Channel_t * chan,`  
`CF_Direction_t direction )`

Find an unused transaction on a channel.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

**Parameters**

<i>chan</i>	Pointer to the CF channel
<i>direction</i>	Intended direction of data flow (TX or RX)

**Returns**

Pointer to a free transaction

**Return values**

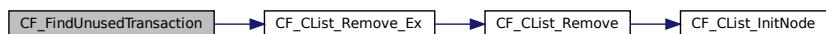
<i>NULL</i>	if no free transactions available.
-------------	------------------------------------

Definition at line 127 of file cf\_utils.c.

References CF\_ASSERT, CF\_CList\_Remove\_Ex(), CF\_QueueIdx\_FREE, CF\_QueueIdx\_HIST, CF\_QueueIdx\_HIST\_FREE, CF\_TxnState\_INIT, CF\_History::cl\_node, CF\_Transaction::cl\_node, container\_of, CF\_History::dir, CF\_Transaction::history, CF\_Channel::qs, and CF\_Transaction::state.

Referenced by CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_StartRxTransaction(), and CF\_CFDP\_TxFile().

Here is the call graph for this function:



**12.62.2.5 CF\_FreeTransaction()** void CF\_FreeTransaction (   
     CF\_Transaction\_t \* *txn*,  
     uint8 *chan* )

Frees and resets a transaction and returns it for later use.

**Assumptions, External Events, and Notes:**

*txn* must not be NULL.

**Parameters**

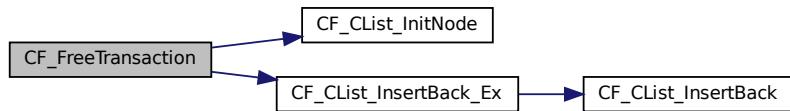
<i>txn</i>	Pointer to the transaction object
<i>chan</i>	The channel number which this transaction is associated with

Definition at line 189 of file cf\_utils.c.

References CF\_AppData, CF\_CList\_InitNode(), CF\_CList\_InsertBack\_Ex(), CF\_QueueIdx\_FREE, CF\_Transaction::chan\_num, CF\_Engine::channels, CF\_Transaction::cl\_node, and CF\_AppData\_t::engine.

Referenced by CF\_CFDP\_InitEngine(), and CF\_CFDP\_RecycleTransaction().

Here is the call graph for this function:



**12.62.2.6 CF\_GetChannelFromTxn()** `CF_Channel_t* CF_GetChannelFromTxn ( CF_Transaction_t * txn )`

Gets the associated channel struct from a transaction.

#### Assumptions, External Events, and Notes:

txn must not be null, and the chan\_num must be set

#### Parameters

<code>txn</code>	Transaction
------------------	-------------

#### Returns

Pointer to CF\_Channel\_t struct associated with the transaction

#### Return values

<code>NULL</code>	if checks failed
-------------------	------------------

Definition at line 43 of file cf\_utils.c.

References CF\_AppData, CF\_NUM\_CHANNELS, CF\_Transaction::chan\_num, CF\_Engine::channels, and CF\_AppData\_t::engine.

Referenced by CF\_CFDP\_AllocChunkList(), CF\_CFDP\_CompleteTick(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_RecycleTransaction(), and CF\_CFDP\_S\_Tick\_NewData().

**12.62.2.7 CF\_GetChunkListHead()** `CF_CListNode_t** CF_GetChunkListHead ( CF_Channel_t * chan, uint8 direction )`

Gets the head of the chunk list for the given channel + direction.

The chunk list contains structs that are available for tracking the chunks associated with files in transit. An entry needs to be pulled from this list for every transaction, and returned to this list when the transaction completes.

#### Parameters

<code>chan</code>	Pointer to channel struct
<code>direction</code>	Whether this is TX or RX

**Returns**

Pointer to list head

Definition at line 65 of file cf\_utils.c.

References CF\_Direction\_NUM, and CF\_Channel::cs.

Referenced by CF\_CFDP\_FindUnusedChunks(), CF\_CFDP\_InitEngine(), and CF\_CFDP\_RecycleTransaction().

```
12.62.2.8 CF_InsertSortPrio() void CF_InsertSortPrio (
    CF_Transaction_t * txn,
    CF_QueueIdx_t queue )
```

Insert a transaction into a priority sorted transaction queue.

**Description**

This function works by walking the queue in reverse to find a transaction with a higher priority than the given transaction. The given transaction is then inserted after that one, since it would be the next lower priority.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

**Parameters**

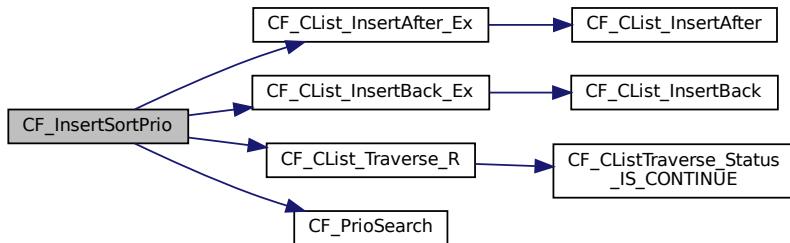
<i>txn</i>	Pointer to the transaction object
<i>queue</i>	Index of queue to insert into

Definition at line 415 of file cf\_utils.c.

References CF\_AppData, CF\_Assert, CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Traverse\_R(), CF\_NUM\_CHANNELS, CF\_PrioSearch(), CF\_Transaction::chan\_num, CF\_Engine::channels, CF\_Transaction::cl\_node, CF\_StateFlags::com, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_Transaction::priority, CF\_Flags::Common::q\_index, CF\_Channel::qs, and CF\_Traverse\_PriorityArg::txn.

Referenced by CF\_CFDP\_SetupTxTransaction(), and CF\_CFDP\_TxFile\_Initiate().

Here is the call graph for this function:



```
12.62.2.9 CF_PrioSearch() CF_CListTraverse_Status_t CF_PrioSearch (
    CF_CListNode_t * node,
    void * context )
```

Searches for the first transaction with a lower priority than given.

**Assumptions, External Events, and Notes:**

node must not be NULL. context must not be NULL.

**Parameters**

<i>node</i>	Node being currently traversed
<i>context</i>	Pointer to CF_Traverse_PriorityArg_t object indicating the priority to search for

**Return values**

<i>CF_CLIST_EXIT</i>	when it's found, which terminates list traversal
<i>CF_CLIST_CONT</i>	when it isn't found, which causes list traversal to continue

Definition at line 391 of file cf\_utils.c.

References CF\_CLIST\_CONT, CF\_CLIST\_EXIT, container\_of, CF\_Transaction::priority, CF\_Traverse\_PriorityArg::priority, and CF\_Traverse\_PriorityArg::txn.

Referenced by CF\_InsertSortPrio().

```
12.62.2.10 CF_ResetHistory() void CF_ResetHistory (
    CF_Channel_t * chan,
    CF_History_t * history )
```

Returns a history structure back to its unused state.

**Description**

There's nothing to do currently other than remove the history from its current queue and put it back on CF\_QueueIdx\_HIST\_FREE.

**Assumptions, External Events, and Notes:**

chan must not be NULL. history must not be NULL.

**Parameters**

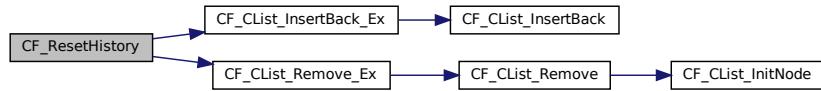
<i>chan</i>	Pointer to the CF channel
<i>history</i>	Pointer to the history entry

Definition at line 177 of file cf\_utils.c.

References CF\_CList\_InsertBack\_Ex(), CF\_CList\_Remove\_Ex(), CF\_QueueIdx\_HIST, CF\_QueueIdx\_HIST\_FREE, and CF\_History::cl\_node.

Referenced by CF\_PurgeHistory().

Here is the call graph for this function:



**12.62.2.11 CF\_Traverse\_WriteHistoryQueueEntryToFile()** `CF_CListTraverse_Status_t CF_Traverse_WriteHistoryQueueEntryToFile (`

`CF_CListNode_t * node,`  
`void * arg )`

Writes a human readable representation of a history queue entry to a file.

This function is a wrapper around `CF_WriteHistoryEntryToFile()` that can be used with `CF_Traverse()` to write history queue entries to the file.

This also implements a direction filter, so only matching entries are actually written to the file.

#### Assumptions, External Events, and Notes:

node must not be NULL. arg must not be NULL.

#### Parameters

<code>node</code>	Node being currently traversed
<code>arg</code>	Pointer to <code>CF_Traverse_WriteHistoryFileArg_t</code> indicating the file information

#### Return values

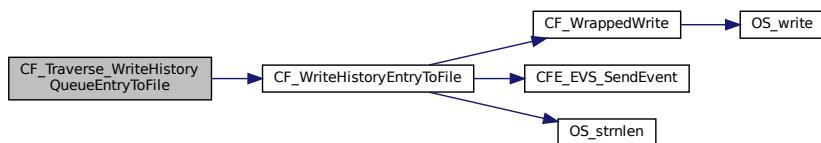
<code>CF_CLIST_CONT</code>	if everything is going well
<code>CF_CLIST_EXIT</code>	if a write error occurred, which means traversal should stop

Definition at line 305 of file cf\_utils.c.

References `CF_CLIST_CONT`, `CF_CLIST_EXIT`, `CF_Direction_NUM`, `CF_WriteHistoryEntryToFile()`, `container_of`, `CF_Traverse_WriteHistoryFileArg::counter`, `CF_History::dir`, `CF_Traverse_WriteHistoryFileArg::error`, `CF_Traverse_WriteHistoryFileArg::fd`, and `CF_Traverse_WriteHistoryFileArg::filter_dir`.

Referenced by `CF_WriteHistoryQueueDataToFile()`.

Here is the call graph for this function:



```
12.62.2.12 CF_Traverse_WriteTxnQueueEntryToFile() CF_CListTraverse_Status_t CF_Traverse_WriteTxn->
QueueEntryToFile (
    CF_CListNode_t * node,
    void * arg )
```

Writes a human readable representation of a transaction history entry to a file.

This function is a wrapper around [CF\\_WriteHistoryEntryToFile\(\)](#) that can be used with [CF\\_Traverse\(\)](#) to write transaction queue entries to the file.

#### Assumptions, External Events, and Notes:

node must not be NULL. arg must not be NULL.

#### Parameters

<i>node</i>	Node being currently traversed
<i>arg</i>	Pointer to <a href="#">CF_Traverse_WriteTxnFileArg_t</a> indicating the file information

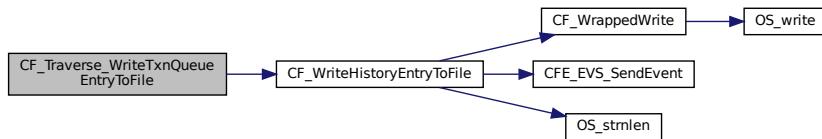
#### Return values

<i>CF_CLIST_CONT</i>	if everything is going well
<i>CF_CLIST_EXIT</i>	if a write error occurred, which means traversal should stop

Definition at line 332 of file cf\_utils.c.

References [CF\\_CLIST\\_CONT](#), [CF\\_CLIST\\_EXIT](#), [CF\\_WriteHistoryEntryToFile\(\)](#), [container\\_of](#), [CF\\_Traverse\\_WriteTxn->FileArg::counter](#), [CF\\_Traverse\\_WriteTxnFileArg::error](#), [CF\\_Traverse\\_WriteTxnFileArg::fd](#), and [CF\\_Transaction::history](#). Referenced by [CF\\_WriteTxnQueueDataToFile\(\)](#).

Here is the call graph for this function:



```
12.62.2.13 CF_TraverseAllTransactions() int32 CF_TraverseAllTransactions (
    CF_Channel_t * chan,
    CF_TraverseAllTransactions_fn_t fn,
    void * context )
```

Traverses all transactions on all active queues and performs an operation on them.

#### Assumptions, External Events, and Notes:

chan must not be NULL. fn must be a valid function. context must not be NULL.

**Parameters**

<i>chan</i>	Channel to operate on
<i>fn</i>	Callback to invoke for all traversed transactions
<i>context</i>	Opaque object to pass to all callbacks

**Returns**

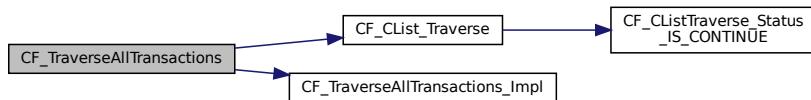
Number of transactions traversed

Definition at line 472 of file cf\_utils.c.

References CF\_CList\_Traverse(), CF\_QueueIdx\_PEND, CF\_QueueIdx\_RX, CF\_TraverseAllTransactions\_Impl(), CF\_TraverseAll\_Arg::counter, and CF\_Channel::qs.

Referenced by CF\_TraverseAllTransactions\_All\_Channels(), and CF\_TsnChanAction().

Here is the call graph for this function:



**12.62.2.14 CF\_TraverseAllTransactions\_All\_Channels()** `int32 CF_TraverseAllTransactions_All_Channels(`

```
CF_TraverseAllTransactions_fn_t fn,
void * context )
```

Traverses all transactions on all channels and performs an operation on them.

**Assumptions, External Events, and Notes:**

*fn* must be a valid function. *context* must not be NULL.

**Parameters**

<i>fn</i>	Callback to invoke for all traversed transactions
<i>context</i>	Opaque object to pass to all callbacks

**Returns**

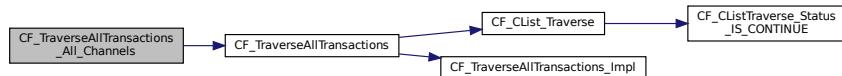
Number of transactions traversed

Definition at line 488 of file cf\_utils.c.

References CF\_AppData, CF\_NUM\_CHANNELS, CF\_TraverseAllTransactions(), CF\_Engine::channels, and CF\_AppData\_t::engine.

Referenced by CF\_TsnChanAction().

Here is the call graph for this function:



### 12.62.2.15 CF\_TraverseAllTransactions\_Impl() CF\_CListTraverse\_Status\_t CF\_TraverseAllTransactions->\_Impl (

```

    CF_CListNode_t * node,
    void * arg )

```

List traversal function performs operation on every active transaction.

**Description**

Called on every transaction via list traversal. Calls another function on that transaction.

**Assumptions, External Events, and Notes:**

node must not be NULL. args must not be NULL.

**Parameters**

<i>node</i>	Node being currently traversed
<i>arg</i>	Intermediate context object from initial call

**Return values**

0	for do not exit early (always continue)
---	---

Definition at line 457 of file cf\_utils.c.

References CF\_CLIST\_CONT, container\_of, CF\_TraverseAll\_Arg::context, CF\_TraverseAll\_Arg::counter, and CF\_TraverseAll\_Arg::fn.

Referenced by CF\_TraverseAllTransactions().

### 12.62.2.16 CF\_TxnStatus\_From\_ConditionCode() CF\_TxnStatus\_t CF\_TxnStatus\_From\_ConditionCode (

```

    CF_CFDP_ConditionCode_t cc )

```

Converts a CFDP condition code to an internal transaction status.

**Assumptions, External Events, and Notes:**

None

**Parameters**

cc	CFDP condition code
----	---------------------

**Returns**

Transaction status code

Definition at line 655 of file cf\_utils.c.

Referenced by CF\_CFDP\_R\_SubstateRecvEof(), and CF\_CFDP\_S\_SubstateRecvFin().

**12.62.2.17 CF\_TxnStatus\_To\_ConditionCode()** [CF\\_CFDP\\_ConditionCode\\_t](#) CF\_TxnStatus\_To\_ConditionCode(  
(

[CF\\_TxnStatus\\_t](#) txn\_stat )

Converts the internal transaction status to a CFDP condition code.

Transaction status is a superset of condition codes, and includes other error conditions for which CFDP will not send FIN/ACK/EOF and thus there is no corresponding condition code.

**Assumptions, External Events, and Notes:**

Not all transaction status codes directly correlate to a CFDP CC

**Parameters**

txn_stat	Transaction status
----------	--------------------

**Returns**

CFDP protocol condition code

Definition at line 589 of file cf\_utils.c.

References CF\_CFDP\_ConditionCode\_CANCEL\_REQUEST\_RECEIVED, CF\_CFDP\_ConditionCode\_INACTIVITY\_DETECTED, CF\_CFDP\_ConditionCode\_NO\_ERROR, CF\_TxnStatus\_ACK\_LIMIT\_NO\_EOF, CF\_TxnStatus\_ACK\_LIMIT\_NO\_FIN, CF\_TxnStatus\_CANCEL\_REQUEST\_RECEIVED, CF\_TxnStatus\_CHECK\_LIMIT\_REACHED, CF\_TxnStatus\_FILE\_CHECKSUM\_FAILURE, CF\_TxnStatus\_FILE\_SIZE\_ERROR, CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_TxnStatus\_INACTIVITY\_DETECTED, CF\_TxnStatus\_INVALID\_FILE\_STRUCTURE, CF\_TxnStatus\_INVALID\_TRANSMISSION\_MODE, CF\_TxnStatus\_IsError(), CF\_TxnStatus\_KEEP\_ALIVE\_LIMIT\_REACHED, CF\_TxnStatus\_NAK\_LIMIT\_REACHED, CF\_TxnStatus\_NO\_ERROR, CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED, CF\_TxnStatus\_SUSPEND\_REQUEST\_RECEIVED, and CF\_TxnStatus\_UNSUPPORTED\_CHECKSUM\_TYPE.

Referenced by CF\_CFDP\_SendEof(), and CF\_CFDP\_SendFin().

Here is the call graph for this function:



```
12.62.2.18 CF_WrappedClose() void CF_WrappedClose (
    osal_id_t fd )
```

Wrap the filesystem close call with a perf counter.

**Assumptions, External Events, and Notes:**

None

**See also**

[OS\\_close\(\)](#) for parameter descriptions

**Parameters**

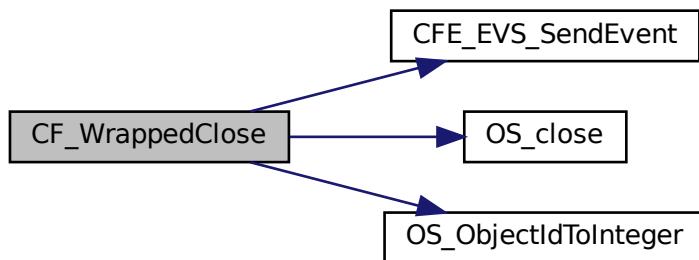
<i>fd</i>	Passed directly to underlying OSAL call
-----------	---

Definition at line 519 of file cf\_utils.c.

References CF\_CFDP\_CLOSE\_ERR\_EID, CF\_PERF\_ID\_FCLOSE, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), OS\_close(), OS\_ObjectIdToInteger(), and OS\_SUCCESS.

Referenced by CF\_CFDP\_CloseFiles(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S\_Init(), and CF\_WriteQueueCmd().

Here is the call graph for this function:



```
12.62.2.19 CF_WrappedLseek() CFE_Status_t CF_WrappedLseek (
    osal_id_t fd,
    off_t offset,
    int mode )
```

Wrap the filesystem lseek call with a perf counter.

**Assumptions, External Events, and Notes:**

fname must not be NULL.

**See also**

[OS\\_Iseek\(\)](#) for parameter descriptions

**Parameters**

<i>fd</i>	Passed directly to underlying OSAL call
<i>offset</i>	Passed directly to underlying OSAL call
<i>mode</i>	Passed directly to underlying OSAL call

**Returns**

Status code from OSAL (byte count or error code)

Definition at line 572 of file cf\_utils.c.

References CF\_PERF\_ID\_FSEEK, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_Iseek().

Referenced by CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_S\_Init(), and CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:



```
12.62.2.20 CF_WrappedOpenCreate() CFE_Status_t CF_WrappedOpenCreate (
    osal_id_t * fd,
    const char * fname,
    int32 flags,
    int32 access )
```

Wrap the filesystem open call with a perf counter.

**Assumptions, External Events, and Notes:**

fname must not be NULL.

**See also**

[OS\\_OpenCreate\(\)](#) for parameter descriptions

**Parameters**

<i>fd</i>	Passed directly to underlying OSAL call
<i>fname</i>	Passed directly to underlying OSAL call
<i>flags</i>	Passed directly to underlying OSAL call
<i>access</i>	Passed directly to underlying OSAL call

## Returns

Status code from OSAL

Definition at line 503 of file cf\_utils.c.

References CF\_PERF\_ID\_FOPEN, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_OpenCreate().

Referenced by CF\_CFDP\_R\_Init(), CF\_CFDP\_S\_Init(), and CF\_WriteQueueCmd().

Here is the call graph for this function:



### 12.62.2.21 CF\_WrappedRead() [CFE\\_Status\\_t](#) CF\_WrappedRead(

```
    osal_id_t fd,  
    void * buf,  
    size_t read_size )
```

Wrap the filesystem read call with a perf counter.

**Assumptions, External Events, and Notes:**

buf must not be NULL.

## See also

[OS\\_read\(\)](#) for parameter descriptions

## Parameters

<i>fd</i>	Passed directly to underlying OSAL call
<i>buf</i>	Passed directly to underlying OSAL call
<i>read_size</i>	Passed directly to underlying OSAL call

**Returns**

Status code from OSAL (byte count or error code)

Definition at line 540 of file cf\_utils.c.

References CF\_PERF\_ID\_FREAD, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_read().

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), and CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:



**12.62.2.22 CF\_WrappedWrite()** [CFE\\_Status\\_t](#) CF\_WrappedWrite (   
   `osal_id_t fd,`  
   `const void * buf,`  
   `size_t write_size )`

Wrap the filesystem write call with a perf counter.

**Assumptions, External Events, and Notes:**

buf must not be NULL.

**See also**

[OS\\_write\(\)](#) for parameter descriptions

**Parameters**

<code>fd</code>	Passed directly to underlying OSAL call
<code>buf</code>	Passed directly to underlying OSAL call
<code>write_size</code>	Passed directly to underlying OSAL call

**Returns**

Status code from OSAL (byte count or error code)

Definition at line 556 of file cf\_utils.c.

References CF\_PERF\_ID\_FWRITE, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_write().

Referenced by CF\_CFDP\_R\_ProcessFd(), and CF\_WriteHistoryEntryToFile().

Here is the call graph for this function:

**12.62.2.23 CF\_WriteHistoryEntryToFile()** `CFE_Status_t CF_WriteHistoryEntryToFile (`

```
    osal_id_t fd,
    const CF_History_t * history )
```

Write a single history to a file.

This creates a human-readable/string representation of the information in the history object, and writes that string to the indicated file.

This implements the common code between writing the history queue and transaction queue to a file, as both ultimately store the same information in a CF\_History\_t object, but have a different method to get to it.

**Assumptions, External Events, and Notes:**

fd should be a valid file descriptor, open for writing.

**Parameters**

<code>fd</code>	Open File descriptor to write to
<code>history</code>	Pointer to CF history object to write

**Return values**

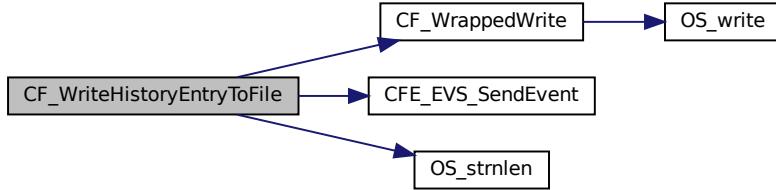
<code>CFE_SUCCESS</code>	on success
<code>CF_ERROR</code>	on error

Definition at line 255 of file cf\_utils.c.

References CF\_ASSERT, CF\_CMD\_WHIST\_WRITE\_ERR\_EID, CF\_Direction\_NUM, CF\_ERROR, CF\_FILENAME\_MAX\_LEN, CF\_WrappedWrite(), CFE\_EVT\_EventType\_ERROR, CFE\_EVT\_SendEvent(), CFE\_SUCCESS, CF\_History::dir, CF\_TxnFilenames::dst\_filename, CF\_History::fnames, OS\_strlen(), CF\_History::peer\_eid, CF\_History::seq\_num, CF\_History::src\_eid, CF\_TxnFilenames::src\_filename, and CF\_History::txn\_stat.

Referenced by CF\_Traverse\_WriteHistoryQueueEntryToFile(), and CF\_Traverse\_WriteTxnQueueEntryToFile().

Here is the call graph for this function:



#### 12.62.2.24 CF\_WriteHistoryQueueDataToFile() [CFE\\_Status\\_t CF\\_WriteHistoryQueueDataToFile \(](#)

```

osal_id_t fd,
CF_Channel_t * chan,
CF_Direction_t dir )

```

Write a history-based queue's entries to a file.

##### Assumptions, External Events, and Notes:

chan must not be NULL.

##### Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>dir</i>	Direction to match/filter

##### Return values

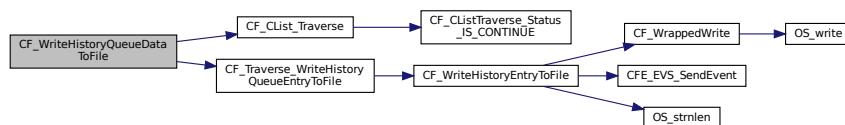
0	on success
1	on error

Definition at line 372 of file cf\_utils.c.

References CF\_CList\_Traverse(), CF\_QueueIdx\_HIST, CF\_Traverse\_WriteHistoryQueueEntryToFile(), CF\_Traverse<->\_WriteHistoryFileArg::counter, CF\_Traverse\_WriteHistoryFileArg::error, CF\_Traverse\_WriteHistoryFileArg::fd, CF\_Traverse<->\_WriteHistoryFileArg::filter\_dir, and CF\_Channel::qs.

Referenced by CF\_WriteQueueCmd().

Here is the call graph for this function:



```
12.62.2.25 CF_WriteTxnQueueDataToFile() CFE_Status_t CF_WriteTxnQueueDataToFile (
    osal_id_t fd,
    CF_Channel_t * chan,
    CF_QueueIdx_t queue )
```

Write a transaction-based queue's transaction history to a file.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

#### Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>queue</i>	Queue Index to write

#### Return values

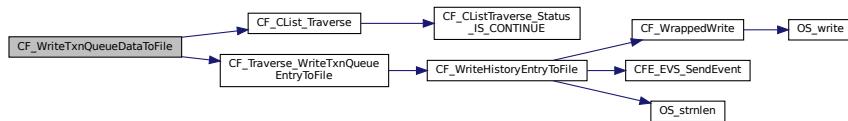
0	on success
1	on error

Definition at line 354 of file cf\_utils.c.

References CF\_CList\_Traverse(), CF\_Traverse\_WriteTxnQueueEntryToFile(), CF\_Traverse\_WriteTxnFileArg::counter, CF\_Traverse\_WriteTxnFileArg::error, CF\_Traverse\_WriteTxnFileArg::fd, and CF\_Channel::qs.

Referenced by CF\_WriteQueueCmd().

Here is the call graph for this function:



## 12.63 apps/cf/fsw/src(cf\_utils.h File Reference

```
#include "cf_cfdp.h"
#include "cf_app.h"
#include "cf_assert.h"
```

#### Data Structures

- struct [CF\\_Traverse\\_TransSeqArg](#)  
*Argument structure for use with CList\_Traverse()*
- struct [CF\\_Traverse\\_WriteHistoryFileArg](#)  
*Argument structure for use with CF\_Traverse\_WriteHistoryQueueEntryToFile()*
- struct [CF\\_Traverse\\_WriteTxnFileArg](#)

Argument structure for use with `CF_Traverse_WriteTxnQueueEntryToFile()`

- struct `CF_TraverseAll_Arg`

Argument structure for use with `CF_TraverseAllTransactions()`

- struct `CF_Traverse_PriorityArg`

Argument structure for use with `CF_CList_Traverse_R()`

## Typedefs

- typedef struct `CF_Traverse_TransSeqArg` `CF_Traverse_TransSeqArg_t`

Argument structure for use with `CList_Traverse()`

- typedef struct `CF_Traverse_WriteHistoryFileArg` `CF_Traverse_WriteHistoryFileArg_t`

Argument structure for use with `CF_Traverse_WriteHistoryQueueEntryToFile()`

- typedef struct `CF_Traverse_WriteTxnFileArg` `CF_Traverse_WriteTxnFileArg_t`

Argument structure for use with `CF_Traverse_WriteTxnQueueEntryToFile()`

- typedef void(\* `CF_TraverseAllTransactions_fn_t`) (`CF_Transaction_t` \*txn, void \*context)

Callback function type for use with `CF_TraverseAllTransactions()`

- typedef struct `CF_TraverseAll_Arg` `CF_TraverseAll_Arg_t`

Argument structure for use with `CF_TraverseAllTransactions()`

- typedef struct `CF_Traverse_PriorityArg` `CF_Traverse_PriorityArg_t`

Argument structure for use with `CF_CList_Traverse_R()`

## Functions

- static void `CF_DequeueTransaction` (`CF_Transaction_t` \*txn)

- static void `CF_MoveTransaction` (`CF_Transaction_t` \*txn, `CF_QueueIdx_t` queue)

- static void `CF_CList_Remove_Ex` (`CF_Channel_t` \*chan, `CF_QueueIdx_t` queueidx, `CF_CListNode_t` \*node)

- static void `CF_CList_InsertAfter_Ex` (`CF_Channel_t` \*chan, `CF_QueueIdx_t` queueidx, `CF_CListNode_t` \*start, `CF_CListNode_t` \*after)

- static void `CF_CList_InsertBack_Ex` (`CF_Channel_t` \*chan, `CF_QueueIdx_t` queueidx, `CF_CListNode_t` \*node)

- `CF_Transaction_t` \* `CF_FindUnusedTransaction` (`CF_Channel_t` \*chan, `CF_Direction_t` direction)

Find an unused transaction on a channel.

- void `CF_ResetHistory` (`CF_Channel_t` \*chan, `CF_History_t` \*history)

Returns a history structure back to its unused state.

- void `CF_FreeTransaction` (`CF_Transaction_t` \*txn, `uint8` chan)

Frees and resets a transaction and returns it for later use.

- `CF_Transaction_t` \* `CF_FindTransactionBySequenceNumber` (`CF_Channel_t` \*chan, `CF_TransactionSeq_t` transaction\_sequence\_number, `CF_EntityId_t` src\_eid)

Finds an active transaction by sequence number.

- `CFE_Status_t` `CF_FindTransactionBySequenceNumber_Impl` (`CF_CListNode_t` \*node, `CF_Traverse_TransSeqArg_t` \*context)

List traversal function to check if the desired sequence number matches.

- `CFE_Status_t` `CF_WriteHistoryEntryToFile` (`osal_id_t` fd, const `CF_History_t` \*history)

Write a single history to a file.

- `CFE_Status_t` `CF_WriteTxnQueueDataToFile` (`osal_id_t` fd, `CF_Channel_t` \*chan, `CF_QueueIdx_t` queue)

Write a transaction-based queue's transaction history to a file.

- `CFE_Status_t` `CF_WriteHistoryQueueDataToFile` (`osal_id_t` fd, `CF_Channel_t` \*chan, `CF_Direction_t` dir)

Write a history-based queue's entries to a file.

- void `CF_InsertSortPrio` (`CF_Transaction_t` \*txn, `CF_QueueIdx_t` queue)

Insert a transaction into a priority sorted transaction queue.

- `int32 CF_TraverseAllTransactions (CF_Channel_t *chan, CF_TraverseAllTransactions_fn_t fn, void *context)`  
*Traverses all transactions on all active queues and performs an operation on them.*
- `int32 CF_TraverseAllTransactions_All_Channels (CF_TraverseAllTransactions_fn_t fn, void *context)`  
*Traverses all transactions on all channels and performs an operation on them.*
- `CF_CListTraverse_Status_t CF_TraverseAllTransactions_Impl (CF_CListNode_t *node, void *arg)`  
*List traversal function performs operation on every active transaction.*
- `CF_CListTraverse_Status_t CF_Traverse_WriteHistoryQueueEntryToFile (CF_CListNode_t *node, void *arg)`  
*Writes a human readable representation of a history queue entry to a file.*
- `CF_CListTraverse_Status_t CF_Traverse_WriteTxnQueueEntryToFile (CF_CListNode_t *node, void *arg)`  
*Writes a human readable representation of a transaction history entry to a file.*
- `CF_CListTraverse_Status_t CF_PrioSearch (CF_CListNode_t *node, void *context)`  
*Searches for the first transaction with a lower priority than given.*
- `CFE_Status_t CF_WrappedOpenCreate (osal_id_t *fd, const char *fname, int32 flags, int32 access)`  
*Wrap the filesystem open call with a perf counter.*
- `void CF_WrappedClose (osal_id_t fd)`  
*Wrap the filesystem close call with a perf counter.*
- `CFE_Status_t CF_WrappedRead (osal_id_t fd, void *buf, size_t read_size)`  
*Wrap the filesystem read call with a perf counter.*
- `CFE_Status_t CF_WrappedWrite (osal_id_t fd, const void *buf, size_t write_size)`  
*Wrap the filesystem write call with a perf counter.*
- `CFE_Status_t CF_WrappedLseek (osal_id_t fd, off_t offset, int mode)`  
*Wrap the filesystem lseek call with a perf counter.*
- `CF_CFDP_ConditionCode_t CF_TxnStatus_To_ConditionCode (CF_TxnStatus_t txn_stat)`  
*Converts the internal transaction status to a CFDP condition code.*
- `CF_TxnStatus_t CF_TxnStatus_From_ConditionCode (CF_CFDP_ConditionCode_t cc)`  
*Converts a CFDP condition code to an internal transaction status.*
- `static bool CF_TxnStatus_IsError (CF_TxnStatus_t txn_stat)`  
*Check if the internal transaction status represents an error.*
- `CF_Channel_t * CF_GetChannelFromTxn (CF_Transaction_t *txn)`  
*Gets the associated channel struct from a transaction.*
- `CF_CListNode_t ** CF_GetChunkListHead (CF_Channel_t *chan, uint8 direction)`  
*Gets the head of the chunk list for the given channel + direction.*
- `CF_CFDP_AckTxnStatus_t CF_CFDP_GetAckTxnStatus (CF_Transaction_t *txn)`  
*Gets the status of this transaction.*

### 12.63.1 Detailed Description

The CF Application utils header file

### 12.63.2 Typedef Documentation

**12.63.2.1 CF\_Traverse\_PriorityArg\_t** `typedef struct CF_Traverse_PriorityArg CF_Traverse_PriorityArg_t`  
 Argument structure for use with `CF_CList_Traverse_R()`  
 This is for searching for transactions of a specific priority

**12.63.2.2 CF\_Traverse\_TransSeqArg\_t** `typedef struct CF_Traverse_TransSeqArg CF_Traverse_TransSeqArg_t`  
Argument structure for use with `CList_Traverse()`

This identifies a specific transaction sequence number and entity ID. The transaction pointer is set by the implementation.

**12.63.2.3 CF\_Traverse\_WriteHistoryFileArg\_t** `typedef struct CF_Traverse_WriteHistoryFileArg CF_Traverse_WriteHistoryFileArg_t`  
Argument structure for use with `CF_Traverse_WriteHistoryQueueEntryToFile()`

This is used for writing status files. It contains a designated file descriptor for output and counters.

When traversing history, the list contains all entries, and may need additional filtering for direction (TX/RX) depending on what information the user has requested.

**12.63.2.4 CF\_Traverse\_WriteTxnFileArg\_t** `typedef struct CF_Traverse_WriteTxnFileArg CF_Traverse_WriteTxnFileArg_t`  
Argument structure for use with `CF_Traverse_WriteTxnQueueEntryToFile()`

This is used for writing status files. It contains a designated file descriptor for output and counters.

When traversing transactions, the entire list is written to the file. No additional filtering is necessary, because the queues themselves are limited in what they contain (therefore "pre-filtered" to some degree).

**12.63.2.5 CF\_TraverseAll\_Arg\_t** `typedef struct CF_TraverseAll_Arg CF_TraverseAll_Arg_t`

Argument structure for use with `CF_TraverseAllTransactions()`

This basically allows for running a CF\_Traverse on several lists at once

**12.63.2.6 CF\_TraverseAllTransactions\_fn\_t** `typedef void(* CF_TraverseAllTransactions_fn_t) (CF_Transaction_t * txn, void * context)`

Callback function type for use with `CF_TraverseAllTransactions()`

#### Parameters

<code>txn</code>	Pointer to current transaction being traversed
<code>context</code>	Opaque object passed from initial call

Definition at line 88 of file cf\_utils.h.

### 12.63.3 Function Documentation

**12.63.3.1 CF\_CFDP\_GetAckTxnStatus()** `CF_CFDP_AckTxnStatus_t CF_CFDP_GetAckTxnStatus ( CF_Transaction_t * txn )`

Gets the status of this transaction.

Determines if the transaction is ACTIVE or TERMINATED. (By definition if it has a txn object then it is not UNRECOGNIZED)

#### Parameters

<code>txn</code>	Transaction
------------------	-------------

**Returns**

`CF_CFDP_AckTxnStatus_t` value corresponding to transaction

Definition at line 87 of file cf\_utils.c.

References `CF_CFDP_AckTxnStatus_ACTIVE`, `CF_CFDP_AckTxnStatus_INVALID`, `CF_CFDP_AckTxnStatus_TERMINATED`, `CF_CFDP_AckTxnStatus_UNRECOGNIZED`, `CF_TxnState_DROP`, `CF_TxnState_HOLD`, `CF_TxnState_R1`, `CF_TxnState_R2`, `CF_TxnState_S1`, `CF_TxnState_S2`, and `CF_Transaction::state`.

Referenced by `CF_CFDP_ArmInactTimer()`, `CF_CFDP_R_Tick()`, `CF_CFDP_S_Tick()`, and `CF_CFDP_SendAck()`.

**12.63.3.2 CF\_CList\_InsertAfter\_Ex()** static void CF\_CList\_InsertAfter\_Ex (

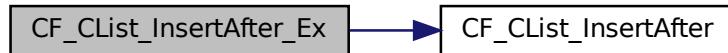
```
    CF_Channel_t * chan,
    CF_QueueIdx_t queueidx,
    CF_CListNode_t * start,
    CF_CListNode_t * after ) [inline], [static]
```

Definition at line 148 of file cf\_utils.h.

References `CF_AppData`, `CF_CList_InsertAfter()`, `CF_HkPacket_Payload::channel_hk`, `CF_Engine::channels`, `CF_AppData_t::engine`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, `CF_HkChannel_Data::q_size`, and `CF_Channel::qs`.

Referenced by `CF_InsertSortPrio()`.

Here is the call graph for this function:

**12.63.3.3 CF\_CList\_InsertBack\_Ex()** static void CF\_CList\_InsertBack\_Ex (

```
    CF_Channel_t * chan,
    CF_QueueIdx_t queueidx,
    CF_CListNode_t * node ) [inline], [static]
```

Definition at line 155 of file cf\_utils.h.

References `CF_AppData`, `CF_CList_InsertBack()`, `CF_HkPacket_Payload::channel_hk`, `CF_Engine::channels`, `CF_AppData_t::engine`, `CF_AppData_t::hk`, `CF_HkPacket::Payload`, `CF_HkChannel_Data::q_size`, and `CF_Channel::qs`.

Referenced by `CF_CFDP_InitEngine()`, `CF_CFDP_RecycleTransaction()`, `CF_CFDP_StartRxTransaction()`, `CF_FreeTransaction()`, `CF_InsertSortPrio()`, and `CF_ResetHistory()`.

Here is the call graph for this function:



```
12.63.3.4 CF_CList_Remove_Ex() static void CF_CList_Remove_Ex (
    CF_Channel_t * chan,
    CF_QueueIdx_t queueidx,
    CF_CListNode_t * node ) [inline], [static]
```

Definition at line 141 of file cf\_utils.h.

References CF\_AppData, CF\_Assert, CF\_CList\_Remove(), CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_AppData\_t::engine, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_HkChannel\_Data::q\_size, and CF\_Channel::qs.

Referenced by CF\_FindUnusedTransaction(), and CF\_ResetHistory().

Here is the call graph for this function:



```
12.63.3.5 CF_DequeueTransaction() static void CF_DequeueTransaction (
    CF_Transaction_t * txn ) [inline], [static]
```

Definition at line 122 of file cf\_utils.h.

References CF\_AppData, CF\_Assert, CF\_CList\_Remove(), CF\_NUM\_CHANNELS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Transaction::cl\_node, CF\_StateFlags::com, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Flags\_Common::q\_index, CF\_HkChannel\_Data::q\_size, and CF\_Channel::qs.

Referenced by CF\_CFDP\_RecycleTransaction(), and CF\_CFDP\_SetupTxTransaction().

Here is the call graph for this function:



```
12.63.3.6 CF_FindTransactionBySequenceNumber() CF_Transaction_t* CF_FindTransactionBySequenceNumber (
    CF_Channel_t * chan,
    CF_TransactionSeq_t transaction_sequence_number,
    CF_EntityId_t src_eid )
```

Finds an active transaction by sequence number.

#### Description

This function traverses the active rx, pending, txa, and txw transaction and looks for the requested transaction.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

**Parameters**

<i>chan</i>	Pointer to the CF channel
<i>transaction_sequence_number</i>	Sequence number to find
<i>src_eid</i>	Entity ID associated with sequence number

**Returns**

Pointer to the given transaction if found

**Return values**

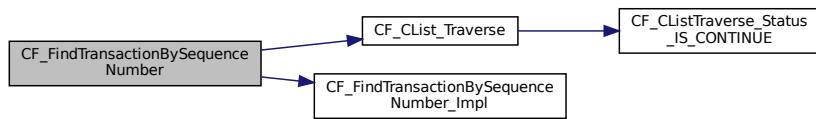
<i>NULL</i>	if the transaction is not found
-------------	---------------------------------

Definition at line 223 of file cf\_utils.c.

References CF\_CList\_Traverse(), CF\_FindTransactionBySequenceNumber\_Impl(), CF\_QueueIdx\_PEND, CF\_QueueIdx\_RX, CF\_QueueIdx\_TX, CF\_Channel::qs, and CF\_Traverse\_TransSeqArg::txn.

Referenced by CF\_CFDP\_ReceivePdu(), and CF\_FindTransactionBySequenceNumberAllChannels().

Here is the call graph for this function:



**12.63.3.7 CF\_FindTransactionBySequenceNumber\_Impl()** *CFE\_Status\_t* CF\_FindTransactionBySequenceNumber\_Impl (

*CF\_CListNode\_t* \* *node*,  
*CF\_Traverse\_TransSeqArg\_t* \* *context* )

List traversal function to check if the desired sequence number matches.

**Assumptions, External Events, and Notes:**

context must not be NULL. node must not be NULL.

**Parameters**

<i>node</i>	Pointer to node currently being traversed
<i>context</i>	Pointer to state object passed through from initial call

**Return values**

<i>1</i>	when it's found, which terminates list traversal
<i>0</i>	when it isn't found, which causes list traversal to continue

Definition at line 203 of file cf\_utils.c.

References container\_of, CF\_Transaction::history, CF\_History::seq\_num, CF\_History::src\_eid, CF\_Traverse\_TransSeqArg::src\_eid, CF\_Traverse\_TransSeqArg::transaction\_sequence\_number, and CF\_Traverse\_TransSeqArg::txn. Referenced by CF\_FindTransactionBySequenceNumber().

**12.63.3.8 CF\_FindUnusedTransaction()** `CF_Transaction_t* CF_FindUnusedTransaction (`  
`CF_Channel_t * chan,`  
`CF_Direction_t direction )`

Find an unused transaction on a channel.

**Assumptions, External Events, and Notes:**

chan must not be NULL.

#### Parameters

<code>chan</code>	Pointer to the CF channel
<code>direction</code>	Intended direction of data flow (TX or RX)

#### Returns

Pointer to a free transaction

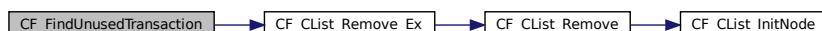
#### Return values

<code>NULL</code>	if no free transactions available.
-------------------	------------------------------------

Definition at line 127 of file cf\_utils.c.

References CF\_ASSERT, CF\_CList\_Remove\_Ex(), CF\_QueueIdx\_FREE, CF\_QueueIdx\_HIST, CF\_QueueIdx\_HIST\_FREE, CF\_TxnState\_INIT, CF\_History::cl\_node, CF\_Transaction::cl\_node, container\_of, CF\_History::dir, CF\_Transaction::history, CF\_Channel::qs, and CF\_Transaction::state. Referenced by CF\_CFDP\_ProcessPlaybackDirectory(), CF\_CFDP\_StartRxTransaction(), and CF\_CFDP\_TxFile().

Here is the call graph for this function:



**12.63.3.9 CF\_FreeTransaction()** `void CF_FreeTransaction (`  
`CF_Transaction_t * txn,`  
`uint8 chan )`

Frees and resets a transaction and returns it for later use.

**Assumptions, External Events, and Notes:**

txn must not be NULL.

### Parameters

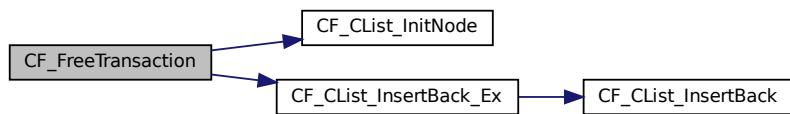
<i>txn</i>	Pointer to the transaction object
<i>chan</i>	The channel number which this transaction is associated with

Definition at line 189 of file cf\_utils.c.

References CF\_AppData, CF\_CList\_InitNode(), CF\_CList\_InsertBack\_Ex(), CF\_QueueIdx\_FREE, CF\_Transaction::chan\_num, CF\_Engine::channels, CF\_Transaction::cl\_node, and CF\_AppData\_t::engine.

Referenced by CF\_CFDP\_InitEngine(), and CF\_CFDP\_RecycleTransaction().

Here is the call graph for this function:



### 12.63.3.10 CF\_GetChannelFromTxn()

```
CF_Channel_t* CF_GetChannelFromTxn (
    CF_Transaction_t * txn )
```

Gets the associated channel struct from a transaction.

#### Assumptions, External Events, and Notes:

txn must not be null, and the chan\_num must be set

### Parameters

<i>txn</i>	Transaction
------------	-------------

### Returns

Pointer to CF\_Channel\_t struct associated with the transaction

### Return values

NULL	if checks failed
------	------------------

Definition at line 43 of file cf\_utils.c.

References CF\_AppData, CF\_NUM\_CHANNELS, CF\_Transaction::chan\_num, CF\_Engine::channels, and CF\_AppData\_t::engine.

Referenced by CF\_CFDP\_AllocChunkList(), CF\_CFDP\_CompleteTick(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_RecycleTransaction(), and CF\_CFDP\_S\_Tick\_NewData().

### 12.63.3.11 CF\_GetChunkListHead()

```
CF_CListNode_t** CF_GetChunkListHead (
    CF_Channel_t * chan,
```

```
    uint8 direction )
```

Gets the head of the chunk list for the given channel + direction.

The chunk list contains structs that are available for tracking the chunks associated with files in transit. An entry needs to be pulled from this list for every transaction, and returned to this list when the transaction completes.

#### Parameters

<i>chan</i>	Pointer to channel struct
<i>direction</i>	Whether this is TX or RX

#### Returns

Pointer to list head

Definition at line 65 of file cf\_utils.c.

References CF\_Direction\_NUM, and CF\_Channel::cs.

Referenced by CF\_CFDP\_FindUnusedChunks(), CF\_CFDP\_InitEngine(), and CF\_CFDP\_RecycleTransaction().

**12.63.3.12 CF\_InsertSortPrio()** void CF\_InsertSortPrio (

```
    CF_Transaction_t * txn,
    CF_QueueIdx_t queue )
```

Insert a transaction into a priority sorted transaction queue.

#### Description

This function works by walking the queue in reverse to find a transaction with a higher priority than the given transaction. The given transaction is then inserted after that one, since it would be the next lower priority.

#### Assumptions, External Events, and Notes:

txn must not be NULL.

#### Parameters

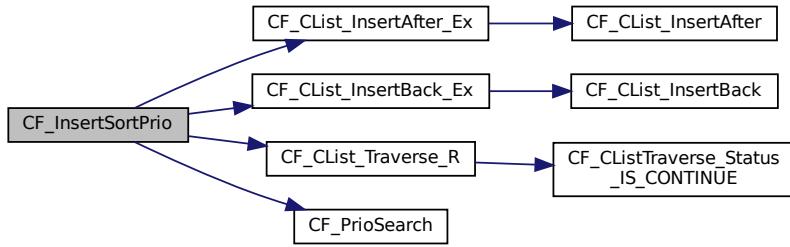
<i>txn</i>	Pointer to the transaction object
<i>queue</i>	Index of queue to insert into

Definition at line 415 of file cf\_utils.c.

References CF\_AppData, CF\_Assert, CF\_CList\_InsertAfter\_Ex(), CF\_CList\_InsertBack\_Ex(), CF\_CList\_Traverse\_R(), CF\_NUM\_CHANNELS, CF\_PrioSearch(), CF\_Transaction::chan\_num, CF\_Engine::channels, CF\_Transaction::cl\_node, CF\_StateFlags::com, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_Transaction::priority, CF\_Flags::Common::q\_index, CF\_Channel::qs, and CF\_Traverse\_PriorityArg::txn.

Referenced by CF\_CFDP\_SetupTxTransaction(), and CF\_CFDP\_TxFile\_Initiate().

Here is the call graph for this function:

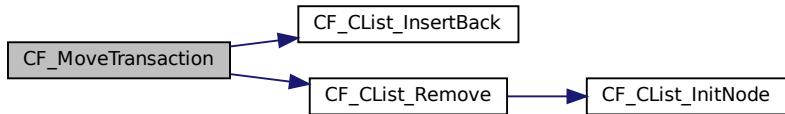


**12.63.3.13 CF\_MoveTransaction()** static void CF\_MoveTransaction ( CF\_Transaction\_t \* txn,  
CF\_QueueIdx\_t queue ) [inline], [static]

Definition at line 130 of file cf\_utils.h.

References CF\_AppData, CF\_Assert, CF\_CList\_InsertBack(), CF\_CList\_Remove(), CF\_NUM\_CHANNELS, CF\_Transaction::chan\_num, CF\_HkPacket\_Payload::channel\_hk, CF\_Engine::channels, CF\_Transaction::cl\_node, CF\_StateFlags::com, CF\_AppData\_t::engine, CF\_Transaction::flags, CF\_AppData\_t::hk, CF\_HkPacket::Payload, CF\_Flags\_Common::q\_index, CF\_HkChannel\_Data::q\_size, and CF\_Channel::qs.

Here is the call graph for this function:



**12.63.3.14 CF\_PrioSearch()** CF\_CListTraverse\_Status\_t CF\_PrioSearch ( CF\_CListNode\_t \* node,  
void \* context )

Searches for the first transaction with a lower priority than given.

**Assumptions, External Events, and Notes:**

node must not be NULL. context must not be NULL.

#### Parameters

<i>node</i>	Node being currently traversed
<i>context</i>	Pointer to CF_Traverse_PriorityArg_t object indicating the priority to search for

**Return values**

<i>CF_CLIST_EXIT</i>	when it's found, which terminates list traversal
<i>CF_CLIST_CONT</i>	when it isn't found, which causes list traversal to continue

Definition at line 391 of file cf\_utils.c.

References *CF\_CLIST\_CONT*, *CF\_CLIST\_EXIT*, *container\_of*, *CF\_Transaction::priority*, *CF\_Traverse\_PriorityArg*↔*:priority*, and *CF\_Traverse\_PriorityArg::txn*.

Referenced by *CF\_InsertSortPrio()*.

**12.63.3.15 CF\_ResetHistory()** `void CF_ResetHistory (`

```
    CF_Channel_t * chan,
    CF_History_t * history )
```

Returns a history structure back to its unused state.

**Description**

There's nothing to do currently other than remove the history from its current queue and put it back on *CF\_QueueIdx\_HIST\_FREE*.

**Assumptions, External Events, and Notes:**

*chan* must not be NULL. *history* must not be NULL.

**Parameters**

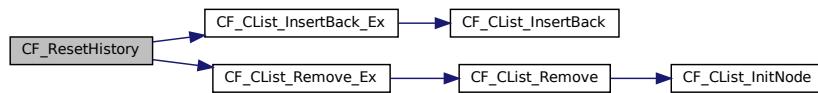
<i>chan</i>	Pointer to the CF channel
<i>history</i>	Pointer to the history entry

Definition at line 177 of file cf\_utils.c.

References *CF\_CList\_InsertBack\_Ex()*, *CF\_CList\_Remove\_Ex()*, *CF\_QueueIdx\_HIST*, *CF\_QueueIdx\_HIST\_FREE*, and *CF\_History::cl\_node*.

Referenced by *CF\_PurgeHistory()*.

Here is the call graph for this function:

**12.63.3.16 CF\_Traverse\_WriteHistoryQueueEntryToFile()** `CF_CListTraverse_Status_t CF_Traverse_Write→`

*HistoryQueueEntryToFile* (

```
    CF_CListNode_t * node,
    void * arg )
```

Writes a human readable representation of a history queue entry to a file.

This function is a wrapper around `CF_WriteHistoryEntryToFile()` that can be used with `CF_Traverse()` to write history queue entries to the file.

This also implements a direction filter, so only matching entries are actually written to the file.

**Assumptions, External Events, and Notes:**

node must not be NULL. arg must not be NULL.

**Parameters**

<i>node</i>	Node being currently traversed
<i>arg</i>	Pointer to CF_Traverse_WriteHistoryFileArg_t indicating the file information

**Return values**

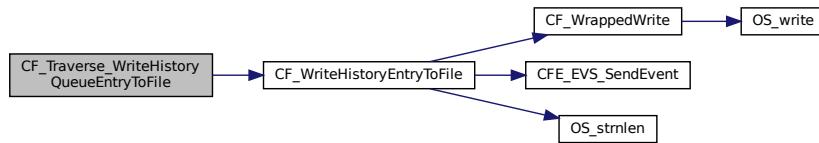
<i>CF_CLIST_CONT</i>	if everything is going well
<i>CF_CLIST_EXIT</i>	if a write error occurred, which means traversal should stop

Definition at line 305 of file cf\_utils.c.

References CF\_CLIST\_CONT, CF\_CLIST\_EXIT, CF\_Direction\_NUM, CF\_WriteHistoryEntryToFile(), container\_of, CF\_Traverse\_WriteHistoryFileArg::counter, CF\_History::dir, CF\_Traverse\_WriteHistoryFileArg::error, CF\_Traverse\_WriteHistoryFileArg::fd, and CF\_Traverse\_WriteHistoryFileArg::filter\_dir.

Referenced by CF\_WriteHistoryQueueDataToFile().

Here is the call graph for this function:



### 12.63.3.17 CF\_Traverse\_WriteTxnQueueEntryToFile() [CF\\_CListTraverse\\_Status\\_t](#) CF\_Traverse\_WriteTxn→

QueueEntryToFile (

```

    CF_CListNode_t * node,
    void * arg
)
```

Writes a human readable representation of a transaction history entry to a file.

This function is a wrapper around `CF_WriteHistoryEntryToFile()` that can be used with `CF_Traverse()` to write transaction queue entries to the file.

**Assumptions, External Events, and Notes:**

node must not be NULL. arg must not be NULL.

**Parameters**

<i>node</i>	Node being currently traversed
<i>arg</i>	Pointer to CF_Traverse_WriteTxnFileArg_t indicating the file information

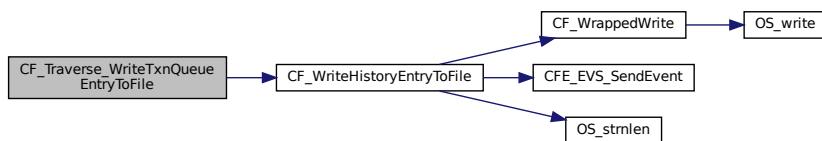
### Return values

<code>CF_CLIST_CONT</code>	if everything is going well
<code>CF_CLIST_EXIT</code>	if a write error occurred, which means traversal should stop

Definition at line 332 of file cf\_utils.c.

References `CF_CLIST_CONT`, `CF_CLIST_EXIT`, `CF_WriteHistoryEntryToFile()`, `container_of`, `CF_Traverse_WriteTxnToFileArg::counter`, `CF_Traverse_WriteTxnFileArg::error`, `CF_Traverse_WriteTxnFileArg::fd`, and `CF_Transaction::history`. Referenced by `CF_WriteTxnQueueDataToFile()`.

Here is the call graph for this function:



```

12.63.3.18 CF_TraverseAllTransactions() int32 CF_TraverseAllTransactions (
    CF_Channel_t * chan,
    CF_TraverseAllTransactions_fn_t fn,
    void * context )
  
```

Traverses all transactions on all active queues and performs an operation on them.

#### Assumptions, External Events, and Notes:

`chan` must not be NULL. `fn` must be a valid function. `context` must not be NULL.

#### Parameters

<code>chan</code>	Channel to operate on
<code>fn</code>	Callback to invoke for all traversed transactions
<code>context</code>	Opaque object to pass to all callbacks

### Returns

Number of transactions traversed

Definition at line 472 of file cf\_utils.c.

References CF\_CList\_Traverse(), CF\_QueueIdx\_PEND, CF\_QueueIdx\_RX, CF\_TraverseAllTransactions\_Impl(), CF\_TraverseAll\_Arg::counter, and CF\_Channel::qs.

Referenced by CF\_TraverseAllTransactions\_All\_Channels(), and CF\_TsnChanAction().

Here is the call graph for this function:



### 12.63.3.19 CF\_TraverseAllTransactions\_All\_Channels()

```

int32 CF_TraverseAllTransactions_All_Channels(
(
    CF_TraverseAllTransactions_fn_t fn,
    void * context )
)
  
```

Traverses all transactions on all channels and performs an operation on them.

#### Assumptions, External Events, and Notes:

`fn` must be a valid function. `context` must not be NULL.

### Parameters

<code>fn</code>	Callback to invoke for all traversed transactions
<code>context</code>	Opaque object to pass to all callbacks

### Returns

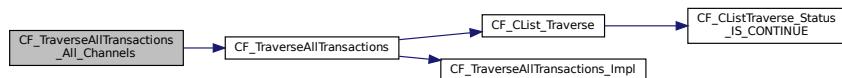
Number of transactions traversed

Definition at line 488 of file cf\_utils.c.

References CF\_AppData, CF\_NUM\_CHANNELS, CF\_TraverseAllTransactions(), CF\_Engine::channels, and CF\_AppData\_t::engine.

Referenced by CF\_TsnChanAction().

Here is the call graph for this function:



```
12.63.3.20 CF_TraverseAllTransactions_Impl() CF_CListTraverse_Status_t CF_TraverseAllTransactions←
_Impl (
    CF_CListNode_t * node,
    void * arg )
```

List traversal function performs operation on every active transaction.

#### Description

Called on every transaction via list traversal. Calls another function on that transaction.

#### Assumptions, External Events, and Notes:

node must not be NULL. args must not be NULL.

#### Parameters

<i>node</i>	Node being currently traversed
<i>arg</i>	Intermediate context object from initial call

#### Return values

<i>0</i>	for do not exit early (always continue)
----------	---

Definition at line 457 of file cf\_utils.c.

References CF\_CLIST\_CONT, container\_of, CF\_TraverseAll\_Arg::context, CF\_TraverseAll\_Arg::counter, and CF\_TraverseAll\_Arg::fn.

Referenced by CF\_TraverseAllTransactions().

```
12.63.3.21 CF_TxnStatus_From_ConditionCode() CF_TxnStatus_t CF_TxnStatus_From_ConditionCode (
    CF_CFDP_ConditionCode_t cc )
```

Converts a CFDP condition code to an internal transaction status.

#### Assumptions, External Events, and Notes:

None

#### Parameters

<i>cc</i>	CFDP condition code
-----------	---------------------

#### Returns

Transaction status code

Definition at line 655 of file cf\_utils.c.

Referenced by CF\_CFDP\_R\_SubstateRecvEof(), and CF\_CFDP\_S\_SubstateRecvFin().

```
12.63.3.22 CF_TxnStatus_IsError() static bool CF_TxnStatus_IsError (
    CF_TxnStatus_t txn_stat ) [inline], [static]
```

Check if the internal transaction status represents an error.

**Assumptions, External Events, and Notes:**

Transaction status is a superset of condition codes, and includes other error conditions for which CFDP will not send FIN/ACK/EOF and thus there is no corresponding condition code.

**Parameters**

<i>txn_stat</i>	Transaction status
-----------------	--------------------

**Returns**

Boolean value indicating if the transaction is in an errored state

**Return values**

<i>true</i>	if an error has occurred during the transaction
<i>false</i>	if no error has occurred during the transaction yet

Definition at line 521 of file cf\_utils.h.

References CF\_TxnStatus\_NO\_ERROR.

Referenced by CF\_CFDP\_TxnIsOK(), and CF\_TxnStatus\_To\_ConditionCode().

**12.63.3.23 CF\_TxnStatus\_To\_ConditionCode()** [CF\\_CFDP\\_ConditionCode\\_t](#) CF\_TxnStatus\_To\_ConditionCode

(

[CF\\_TxnStatus\\_t](#) *txn\_stat* )

Converts the internal transaction status to a CFDP condition code.

Transaction status is a superset of condition codes, and includes other error conditions for which CFDP will not send FIN/ACK/EOF and thus there is no corresponding condition code.

**Assumptions, External Events, and Notes:**

Not all transaction status codes directly correlate to a CFDP CC

**Parameters**

<i>txn_stat</i>	Transaction status
-----------------	--------------------

**Returns**

CFDP protocol condition code

Definition at line 589 of file cf\_utils.c.

References CF\_CFDP\_ConditionCode\_CANCEL\_REQUEST\_RECEIVED, CF\_CFDP\_ConditionCode\_INACTIVITY\_DETECTED, CF\_CFDP\_ConditionCode\_NO\_ERROR, CF\_TxnStatus\_ACK\_LIMIT\_NO\_EOF, CF\_TxnStatus\_ACK\_LIMIT\_NO\_FIN, CF\_TxnStatus\_CANCEL\_REQUEST\_RECEIVED, CF\_TxnStatus\_CHECK\_LIMIT\_REACHED, CF\_TxnStatus\_FILE\_CHECKSUM\_FAILURE, CF\_TxnStatus\_FILE\_SIZE\_ERROR, CF\_TxnStatus\_FILESTORE\_REJECTION, CF\_TxnStatus\_INACTIVITY\_DETECTED, CF\_TxnStatus\_INVALID\_FILE\_STRUCTURE, CF\_TxnStatus\_INVALID\_TRANSMISSION\_MODE, CF\_TxnStatus\_IsError(), CF\_TxnStatus\_KEEP\_ALIVE\_LIMIT\_REACHED, CF\_TxnStatus\_NAK\_LIMIT\_REACHED, CF\_TxnStatus\_NO\_ERROR, CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED, CF\_TxnStatus\_SUSPEND\_REQUEST\_RECEIVED, and CF\_TxnStatus\_UNSUPPORTED\_CHECKSUM\_TYPE.

Referenced by CF\_CFDP\_SendEof(), and CF\_CFDP\_SendFin().

Here is the call graph for this function:



**12.63.3.24 CF\_WrappedClose()** `void CF_WrappedClose (`  
    `osal_id_t fd )`

Wrap the filesystem close call with a perf counter.

**Assumptions, External Events, and Notes:**

None

**See also**

[OS\\_close\(\)](#) for parameter descriptions

**Parameters**

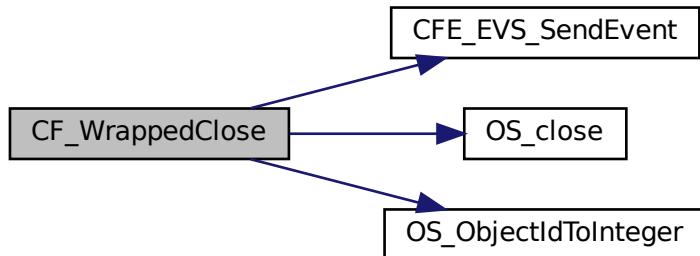
<code>fd</code>	Passed directly to underlying OSAL call
-----------------	---

Definition at line 519 of file cf\_utils.c.

References CF\_CFDP\_CLOSE\_ERR\_EID, CF\_PERF\_ID\_FCLOSE, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, CFE\_EVS\_EventType\_ERROR, CFE\_EVS\_SendEvent(), OS\_close(), OS\_ObjectIdToInteger(), and OS\_SUCCESS.

Referenced by CF\_CFDP\_CloseFiles(), CF\_CFDP\_FinishTransaction(), CF\_CFDP\_R\_HandleFileRetention(), CF\_CFDP\_RecycleTransaction(), CF\_CFDP\_S\_Init(), and CF\_WriteQueueCmd().

Here is the call graph for this function:



```
12.63.3.25 CF_WrappedLseek() CFE_Status_t CF_WrappedLseek (
    osal_id_t fd,
    off_t offset,
    int mode )
```

Wrap the filesystem lseek call with a perf counter.

**Assumptions, External Events, and Notes:**

fname must not be NULL.

**See also**

[OS\\_lseek\(\)](#) for parameter descriptions

**Parameters**

<i>fd</i>	Passed directly to underlying OSAL call
<i>offset</i>	Passed directly to underlying OSAL call
<i>mode</i>	Passed directly to underlying OSAL call

**Returns**

Status code from OSAL (byte count or error code)

Definition at line 572 of file cf\_utils.c.

References CF\_PERF\_ID\_FSEEK, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_lseek().

Referenced by CF\_CFDP\_R\_CalcCrcStart(), CF\_CFDP\_R\_ProcessFd(), CF\_CFDP\_S\_Init(), and CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:



```
12.63.3.26 CF_WrappedOpenCreate() CFE_Status_t CF_WrappedOpenCreate (
    osal_id_t * fd,
    const char * fname,
    int32 flags,
    int32 access )
```

Wrap the filesystem open call with a perf counter.

**Assumptions, External Events, and Notes:**

fname must not be NULL.

**See also**

[OS\\_OpenCreate\(\)](#) for parameter descriptions

**Parameters**

<i>fd</i>	Passed directly to underlying OSAL call
<i>fname</i>	Passed directly to underlying OSAL call
<i>flags</i>	Passed directly to underlying OSAL call
<i>access</i>	Passed directly to underlying OSAL call

**Returns**

Status code from OSAL

Definition at line 503 of file cf\_utils.c.

References CF\_PERF\_ID\_FOPEN, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_OpenCreate().

Referenced by CF\_CFDP\_R\_Init(), CF\_CFDP\_S\_Init(), and CF\_WriteQueueCmd().

Here is the call graph for this function:



**12.63.3.27 CF\_WrappedRead()** `CFE_Status_t CF_WrappedRead (`  
    `osal_id_t fd,`  
    `void * buf,`  
    `size_t read_size )`

Wrap the filesystem read call with a perf counter.

**Assumptions, External Events, and Notes:**

buf must not be NULL.

**See also**

[OS\\_read\(\)](#) for parameter descriptions

**Parameters**

<i>fd</i>	Passed directly to underlying OSAL call
<i>buf</i>	Passed directly to underlying OSAL call
<i>read_size</i>	Passed directly to underlying OSAL call

**Returns**

Status code from OSAL (byte count or error code)

Definition at line 540 of file cf\_utils.c.

References CF\_PERF\_ID\_FREAD, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_read().

Referenced by CF\_CFDP\_R\_CalcCrcChunk(), and CF\_CFDP\_S\_SendFileData().

Here is the call graph for this function:

**12.63.3.28 CF\_WrappedWrite()** [CFE\\_Status\\_t](#) CF\_WrappedWrite (

```
    osal_id_t fd,  
    const void * buf,  
    size_t write_size )
```

Wrap the filesystem write call with a perf counter.

**Assumptions, External Events, and Notes:**

buf must not be NULL.

**See also**

[OS\\_write\(\)](#) for parameter descriptions

**Parameters**

<i>fd</i>	Passed directly to underlying OSAL call
<i>buf</i>	Passed directly to underlying OSAL call
<i>write_size</i>	Passed directly to underlying OSAL call

**Returns**

Status code from OSAL (byte count or error code)

Definition at line 556 of file cf\_utils.c.

References CF\_PERF\_ID\_FWRITE, CFE\_ES\_PerfLogEntry, CFE\_ES\_PerfLogExit, and OS\_write().

Referenced by CF\_CFDP\_R\_ProcessFd(), and CF\_WriteHistoryEntryToFile().

Here is the call graph for this function:



**12.63.3.29 CF\_WriteHistoryEntryToFile()** `CFE_Status_t CF_WriteHistoryEntryToFile (`  
    `osal_id_t fd,`  
    `const CF_History_t * history )`

Write a single history to a file.

This creates a human-readable/string representation of the information in the history object, and writes that string to the indicated file.

This implements the common code between writing the history queue and transaction queue to a file, as both ultimately store the same information in a CF\_History\_t object, but have a different method to get to it.

**Assumptions, External Events, and Notes:**

fd should be a valid file descriptor, open for writing.

**Parameters**

<code>fd</code>	Open File descriptor to write to
<code>history</code>	Pointer to CF history object to write

**Return values**

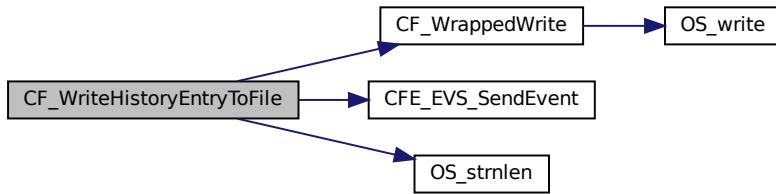
<code>CFE_SUCCESS</code>	on success
<code>CF_ERROR</code>	on error

Definition at line 255 of file cf\_utils.c.

References CF\_ASSERT, CF\_CMD\_WHIST\_WRITE\_ERR\_EID, CF\_Direction\_NUM, CF\_ERROR, CF\_FILENAME\_MAX\_LEN, CF\_WrappedWrite(), CFE\_EVT\_EventType\_ERROR, CFE\_EVT\_SendEvent(), CFE\_SUCCESS, CF\_History::dir, CF\_TxnFilenames::dst\_filename, CF\_History::fnames, OS\_strlen(), CF\_History::peer\_eid, CF\_History::seq\_num, CF\_History::src\_eid, CF\_TxnFilenames::src\_filename, and CF\_History::txn\_stat.

Referenced by CF\_Traverse\_WriteHistoryQueueEntryToFile(), and CF\_Traverse\_WriteTxnQueueEntryToFile().

Here is the call graph for this function:



### 12.63.3.30 CF\_WriteHistoryQueueDataToFile() CFE\_Status\_t CF\_WriteHistoryQueueDataToFile (

```

osal_id_t fd,
CF_Channel_t * chan,
CF_Direction_t dir )

```

Write a history-based queue's entries to a file.

#### Assumptions, External Events, and Notes:

chan must not be NULL.

#### Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>dir</i>	Direction to match/filter

#### Return values

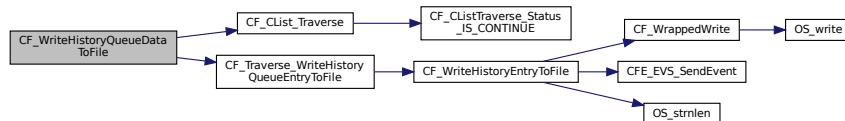
0	on success
1	on error

Definition at line 372 of file cf\_utils.c.

References CF\_CList\_Traverse(), CF\_QueueIdx\_HIST, CF\_Traverse\_WriteHistoryQueueEntryToFile(), CF\_Traverse<->\_WriteHistoryFileArg::counter, CF\_Traverse\_WriteHistoryFileArg::error, CF\_Traverse\_WriteHistoryFileArg::fd, CF\_Traverse<->\_WriteHistoryFileArg::filter\_dir, and CF\_Channel::qs.

Referenced by CF\_WriteQueueCmd().

Here is the call graph for this function:



```
12.63.3.31 CF_WriteTxnQueueDataToFile() CFE_Status_t CF_WriteTxnQueueDataToFile (
    osal_id_t fd,
    CF_Channel_t * chan,
    CF_QueueIdx_t queue )
```

Write a transaction-based queue's transaction history to a file.

**Assumptions, External Events, and Notes:**

chan must not be NULL.

#### Parameters

<i>fd</i>	Open File descriptor to write to
<i>chan</i>	Pointer to associated CF channel object
<i>queue</i>	Queue Index to write

#### Return values

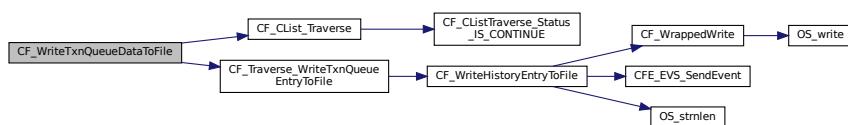
0	on success
1	on error

Definition at line 354 of file cf\_utils.c.

References CF\_CList\_Traverse(), CF\_Traverse\_WriteTxnQueueEntryToFile(), CF\_Traverse\_WriteTxnFileArg::counter, CF\_Traverse\_WriteTxnFileArg::error, CF\_Traverse\_WriteTxnFileArg::fd, and CF\_Channel::qs.

Referenced by CF\_WriteQueueCmd().

Here is the call graph for this function:



## 12.64 apps/cf/fsw/src/cf\_verify.h File Reference

```
#include "cfe.h"
#include "cf_platform_cfg.h"
#include "cf_perfids.h"
```

### 12.64.1 Detailed Description

The CF Application configuration verification header

## 12.65 apps/cf/fsw/src/cf\_version.h File Reference

#### Macros

- #define CF\_MAJOR\_VERSION (7)

- `#define CF_MINOR_VERSION (0)`  
*Minor version number.*
- `#define CF_REVISION (0)`  
*Revision number.*

### 12.65.1 Detailed Description

The CFS CFDP (CF) Application header file containing version number

## 12.66 apps/cf/fsw/tables/cf\_def\_config.c File Reference

```
#include "cfe.h"
#include "cfe_tbl_filedef.h"
#include "cf_tbldefs.h"
```

### Variables

- `CF_ConfigTable_t CF_config_table`

### 12.66.1 Detailed Description

The CF Application default configuration table

### 12.66.2 Variable Documentation

#### 12.66.2.1 CF\_config\_table `CF_ConfigTable_t CF_config_table`

Definition at line 28 of file cf\_def\_config.c.

## 12.67 buildosal\_public\_api/inc/osconfig.h File Reference

### Macros

- `#define OSAL_CONFIG_INCLUDE_DYNAMIC_LOADER`  
*Configuration file Operating System Abstraction Layer.*
- `#define OSAL_CONFIG_INCLUDE_NETWORK`
- `#define OSAL_CONFIG_INCLUDE_STATIC_LOADER`
- `#define OSAL_CONFIG_CONSOLE_ASYNC`
- `#define OS_MAX_TASKS 64`  
*The maximum number of tasks to support.*
- `#define OS_MAX_QUEUES 64`  
*The maximum number of queues to support.*
- `#define OS_MAX_COUNT_SEMAPHORES 20`  
*The maximum number of counting semaphores to support.*
- `#define OS_MAX_BIN_SEMAPHORES 20`  
*The maximum number of binary semaphores to support.*
- `#define OS_MAX_MUTEXES 20`  
*The maximum number of mutexes to support.*
- `#define OS_MAX_RWLOCKS 20`

- `#define OS_MAX_CONDVAR 4`  
*The maximum number of condition variables to support.*
- `#define OS_MAX_MODULES 20`  
*The maximum number of modules to support.*
- `#define OS_MAX_TIMEBASES 5`  
*The maximum number of timebases to support.*
- `#define OS_MAX_TIMERS 10`  
*The maximum number of timer callbacks to support.*
- `#define OS_MAX_NUM_OPEN_FILES 50`  
*The maximum number of concurrently open files to support.*
- `#define OS_MAX_NUM_OPEN_DIRS 4`  
*The maximum number of concurrently open directories to support.*
- `#define OS_MAX_FILE_SYSTEMS 14`  
*The maximum number of file systems to support.*
- `#define OS_MAX_SYM_LEN 64`  
*The maximum length of symbols.*
- `#define OS_MAX_FILE_NAME 20`  
*The maximum length of OSAL file names.*
- `#define OS_MAX_PATH_LEN 64`  
*The maximum length of OSAL path names.*
- `#define OS_MAX_API_NAME 20`  
*The maximum length of OSAL resource names.*
- `#define OS_SOCKADDR_MAX_LEN 28`  
*The maximum size of the socket address structure.*
- `#define OS_BUFFER_SIZE 172`  
*The maximum size of output produced by a single `OS_printf()`*
- `#define OS_BUFFER_MSG_DEPTH 100`  
*The maximum number of `OS_printf()` output strings to buffer.*
- `#define OS_UTILITYTASK_PRIORITY 245`  
*Priority level of the background utility task.*
- `#define OS_UTILITYTASK_STACK_SIZE 2048`  
*The stack size of the background utility task.*
- `#define OS_MAX_CMD_LEN 1000`  
*The maximum size of a shell command.*
- `#define OS_QUEUE_MAX_DEPTH 50`  
*The maximum depth of OSAL queues.*
- `#define OS_SHELL_CMD_INPUT_FILE_NAME ""`  
*The name of the temporary file used to store shell commands.*
- `#define OS_PRINTF_CONSOLE_NAME ""`  
*The name of the primary console device.*
- `#define OS_ADD_TASK_FLAGS 0`  
*Flags added to all tasks on creation.*
- `#define OS_MAX_CONSOLES 1`  
*The maximum number of console devices to support.*
- `#define OS_MODULE_FILE_EXTENSION ".so"`  
*The system-specific file extension used on loadable module files.*
- `#define OS_FS_DEV_NAME_LEN 32`
- `#define OS_FS_PHYS_NAME_LEN 64`
- `#define OS_FS_VOL_NAME_LEN 32`

### 12.67.1 Macro Definition Documentation

#### 12.67.1.1 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 261 of file osconfig.h.

#### 12.67.1.2 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 194 of file osconfig.h.

#### 12.67.1.3 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 187 of file osconfig.h.

#### 12.67.1.4 OS\_FS\_DEV\_NAME\_LEN #define OS\_FS\_DEV\_NAME\_LEN 32

Device name length

Definition at line 288 of file osconfig.h.

#### 12.67.1.5 OS\_FS\_PHYS\_NAME\_LEN #define OS\_FS\_PHYS\_NAME\_LEN 64

Physical drive name length

Definition at line 289 of file osconfig.h.

#### 12.67.1.6 OS\_FS\_VOL\_NAME\_LEN #define OS\_FS\_VOL\_NAME\_LEN 32

Volume name length

Definition at line 290 of file osconfig.h.

#### 12.67.1.7 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 170 of file osconfig.h.

#### 12.67.1.8 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.

**12.67.1.9 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 225 of file osconfig.h.

**12.67.1.10 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 86 of file osconfig.h.

**12.67.1.11 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 276 of file osconfig.h.

**12.67.1.12 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.

**12.67.1.13 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 149 of file osconfig.h.

**12.67.1.14 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 128 of file osconfig.h.

**12.67.1.15 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 93 of file osconfig.h.

**12.67.1.16 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.

**12.67.1.17 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 121 of file osconfig.h.

**12.67.1.18 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 114 of file osconfig.h.

**12.67.1.19 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 161 of file osconfig.h.

**12.67.1.20 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.

**12.67.1.21 OS\_MAX\_RWLOCKS** #define OS\_MAX\_RWLOCKS 20

The maximum number of rwlocks to support.

Based on the OSAL\_CONFIG\_MAX\_RWLOCKS configuration option

Definition at line 79 of file osconfig.h.

**12.67.1.22 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 137 of file osconfig.h.

**12.67.1.23 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.

**12.67.1.24 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 100 of file osconfig.h.

**12.67.1.25 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 107 of file osconfig.h.

**12.67.1.26 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 283 of file osconfig.h.

**12.67.1.27 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 252 of file osconfig.h.

**12.67.1.28 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 232 of file osconfig.h.

**12.67.1.29 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 242 of file osconfig.h.

**12.67.1.30 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 180 of file osconfig.h.

**12.67.1.31 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 204 of file osconfig.h.

**12.67.1.32 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 214 of file osconfig.h.

**12.67.1.33 OSAL\_CONFIG\_CONSOLE\_ASYNC** #define OSAL\_CONFIG\_CONSOLE\_ASYNC

Definition at line 27 of file osconfig.h.

**12.67.1.34 OSAL\_CONFIG\_INCLUDE\_DYNAMIC\_LOADER** #define OSAL\_CONFIG\_INCLUDE\_DYNAMIC\_LOADER

Configuration file Operating System Abstraction Layer.

The specific definitions in this file may only be modified by setting the respective OSAL configuration options in the CMake build.

Any direct modifications to the generated copy will be overwritten each time CMake executes.

**Note**

This file was automatically generated by CMake from /home/runner/work/CF/CF/osal/default\_config.cmake

Definition at line 21 of file osconfig.h.

**12.67.1.35 OSAL\_CONFIG\_INCLUDE\_NETWORK** #define OSAL\_CONFIG\_INCLUDE\_NETWORK

Definition at line 22 of file osconfig.h.

**12.67.1.36 OSAL\_CONFIG\_INCLUDE\_STATIC\_LOADER** #define OSAL\_CONFIG\_INCLUDE\_STATIC\_LOADER

Definition at line 23 of file osconfig.h.

## 12.68 cfe/cmake/sample\_defs/cfe\_perfids.h File Reference

### Macros

- #define CFE\_MISSION\_ES\_PERF\_EXIT\_BIT 31

*bit (31) is reserved by the perf utilities*

#### cFE Performance Monitor IDs (Reserved IDs 0-31)

- #define CFE\_MISSION\_ES\_MAIN\_PERF\_ID 1

*Performance ID for Executive Services Task.*

- #define CFE\_MISSION\_EVS\_MAIN\_PERF\_ID 2

*Performance ID for Events Services Task.*

- #define CFE\_MISSION\_TBL\_MAIN\_PERF\_ID 3

*Performance ID for Table Services Task.*

- #define CFE\_MISSION\_SB\_MAIN\_PERF\_ID 4  
*Performance ID for Software Bus Services Task.*
- #define CFE\_MISSION\_SB\_MSG\_LIM\_PERF\_ID 5  
*Performance ID for Software Bus Msg Limit Errors.*
- #define CFE\_MISSION\_SB\_PIPE\_OFLOW\_PERF\_ID 27  
*Performance ID for Software Bus Pipe Overflow Errors.*
- #define CFE\_MISSION\_TIME\_MAIN\_PERF\_ID 6  
*Performance ID for Time Services Task.*
- #define CFE\_MISSION\_TIME\_TONE1HZISR\_PERF\_ID 7  
*Performance ID for 1 Hz Tone ISR.*
- #define CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID 8  
*Performance ID for 1 Hz Local ISR.*
- #define CFE\_MISSION\_TIME\_SENDMET\_PERF\_ID 9  
*Performance ID for Time ToneSendMET.*
- #define CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID 10  
*Performance ID for 1 Hz Local Task.*
- #define CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID 11  
*Performance ID for 1 Hz Tone Task.*

### 12.68.1 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:

### 12.68.2 Macro Definition Documentation

#### 12.68.2.1 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 cfe\_perfids.h.

#### 12.68.2.2 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 cfe\_perfids.h.

#### 12.68.2.3 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 cfe\_perfids.h.

#### 12.68.2.4 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 cfe\_perfids.h.

#### 12.68.2.5 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 cfe\_perfids.h.

**12.68.2.6 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 cfe\_perfids.h.

**12.68.2.7 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 cfe\_perfids.h.

**12.68.2.8 CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID** #define CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID 8  
Performance ID for 1 Hz Local ISR.  
Definition at line 51 of file cfe\_perfids.h.

**12.68.2.9 CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID** #define CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID 10  
Performance ID for 1 Hz Local Task.  
Definition at line 54 of file cfe\_perfids.h.

**12.68.2.10 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 cfe\_perfids.h.

**12.68.2.11 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 cfe\_perfids.h.

**12.68.2.12 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 cfe\_perfids.h.

**12.68.2.13 CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID** #define CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID 11  
Performance ID for 1 Hz Tone Task.  
Definition at line 55 of file cfe\_perfids.h.

## 12.69 cfe/cmake/sample\_defs/example\_2x32bit\_cfe\_es\_memaddress.h File Reference

```
#include "common_types.h"
```

### Data Structures

- struct **CFE\_ES\_MemOffset\_t**  
*Type used for memory sizes and offsets in commands and telemetry.*
- struct **CFE\_ES\_MemAddress\_t**

Type used for memory addresses in command and telemetry messages.

## Macros

- `#define CFE_ES_MEMOFFSET_C(x) CFE_ES_MemOffset_FromNative(x)`  
*Memory Offset initializer wrapper.*
- `#define CFE_ES_MEMOFFSET_TO_SIZET(x) CFE_ES_MemOffset_ToNative(&(x))`  
*Memory Offset to integer value (size\_t) wrapper.*
- `#define CFE_ES_MEMADDRESS_C(x) CFE_ES_MemAddress_FromNative((cpuaddr)(x))`  
*Memory Address initializer wrapper.*
- `#define CFE_ES_MEMADDRESS_TO_PTR(x) ((void *)CFE_ES_MemAddress_ToNative(&(x)))`  
*Memory Address to pointer wrapper.*

## Functions

- `static size_t CFE_ES_MemOffset_ToNative (const CFE_ES_MemOffset_t *val)`
- `static CFE_ES_MemOffset_t CFE_ES_MemOffset_FromNative (size_t val)`
- `static cpuaddr CFE_ES_MemAddress_ToNative (const CFE_ES_MemAddress_t *val)`
- `static CFE_ES_MemAddress_t CFE_ES_MemAddress_FromNative (cpuaddr val)`

### 12.69.1 Detailed Description

Defines memory addresses and offsets to be 64 bit integer values

This expands the traditional 32 bit memory addresses in commands and telemetry out to 64 bits for compatibility with modern CPUs, but does so as a pair of 32 bit values rather than a single 64 bit value. This is done to avoid the introduction of implicit padding if addresses or sizes are not aligned at 64 bit offsets within the respective parent structure. This does necessitate that all access to these values is done via the provided conversion macros. Attempts to directly assign these values to an integer type will fail to compile.

### 12.69.2 Macro Definition Documentation

#### 12.69.2.1 CFE\_ES\_MEMADDRESS\_C `#define CFE_ES_MEMADDRESS_C (` `x ) CFE_ES_MemAddress_FromNative ((cpuaddr) (x))`

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 144 of file example\_2x32bit\_cfe\_es\_memaddress.h.

#### 12.69.2.2 CFE\_ES\_MEMADDRESS\_TO\_PTR `#define CFE_ES_MEMADDRESS_TO_PTR (` `x ) ((void *)CFE_ES_MemAddress_ToNative (& (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 152 of file example\_2x32bit\_cfe\_es\_memaddress.h.

#### 12.69.2.3 CFE\_ES\_MEMOFFSET\_C `#define CFE_ES_MEMOFFSET_C (` `x ) CFE_ES_MemOffset_FromNative (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 88 of file example\_2x32bit\_cfe\_es\_memaddress.h.

**12.69.2.4 CFE\_ES\_MEMOFFSET\_TO\_SIZE\_T** #define CFE\_ES\_MEMOFFSET\_TO\_SIZE\_T( x ) CFE\_ES\_MemOffset\_ToNative( &(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 96 of file example\_2x32bit\_cfe\_es\_memaddress.h.

### 12.69.3 Function Documentation

**12.69.3.1 CFE\_ES\_MemAddress\_FromNative()** static CFE\_ES\_MemAddress\_t CFE\_ES\_MemAddress\_FromNative(

cpuaddr val ) [inline], [static]

Definition at line 130 of file example\_2x32bit\_cfe\_es\_memaddress.h.

References CFE\_ES\_MemAddress\_t::bits.

**12.69.3.2 CFE\_ES\_MemAddress\_ToNative()** static cpuaddr CFE\_ES\_MemAddress\_ToNative( const CFE\_ES\_MemAddress\_t \* val ) [inline], [static]

Definition at line 122 of file example\_2x32bit\_cfe\_es\_memaddress.h.

References CFE\_ES\_MemAddress\_t::bits.

**12.69.3.3 CFE\_ES\_MemOffset\_FromNative()** static CFE\_ES\_MemOffset\_t CFE\_ES\_MemOffset\_FromNative( size\_t val ) [inline], [static]

Definition at line 73 of file example\_2x32bit\_cfe\_es\_memaddress.h.

References CFE\_ES\_MemOffset\_t::bits.

**12.69.3.4 CFE\_ES\_MemOffset\_ToNative()** static size\_t CFE\_ES\_MemOffset\_ToNative( const CFE\_ES\_MemOffset\_t \* val ) [inline], [static]

Definition at line 65 of file example\_2x32bit\_cfe\_es\_memaddress.h.

References CFE\_ES\_MemOffset\_t::bits.

## 12.70 cfe/cmake/sample\_defs/example\_32bit\_cfe\_es\_memaddress.h File Reference

```
#include "common_types.h"
```

### 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 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.*

### 12.70.1 Detailed Description

Example header file override that defines memory addresses and offsets to be a 32 bit integer values. This is backward compatible with prior CFS versions where all memory references were assumed to be 32 bits in size. It is still possible to execute this on a 64-bit CPU, but all address will be truncated to the lower 32 bits. One will also not be able to send commands that require a memory address, as the address will be incomplete. To use this implementation, clone this file as "cfe\_es\_memaddress.h" in your local defs dir.

### 12.70.2 Macro Definition Documentation

#### 12.70.2.1 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 102 of file example\_32bit\_cfe\_es\_memaddress.h.

#### 12.70.2.2 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 110 of file example\_32bit\_cfe\_es\_memaddress.h.

#### 12.70.2.3 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 65 of file example\_32bit\_cfe\_es\_memaddress.h.

#### 12.70.2.4 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 73 of file example\_32bit\_cfe\_es\_memaddress.h.

### 12.70.3 Typedef Documentation

**12.70.3.1 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 94 of file example\_32bit\_cfe\_es\_memaddress.h.

**12.70.3.2 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 57 of file example\_32bit\_cfe\_es\_memaddress.h.

**12.71 cfe/cmake/sample\_defs/example\_64bit\_cfe\_es\_memaddress.h File Reference**

```
#include "common_types.h"
```

**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))`  
*Memory Address initializer wrapper.*
- `#define CFE_ES_MEMADDRESS_TO_PTR(x) ((void *) (cpuaddr)(x))`  
*Memory Address to pointer wrapper.*

**Typedefs**

- `typedef uint64 CFE_ES_MemOffset_t`  
*Type used for memory sizes and offsets in commands and telemetry.*
- `typedef uint64 CFE_ES_MemAddress_t`  
*Type used for memory addresses in command and telemetry messages.*

**12.71.1 Detailed Description**

Example header file override that defines memory addresses and offsets to be a full 64 bit integer value.

This is the simplest and most efficient approach to use 64 bit addressing, but may introduce unexpected padding in certain cases. On most systems a uint64 value needs to be aligned on a 64-bit boundary, and CFS has some TLM and CMD structures where this is not the case. These will be padded by the compiler and may cause interoperability issues. To use this implementation, clone this file as "cfe\_es\_memaddress.h" in your local defs dir.

### 12.71.2 Macro Definition Documentation

**12.71.2.1 CFE\_ES\_MEMADDRESS\_C** `#define CFE_ES_MEMADDRESS_C (`  
`x ) ((CFE_ES_MemAddress_t)(cpuaddr))(x))`

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 81 of file example\_64bit\_cfe\_es\_memaddress.h.

**12.71.2.2 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 89 of file example\_64bit\_cfe\_es\_memaddress.h.

**12.71.2.3 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 58 of file example\_64bit\_cfe\_es\_memaddress.h.

**12.71.2.4 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 66 of file example\_64bit\_cfe\_es\_memaddress.h.

### 12.71.3 Typedef Documentation

**12.71.3.1 CFE\_ES\_MemAddress\_t** `typedef uint64 CFE_ES_MemAddress_t`

Type used for memory addresses in command and telemetry messages.

Use a full 64-bit integer value for memory offsets

Definition at line 73 of file example\_64bit\_cfe\_es\_memaddress.h.

**12.71.3.2 CFE\_ES\_MemOffset\_t** `typedef uint64 CFE_ES_MemOffset_t`

Type used for memory sizes and offsets in commands and telemetry.

Use a full 64-bit integer value for memory offsets

Definition at line 50 of file example\_64bit\_cfe\_es\_memaddress.h.

## 12.72 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 32
- #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

##### 12.72.1 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/TLM 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.

## 12.72.2 Macro Definition Documentation

**12.72.2.1 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.

**12.72.2.2 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.

**12.72.2.3 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.

**12.72.2.4 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.

**12.72.2.5 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.

**12.72.2.6 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.

**12.72.2.7 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.

**12.72.2.8 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.

**12.72.2.9 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.

**12.72.2.10 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.

**12.72.2.11 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.

**12.72.2.12 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.

**12.72.2.13 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.

**12.72.2.14 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.

**12.72.2.15 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.

**12.72.2.16 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.

**12.72.2.17 CFE\_MISSION\_SB\_MAX\_PIPES #define CFE\_MISSION\_SB\_MAX\_PIPES 32****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.

**12.72.2.18 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.

```
12.72.2.19 CFE_MISSION_TBL_MAX_FULL_NAME_LEN #define CFE_MISSION_TBL_MAX_FULL_NAME_LEN (CFE_MISSION_TBL_MA  
+ 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.

```
12.72.2.20 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.

```
12.72.2.21 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.

**12.72.2.22 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.

**12.72.2.23 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.

**12.72.2.24 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC** #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC false  
Definition at line 433 of file example\_mission\_cfg.h.

**12.72.2.25 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.

**12.72.2.26 CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS 0  
Definition at line 527 of file example\_mission\_cfg.h.

**12.72.2.27 CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS 1000  
Definition at line 528 of file example\_mission\_cfg.h.

**12.72.2.28 CFE\_MISSION\_TIME\_DEF\_LEAPS** #define CFE\_MISSION\_TIME\_DEF\_LEAPS 37  
Definition at line 525 of file example\_mission\_cfg.h.

**12.72.2.29 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.

**12.72.2.30 CFE\_MISSION\_TIME\_DEF\_MET\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_MET\_SUBS 0  
Definition at line 520 of file example\_mission\_cfg.h.

**12.72.2.31 CFE\_MISSION\_TIME\_DEF\_STCF\_SECS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SECS 1000000  
Definition at line 522 of file example\_mission\_cfg.h.

**12.72.2.32 CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS 0  
Definition at line 523 of file example\_mission\_cfg.h.

**12.72.2.33 CFE\_MISSION\_TIME\_EPOCH\_DAY** #define CFE\_MISSION\_TIME\_EPOCH\_DAY 1  
Definition at line 546 of file example\_mission\_cfg.h.

**12.72.2.34 CFE\_MISSION\_TIME\_EPOCH\_HOUR** #define CFE\_MISSION\_TIME\_EPOCH\_HOUR 0  
Definition at line 547 of file example\_mission\_cfg.h.

**12.72.2.35 CFE\_MISSION\_TIME\_EPOCH\_MICROS** #define CFE\_MISSION\_TIME\_EPOCH\_MICROS 0  
Definition at line 550 of file example\_mission\_cfg.h.

**12.72.2.36 CFE\_MISSION\_TIME\_EPOCH\_MINUTE** #define CFE\_MISSION\_TIME\_EPOCH\_MINUTE 0  
Definition at line 548 of file example\_mission\_cfg.h.

**12.72.2.37 CFE\_MISSION\_TIME\_EPOCH\_SECOND** #define CFE\_MISSION\_TIME\_EPOCH\_SECOND 0  
Definition at line 549 of file example\_mission\_cfg.h.

**12.72.2.38 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.

**12.72.2.39 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.

**12.72.2.40 CFE\_MISSION\_TIME\_MAX\_ELAPSED** #define CFE\_MISSION\_TIME\_MAX\_ELAPSED 200000

Definition at line 494 of file example\_mission\_cfg.h.

**12.72.2.41 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.

**12.73 cfe/cmake/sample\_defs/example\_platform\_cfg.h File Reference**

**Macros**

- #define CFE\_PLATFORM\_ENDIAN CCSDS\_LITTLE\_ENDIAN
- #define CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC 30000
- #define CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY 68
- #define CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING "/cf"
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING "/ram"
- #define CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS 32
- #define CFE\_PLATFORM\_ES\_MAX\_LIBRARIES 10
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES 20
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE 256
- #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072
- #define CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE 30
- #define CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS 8
- #define CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE 1000
- #define CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT 5
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE 512
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS 4096
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED 30
- #define CFE\_PLATFORM\_ES\_CDS\_SIZE (128 \* 1024)
- #define CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE (1024 \* 1024)
- #define CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN 4
- #define CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE "/cf/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE "/ram/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE "/ram/cfe\_es\_app\_info.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE "/ram/cfe\_es\_taskinfo.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE "/ram/cfe\_es\_syslog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE "/ram/cfe\_erlog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME "/ram/cfe\_es\_perf.dat"

- #define CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE "/ram/cfe\_cds\_reg.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE 0
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE 1
- #define CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE 10000
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE 0
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL ~CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE 0
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL ~CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY 200
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY 20
- #define CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS 50
- #define CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES 512
- #define CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS 2
- #define CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS 17
- #define CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS 10
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 32
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 32
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384

- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC 50
- #define CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC 1000
- #define CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY 61
- #define CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS 8
- #define CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST 32
- #define CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC 15
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE "/ram/cfe\_evs.log"
- #define CFE\_PLATFORM\_EVS\_LOG\_MAX 20
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE "/ram/cfe\_evs\_app.dat"
- #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT 0x0001
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG 0xE
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE 1
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE CFE\_EVS\_MsgFormat\_LONG
- #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256
- #define CFE\_PLATFORM\_SB\_MAX\_PIPES 64
- #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4
- #define CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES 524288
- #define CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID 0x1FFF
- #define CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME "/ram/cfe\_sb\_route.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME "/ram/cfe\_sb\_pipe.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME "/ram/cfe\_sb\_msgmap.dat"
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_SB\_SEND\_NO\_SUBS\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_EVS\_FIRST\_4\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_SB\_DUP\_SUBSCRIP\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_EVS\_FIRST\_4\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT3 CFE\_SB\_MSGID\_LIM\_ERR\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_EVS\_FIRST\_16\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT4 CFE\_SB\_Q\_FULL\_ERR\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_EVS\_FIRST\_16\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT5 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT6 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT7 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT8 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK8 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 20
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 36
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 512

- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE (CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE + 128)
- #define CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY 64
- #define CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY 70
- #define CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TBL\_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\_ONEHZ\_TASK\_PRIORITY 25
- #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE 8192

### 12.73.1 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.

### 12.73.2 Macro Definition Documentation

**12.73.2.1 CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC** `#define CFE_PLATFORM_CORE_MAX_STARTUP_←  
MSEC 30000`

**Purpose** CFE core application startup timeout

#### Description:

The upper limit for the amount of time that the cFE core applications (ES, SB, EVS, TIME, TBL) are each allotted to reach their respective "ready" states.

The CFE "main" thread starts individual tasks for each of the core applications (except FS). Each of these must perform some initialization work before the next core application can be started, so the main thread waits to ensure that the application has reached the "ready" state before starting the next application.

If any core application fails to start, then it indicates a major problem with the system and startup is aborted.

Units are in milliseconds

#### Limits:

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 84 of file example\_platform\_cfg.h.

**12.73.2.2 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.

**12.73.2.3 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 288 of file example\_platform\_cfg.h.

**12.73.2.4 CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE** #define CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE 1000

**Purpose** Define ES Application Control Scan Rate

**Description:**

ES Application Control Scan Rate. This parameter controls the speed that ES scans the Application Table looking for App Delete/Restart/Reload requests. All Applications are deleted, restarted, or reloaded by the ES Application. ES will periodically scan for control requests to process. The scan rate is controlled by this parameter, which is given in milliseconds. A value of 1000 means that ES will scan the Application Table once per second. Be careful not to set the value of this too low, because ES will use more CPU cycles scanning the table.

**Limits**

There is a lower limit of 100 and an upper limit of 20000 on this configuration parameter. millisecond units.

Definition at line 259 of file example\_platform\_cfg.h.

**12.73.2.5 CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE 80000

Definition at line 830 of file example\_platform\_cfg.h.

```
12.73.2.6 CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES #define CFE_PLATFORM_ES_CDS_MAX_NUM_←  
ENTRIES 512
```

**Purpose** Define Maximum Number of Registered CDS Blocks

Description:

Maximum number of registered CDS Blocks

Limits

There is a lower limit of 8. There are no restrictions on the upper limit however, the maximum number of CDS entries is system dependent and should be verified.

Definition at line 720 of file example\_platform\_cfg.h.

```
12.73.2.7 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_01 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_←  
SIZE_01 8
```

**Purpose** Define ES Critical Data Store Memory Pool Block Sizes

Description:

Intermediate ES Critical Data Store Memory Pool Block Sizes

Limits

These sizes MUST be increasing and MUST be an integral multiple of 4.

Definition at line 814 of file example\_platform\_cfg.h.

```
12.73.2.8 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_←  
SIZE_02 16
```

Definition at line 815 of file example\_platform\_cfg.h.

```
12.73.2.9 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_←  
SIZE_03 32
```

Definition at line 816 of file example\_platform\_cfg.h.

```
12.73.2.10 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_←  
SIZE_04 48
```

Definition at line 817 of file example\_platform\_cfg.h.

```
12.73.2.11 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_←  
SIZE_05 64
```

Definition at line 818 of file example\_platform\_cfg.h.

**12.73.2.12 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96

Definition at line 819 of file example\_platform\_cfg.h.

**12.73.2.13 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128

Definition at line 820 of file example\_platform\_cfg.h.

**12.73.2.14 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160

Definition at line 821 of file example\_platform\_cfg.h.

**12.73.2.15 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256

SIZE\_09 256

Definition at line 822 of file example\_platform\_cfg.h.

**12.73.2.16 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512

SIZE\_10 512

Definition at line 823 of file example\_platform\_cfg.h.

**12.73.2.17 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024

SIZE\_11 1024

Definition at line 824 of file example\_platform\_cfg.h.

**12.73.2.18 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048

SIZE\_12 2048

Definition at line 825 of file example\_platform\_cfg.h.

**12.73.2.19 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096

SIZE\_13 4096

Definition at line 826 of file example\_platform\_cfg.h.

**12.73.2.20 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192

SIZE\_14 8192

Definition at line 827 of file example\_platform\_cfg.h.

**12.73.2.21 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384

SIZE\_15 16384

Definition at line 828 of file example\_platform\_cfg.h.

```
12.73.2.22 CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 #define CFE_PLATFORM_ES_CDS_MEM_BLOCK_SIZE_16 32768
```

Definition at line 829 of file example\_platform\_cfg.h.

```
12.73.2.23 CFE_PLATFORM_ES_CDS_SIZE #define CFE_PLATFORM_ES_CDS_SIZE (128 * 1024)
```

**Purpose** Define Critical Data Store Size

**Description:**

Defines the Critical Data Store (CDS) area size in bytes size. The CDS is one of four memory areas that are preserved during a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as USER\_RESERVED\_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

**Limits**

There is a lower limit of 8192 and an upper limit of `UINT_MAX` (4 Gigabytes) on this configuration parameter.

Definition at line 365 of file example\_platform\_cfg.h.

```
12.73.2.24 CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE #define CFE_PLATFORM_ES_DEFAULT_APP_LOG_FILE "/ram/cfe_es_app_info.log"
```

**Purpose** Default Application Information Filename

**Description:**

The value of this constant defines the filename used to store information pertaining to all of the Applications that are registered with Executive Services. This filename is used only when no filename is specified in the command to query all system apps.

**Limits**

The length of each string, including the NULL terminator cannot exceed the `OS_MAX_PATH_LEN` value.

Definition at line 447 of file example\_platform\_cfg.h.

```
12.73.2.25 CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE #define CFE_PLATFORM_ES_DEFAULT_CDS_REG_DUMP_FILE "/ram/cfe_cds_reg.log"
```

**Purpose** Default Critical Data Store Registry Filename

**Description:**

The value of this constant defines the filename used to store the Critical Data Store Registry. This filename is used only when no filename is specified in the command to stop performance data collecting.

**Limits**

The length of each string, including the NULL terminator cannot exceed the `OS_MAX_PATH_LEN` value.

Definition at line 521 of file example\_platform\_cfg.h.

**12.73.2.26 CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_←  
FILE "/ram/cfe\_erlog.log"

**Purpose** Default Exception and Reset (ER) Log Filename

**Description:**

The value of this constant defines the filename used to store the Exception and Reset (ER) Log. This filename is used only when no filename is specified in the command to dump the ER log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 493 of file example\_platform\_cfg.h.

**12.73.2.27 CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME** #define CFE\_PLATFORM\_ES\_DEFAULT\_←  
PERF\_DUMP\_FILENAME "/ram/cfe\_es\_perf.dat"

**Purpose** Default Performance Data Filename

**Description:**

The value of this constant defines the filename used to store the Performance Data. This filename is used only when no filename is specified in the command to stop performance data collecting.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 507 of file example\_platform\_cfg.h.

**12.73.2.28 CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE** #define CFE\_PLATFORM\_ES\_DEFAULT\_POR\_←  
\_SYSLOG\_MODE 0

**Purpose** Define Default System Log Mode following Power On Reset

**Description:**

Defines the default mode for the operation of the ES System log following a power on reset. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest message in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. This constant may hold a value of either 0 or 1 depending on the desired default. Overwrite Mode = 0, Discard Mode = 1.

**Limits**

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 539 of file example\_platform\_cfg.h.

```
12.73.2.29 CFE_PLATFORM_ES_DEFAULT_PR_SYSLOG_MODE #define CFE_PLATFORM_ES_DEFAULT_PR_←  
SYSLOG_MODE 1
```

**Purpose** Define Default System Log Mode following Processor Reset

Description:

Defines the default mode for the operation of the ES System log following a processor reset. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest message in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. This constant may hold a value of either 0 or 1 depending on the desired default. Overwrite Mode = 0, Discard Mode = 1.

Limits

There is a lower limit of 0 and an upper limit of 1 on this configuration parameter.

Definition at line 557 of file example\_platform\_cfg.h.

```
12.73.2.30 CFE_PLATFORM_ES_DEFAULT_STACK_SIZE #define CFE_PLATFORM_ES_DEFAULT_STACK_←  
SIZE 8192
```

**Purpose** Define Default Stack Size for an Application

Description:

This parameter defines a default stack size. This parameter is used by the cFE Core Applications.

Limits

There is a lower limit of 2048. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 707 of file example\_platform\_cfg.h.

```
12.73.2.31 CFE_PLATFORM_ES_DEFAULT_SYSLOG_FILE #define CFE_PLATFORM_ES_DEFAULT_SYSLOG_←  
FILE "/ram/cfe_es_syslog.log"
```

**Purpose** Default System Log Filename

Description:

The value of this constant defines the filename used to store important information (as ASCII text strings) that might not be able to be sent in an Event Message. This filename is used only when no filename is specified in the command to dump the system log. No file specified in the cmd means the first character in the cmd filename is a NULL terminator (zero).

Limits

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 478 of file example\_platform\_cfg.h.

**12.73.2.32 CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE "/ram/cfe\_es\_taskinfo.log"

**Purpose** Default Application Information Filename

**Description:**

The value of this constant defines the filename used to store information pertaining to all of the Applications that are registered with Executive Services. This filename is used only when no filename is specified in the the command to query all system tasks.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 462 of file example\_platform\_cfg.h.

**12.73.2.33 CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES** #define CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES 20

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

**Description:**

Defines the maximum number of ER (Exception and Reset) log entries

**Limits**

There is a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of log entries is system dependent and should be verified.

Definition at line 186 of file example\_platform\_cfg.h.

**12.73.2.34 CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE** #define CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE 256

**Purpose** Maximum size of CPU Context in ES Error Log

**Description:**

This should be large enough to accommodate the CPU context information supplied by the PSP on the given platform.

**Limits:**

Must be greater than zero and a multiple of sizeof(uint32). Limited only by the available memory and the number of entries in the error log. Any context information beyond this size will be truncated.

Definition at line 200 of file example\_platform\_cfg.h.

**12.73.2.35 CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS** #define CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS 32

**Purpose** Define Max Number of Applications

**Description:**

Defines the maximum number of applications that can be loaded into the system. This number does not include child tasks.

**Limits**

There is a lower limit of 6. The lower limit corresponds to the cFE internal applications. There are no restrictions on the upper limit however, the maximum number of applications is system dependent and should be verified. AppIDs that are checked against this configuration are defined by a 32 bit data word.

Definition at line 159 of file example\_platform\_cfg.h.

**12.73.2.36 CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000

Definition at line 803 of file example\_platform\_cfg.h.

**12.73.2.37 CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS** #define CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS 8

**Purpose** Define Max Number of Generic Counters

**Description:**

Defines the maximum number of Generic Counters that can be registered.

**Limits**

This parameter has a lower limit of 1 and an upper limit of 65535.

Definition at line 240 of file example\_platform\_cfg.h.

**12.73.2.38 CFE\_PLATFORM\_ES\_MAX\_LIBRARIES** #define CFE\_PLATFORM\_ES\_MAX\_LIBRARIES 10

**Purpose** Define Max Number of Shared libraries

**Description:**

Defines the maximum number of cFE Shared libraries that can be loaded into the system.

**Limits**

There is a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of libraries is system dependent and should be verified.

Definition at line 173 of file example\_platform\_cfg.h.

**12.73.2.39 CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS** #define CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS 10

**Purpose** Maximum number of memory pools

**Description:**

The upper limit for the number of memory pools that can concurrently exist within the system.

The CFE\_SB and CFE\_TBL core subsystems each define a memory pool. Individual applications may also create memory pools, so this value should be set sufficiently high enough to support the applications being used on this platform.

**Limits:**

Must be at least 2 to support CFE core - SB and TBL pools. No specific upper limit.

Definition at line 768 of file example\_platform\_cfg.h.

**12.73.2.40 CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS** #define CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS 2

**Purpose** Define Number of Processor Resets Before a Power On Reset

**Description:**

Number of Processor Resets before a Power On Reset is called. If set to 2, then 2 processor resets will occur, and the 3rd processor reset will be a power on reset instead.

**Limits**

There is a lower limit of 0. There are no restrictions on the upper limit however, the maximum number of processor resets may be system dependent and should be verified.

Definition at line 735 of file example\_platform\_cfg.h.

**12.73.2.41 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 8

**Purpose** Define Default ES Memory Pool Block Sizes

**Description:**

Default Intermediate ES Memory Pool Block Sizes. If an application is using the CFE\_ES Memory Pool APIs ([CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_GetPoolBuf](#) and [CFE\\_ES\\_PutPoolBuf](#)) but finds these sizes inappropriate for their use, they may wish to use the [CFE\\_ES\\_PoolCreateEx](#) API to specify their own intermediate block sizes

**Limits**

These sizes MUST be increasing and MUST be an integral multiple of 4. Also, CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE must be larger than CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE and both CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE and CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE. Note that if Table Services have been removed from the CFE, the table size limits are still enforced although the table size definitions may be reduced.

Definition at line 787 of file example\_platform\_cfg.h.

**12.73.2.42 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16  
Definition at line 788 of file example\_platform\_cfg.h.

**12.73.2.43 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 32  
Definition at line 789 of file example\_platform\_cfg.h.

**12.73.2.44 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 48  
Definition at line 790 of file example\_platform\_cfg.h.

**12.73.2.45 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 64  
Definition at line 791 of file example\_platform\_cfg.h.

**12.73.2.46 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 96  
Definition at line 792 of file example\_platform\_cfg.h.

**12.73.2.47 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 128  
Definition at line 793 of file example\_platform\_cfg.h.

**12.73.2.48 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 160  
Definition at line 794 of file example\_platform\_cfg.h.

**12.73.2.49 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 256  
Definition at line 795 of file example\_platform\_cfg.h.

**12.73.2.50 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 512  
Definition at line 796 of file example\_platform\_cfg.h.

**12.73.2.51 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 1024  
Definition at line 797 of file example\_platform\_cfg.h.

**12.73.2.52 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 2048  
Definition at line 798 of file example\_platform\_cfg.h.

**12.73.2.53 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 4096  
Definition at line 799 of file example\_platform\_cfg.h.

**12.73.2.54 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 8192  
14 8192  
Definition at line 800 of file example\_platform\_cfg.h.

**12.73.2.55 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 16384  
15 16384  
Definition at line 801 of file example\_platform\_cfg.h.

**12.73.2.56 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 32768  
16 32768  
Definition at line 802 of file example\_platform\_cfg.h.

**12.73.2.57 CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN** #define CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN 4

**Purpose** Define Memory Pool Alignment Size

**Description:**

Ensures that buffers obtained from a memory pool are aligned to a certain minimum block size. Note the allocator will always align to the minimum required by the CPU architecture. This may be set greater than the CPU requirement as desired for optimal performance.

For some architectures/applications it may be beneficial to set this to the cache line size of the target CPU, or to use special SIMD instructions that require a more stringent memory alignment.

**Limits**

This must always be a power of 2, as it is used as a binary address mask.

Definition at line 404 of file example\_platform\_cfg.h.

**12.73.2.58 CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING** #define CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING "/cf"

**Purpose** Default virtual path for persistent storage

**Description:**

This configures the default location in the virtual file system for persistent/non-volatile storage. Files such as the startup script, app/library dynamic modules, and configuration tables are expected to be stored in this directory.

Definition at line 127 of file example\_platform\_cfg.h.

**12.73.2.59 CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE** #define CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE "/cf/cfe\_es\_startup.scr"

**Purpose** ES Nonvolatile Startup Filename

**Description:**

The value of this constant defines the path and name of the file that contains a list of modules that will be loaded and started by the cFE after the cFE finishes its startup sequence.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 418 of file example\_platform\_cfg.h.

**12.73.2.60 CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE** `#define CFE_PLATFORM_ES_OBJECT_TABLE_SIZE 30`**Purpose** Define Number of entries in the ES Object table**Description:**

Defines the number of entries in the ES Object table. This table controls the core cFE startup.

**Limits**

There is a lower limit of 15. There are no restrictions on the upper limit however, the maximum object table size is system dependent and should be verified.

Definition at line 229 of file example\_platform\_cfg.h.

**12.73.2.61 CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY** `#define CFE_PLATFORM_ES_PERF_CHILD_MS_DELAY 20`**Purpose** Define Performance Analyzer Child Task Delay**Description:**

This parameter defines the delay time (in milliseconds) between performance data file writes performed by the Executive Services Performance Analyzer Child Task.

**Limits**

It is recommended this parameter be greater than or equal to 20ms. This parameter is limited by the maximum value allowed by the data type. In this case, the data type is an unsigned 32-bit integer, so the valid range is 0 to 0xFFFFFFFF.

Definition at line 681 of file example\_platform\_cfg.h.

**12.73.2.62 CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY** `#define CFE_PLATFORM_ES_PERF_CHILD_PRIORITY 200`**Purpose** Define Performance Analyzer Child Task Priority**Description:**

This parameter defines the priority of the child task spawned by the Executive Services to write performance data to a file. Lower numbers are higher priority, with 1 being the highest priority in the case of a child task.

**Limits**

Valid range for a child task is 1 to 255 however, the priority cannot be higher (lower number) than the ES parent application priority.

Definition at line 652 of file example\_platform\_cfg.h.

**12.73.2.63 CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE 4096

**Purpose** Define Performance Analyzer Child Task Stack Size

**Description:**

This parameter defines the stack size of the child task spawned by the Executive Services to write performance data to a file.

**Limits**

It is recommended this parameter be greater than or equal to 4KB. This parameter is limited by the maximum value allowed by the data type. In this case, the data type is an unsigned 32-bit integer, so the valid range is 0 to 0xFFFFFFFF.

Definition at line 666 of file example\_platform\_cfg.h.

**12.73.2.64 CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE** #define CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE 10000

**Purpose** Define Max Size of Performance Data Buffer

**Description:**

Defines the maximum size of the performance data buffer. Units are number of performance data entries. An entry is defined by a 32 bit data word followed by a 64 bit time stamp.

**Limits**

There is a lower limit of 1025. There are no restrictions on the upper limit however, the maximum buffer size is system dependent and should be verified. The units are number of entries. An entry is defined by a 32 bit data word followed by a 64 bit time stamp.

Definition at line 573 of file example\_platform\_cfg.h.

**12.73.2.65 CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS** #define CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS 50

**Purpose** Define Performance Analyzer Child Task Number of Entries Between Delay

**Description:**

This parameter defines the number of performance analyzer entries the Performance Analyzer Child Task will write to the file between delays.

Definition at line 691 of file example\_platform\_cfg.h.

**12.73.2.66 CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL** #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL ~CFE\_PLATFORM\_ES\_PE

**Purpose** Define Filter Mask Setting for Enabling All Performance Entries

**Description:**

Defines the filter mask for enabling all performance entries. The value is a bit mask. For each bit, 0 means the corresponding entry is disabled and 1 means it is enabled.

Definition at line 593 of file example\_platform\_cfg.h.

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

```
#define CFE_PLATFORM_ES_PERF_FILTMASK_INIT CFE_PLATFORM_ES_PERF_FILTMASK_INIT
```

**Purpose** Define Default Filter Mask Setting for Performance Data Buffer

**Description:**

Defines the default filter mask for the performance data buffer. The value is a bit mask. For each bit, 0 means the corresponding entry is disabled and 1 means it is enabled.

Definition at line 604 of file example\_platform\_cfg.h.

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

```
#define CFE_PLATFORM_ES_PERF_FILTMASK_NONE 0
```

**Purpose** Define Filter Mask Setting for Disabling All Performance Entries

**Description:**

Defines the filter mask for disabling all performance entries. The value is a bit mask. For each bit, 0 means the corresponding entry is disabled and 1 means it is enabled.

Definition at line 583 of file example\_platform\_cfg.h.

**12.73.2.69 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 626 of file example\_platform\_cfg.h.

**12.73.2.70 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 637 of file example\_platform\_cfg.h.

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

```
#define CFE_PLATFORM_ES_PERF_TRIGMASK_NONE 0
```

**Purpose** Define Default Filter Trigger Setting for Disabling All Performance Entries

**Description:**

Defines the default trigger mask for disabling all performance data entries. The value is a bit mask. For each bit, 0 means the trigger for the corresponding entry is disabled and 1 means it is enabled.

Definition at line 615 of file example\_platform\_cfg.h.

**12.73.2.72 CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS** #define CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS 17

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

**Description:**

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS.

**Limits:**

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

The ES and CDS block size lists must correlate with this value

Definition at line 750 of file example\_platform\_cfg.h.

**12.73.2.73 CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING "/ram"

**Purpose** Default virtual path for volatile storage

**Description:**

The **CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING** parameter is used to set the cFE mount path for the CFE RAM disk. This is a parameter for missions that do not want to use the default value of "/ram", or for missions that need to have a different value for different CPUs or Spacecraft. Note that the vxWorks OSAL cannot currently handle names that have more than one path separator in it. The names "/ram", "/ramdisk", "/disk123" will all work, but "/disks/ram" will not. Multiple separators can be used with the posix or RTEMS ports.

Definition at line 143 of file example\_platform\_cfg.h.

**12.73.2.74 CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS 4096

**Purpose** ES Ram Disk Number of Sectors

**Description:**

Defines the ram disk number of sectors. The ram disk is one of four memory areas that are preserved on a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as USER\_RESERVED\_MEM in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

**Limits**

There is a lower limit of 128. There are no restrictions on the upper limit however, the maximum number of RAM sectors is system dependent and should be verified.

Definition at line 324 of file example\_platform\_cfg.h.

```
12.73.2.75 CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED #define CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED 30
```

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

Description:

The `CFE_PLATFORM_ES_RAM_DISK_PERCENT_RESERVED` parameter is used to make sure that the Volatile ( RAM ) Disk has a defined amount of free space during a processor reset. The cFE uses the Volatile disk to decompress cFE applications during system startup. If this Volatile disk happens to get filled with logs and misc files, then a processor reset may not work, because there will be no room to decompress cFE apps. To solve that problem, this parameter sets the "Low Water Mark" for disk space on a Processor reset. It should be set to allow the largest cFE Application to be decompressed. During a Processor reset, if there is not sufficient space left on the disk, it will be re-formatted in order to clear up some space.

This feature can be turned OFF by setting the parameter to 0.

Limits

There is a lower limit of 0 and an upper limit of 75 on this configuration parameter. Units are percentage. A setting of zero will turn this feature off.

Definition at line 348 of file example\_platform\_cfg.h.

```
12.73.2.76 CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE #define CFE_PLATFORM_ES_RAM_DISK_SECTOR_SIZE 512
```

**Purpose** ES Ram Disk Sector Size

Description:

Defines the ram disk sector size. The ram disk is 1 of 4 memory areas that are preserved on a processor reset. NOTE: Changing this value changes memory allocation, and may require changes to platform specific values (in CFE\_PSP) such as `USER_RESERVED_MEM` in VxWorks depending on the memory areas being used for preserved data and on OS specific behavior.

Limits

There is a lower limit of 128. There are no restrictions on the upper limit however, the maximum RAM disk sector size is system dependent and should be verified.

Definition at line 306 of file example\_platform\_cfg.h.

```
12.73.2.77 CFE_PLATFORM_ES_START_TASK_PRIORITY #define CFE_PLATFORM_ES_START_TASK_PRIORITY 68
```

**Purpose** Define ES Task Priority

Description:

Defines the cFE\_ES Task priority.

Limits

Not Applicable

Definition at line 100 of file example\_platform\_cfg.h.

**12.73.2.78 CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_START\_TASK\_←  
STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define ES Task Stack Size

**Description:**

Defines the cFE\_ES Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 115 of file example\_platform\_cfg.h.

**12.73.2.79 CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_←  
\_SCRIPT\_TIMEOUT\_MSEC 1000

**Purpose** Startup script timeout

**Description:**

The upper limit for the total amount of time that all apps listed in the CFE ES startup script may take to all become ready.

Unlike the "core" app timeout, this is a soft limit; if the allotted time is exceeded, it probably indicates an issue with one of the apps, but does not cause CFE ES to take any additional action other than logging the event to the syslog.

Units are in milliseconds

**Limits:**

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 870 of file example\_platform\_cfg.h.

**12.73.2.80 CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_←  
POLL\_MSEC 50

**Purpose** Poll timer for startup sync delay

**Description:**

During startup, some tasks may need to synchronize their own initialization with the initialization of other applications in the system.

CFE ES implements an API to accomplish this, that performs a task delay (sleep) while polling the overall system state until other tasks are ready.

This value controls the amount of time that the CFE\_ES\_ApplicationSyncDelay will sleep between each check of the system state. This should be large enough to allow other tasks to run, but not so large as to noticeably delay the startup completion.

Units are in milliseconds

**Limits:**

Must be defined as an integer value that is greater than or equal to zero.

Definition at line 852 of file example\_platform\_cfg.h.

**12.73.2.81 CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE** #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072

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

**Description:**

Defines the size in bytes of the cFE system log. The system log holds variable length strings that are terminated by a linefeed and null character.

**Limits**

There is a lower limit of 512. There are no restrictions on the upper limit however, the maximum system log size is system dependent and should be verified.

Definition at line 215 of file example\_platform\_cfg.h.

**12.73.2.82 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 385 of file example\_platform\_cfg.h.

**12.73.2.83 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 432 of file example\_platform\_cfg.h.

**12.73.2.84 CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC** #define CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC 15

**Purpose** Sustained number of event messages per second per app before squelching

**Description:**

Sustained number of events that may be emitted per app per second.

**Limits**

This number must be less than or equal to [CFE\\_PLATFORM\\_EVS\\_MAX\\_APP\\_EVENT\\_BURST](#). Values lower than 8 may cause functional and unit test failures.

Definition at line 938 of file example\_platform\_cfg.h.

**12.73.2.85 CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE** #define CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE "/ram/cfe\_evs\_app.dat"

**Purpose** Default EVS Application Data Filename

**Description:**

The value of this constant defines the filename used to store the EVS Application Data(event counts/filtering information). This filename is used only when no filename is specified in the command to dump the event log.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 979 of file example\_platform\_cfg.h.

**12.73.2.86 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE** #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE "/ram/cfe\_evs.log"

**Purpose** Default Event Log Filename

**Description:**

The value of this constant defines the filename used to store the Event Services local event log. This filename is used only when no filename is specified in the command to dump the event log.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 952 of file example\_platform\_cfg.h.

**12.73.2.87 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE** #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE 1

**Purpose** Default EVS Local Event Log Mode

**Description:**

Defines a state of overwrite(0) or discard(1) for the operation of the EVS local event log. The log may operate in either Overwrite mode = 0, where once the log becomes full the oldest event in the log will be overwritten, or Discard mode = 1, where once the log becomes full the contents of the log are preserved and the new event is discarded. Overwrite Mode = 0, Discard Mode = 1.

**Limits**

The valid settings are 0 or 1

Definition at line 1026 of file example\_platform\_cfg.h.

**12.73.2.88 CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE** #define CFE\_PLATFORM\_EVS\_DEFAULT\_←  
MSG\_FORMAT\_MODE CFE\_EVS\_MsgFormat\_LONG

**Purpose** Default EVS Message Format Mode

**Description:**

Defines the default message format (long or short) for event messages being sent to the ground. Choose between [CFE\\_EVS\\_MsgFormat\\_LONG](#) or [CFE\\_EVS\\_MsgFormat\\_SHORT](#).

**Limits**

The valid settings are [CFE\\_EVS\\_MsgFormat\\_LONG](#) or [CFE\\_EVS\\_MsgFormat\\_SHORT](#)

Definition at line 1039 of file example\_platform\_cfg.h.

**12.73.2.89 CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG** #define CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_←  
FLAG 0xE

**Purpose** Default EVS Event Type Filter Mask

**Description:**

Defines a state of on or off for all four event types. The term event 'type' refers to the criticality level and may be Debug, Informational, Error or Critical. Each event type has a bit position. (bit 0 = Debug, bit 1 = Info, bit 2 = Error, bit 3 = Critical). This is a global setting, meaning it applies to all applications. To filter an event type, set its bit to zero. For example, 0xE means Debug = OFF, Info = ON, Error = ON, Critical = ON

**Limits**

The valid settings are 0x0 to 0xF.

Definition at line 1010 of file example\_platform\_cfg.h.

**12.73.2.90 CFE\_PLATFORM\_EVS\_LOG\_MAX** #define CFE\_PLATFORM\_EVS\_LOG\_MAX 20

**Purpose** Maximum Number of Events in EVS Local Event Log

**Description:**

Dictates the EVS local event log capacity. Units are the number of events.

**Limits**

There are no restrictions on the lower and upper limits however, the maximum log size is system dependent and should be verified.

Definition at line 964 of file example\_platform\_cfg.h.

**12.73.2.91 CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST** #define CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST 32

**Purpose** Maximum number of event before squelching

**Description:**

Maximum number of events that may be emitted per app per second. Setting this to 0 will cause events to be unrestricted.

**Limits**

This number must be less than or equal to INT\_MAX/1000

Definition at line 926 of file example\_platform\_cfg.h.

**12.73.2.92 CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS** #define CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS 8

**Purpose** Define Maximum Number of Event Filters per Application

**Description:**

Maximum number of events that may be filtered per application.

**Limits**

There are no restrictions on the lower and upper limits however, the maximum number of event filters is system dependent and should be verified.

Definition at line 914 of file example\_platform\_cfg.h.

**12.73.2.93 CFE\_PLATFORM\_EVS\_PORT\_DEFAULT** #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT 0x0001

**Purpose** Default EVS Output Port State

**Description:**

Defines the default port state (enabled or disabled) for the four output ports defined within the Event Service. Port 1 is usually the uart output terminal. To enable a port, set the proper bit to a 1. Bit 0 is port 1, bit 1 is port2 etc.

**Limits**

The valid settings are 0x0 to 0xF.

Definition at line 993 of file example\_platform\_cfg.h.

```
12.73.2.94 CFE_PLATFORM_EVS_START_TASK_PRIORITY #define CFE_PLATFORM_EVS_START_TASK_←  
PRIORITY 61
```

**Purpose** Define EVS Task Priority

**Description:**

Defines the cFE\_EVS Task priority.

**Limits**

Not Applicable

Definition at line 886 of file example\_platform\_cfg.h.

```
12.73.2.95 CFE_PLATFORM_EVS_START_TASK_STACK_SIZE #define CFE_PLATFORM_EVS_START_TASK_←  
STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE
```

**Purpose** Define EVS Task Stack Size

**Description:**

Defines the cFE\_EVS Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 901 of file example\_platform\_cfg.h.

```
12.73.2.96 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 1134 of file example\_platform\_cfg.h.

**12.73.2.97 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 1204 of file example\_platform\_cfg.h.

**12.73.2.98 CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT** #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4

**Purpose** Default Subscription Message Limit

**Description:**

Dictates the default Message Limit when using the [CFE\\_SB\\_Subscribe](#) API. This will limit the number of messages with a specific message ID that can be received through a subscription. This only changes the default; other message limits can be set on a per subscription basis using [CFE\\_SB\\_SubscribeEx](#).

**Limits**

This parameter has a lower limit of 4 and an upper limit of 65535.

Definition at line 1112 of file example\_platform\_cfg.h.

**12.73.2.99 CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_←  
FILENAME "/ram/cfe\_sb\_pipe.dat"

**Purpose** Default Pipe Information Filename

**Description:**

The value of this constant defines the filename used to store the software bus pipe information. This filename is used only when no filename is specified in the command.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 1187 of file example\_platform\_cfg.h.

**12.73.2.100 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 1173 of file example\_platform\_cfg.h.

**12.73.2.101 CFE\_PLATFORM\_SB\_FILTER\_MASK1** #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 [CFE\\_EVS\\_FIRST\\_4\\_STOP](#)  
Definition at line 1222 of file example\_platform\_cfg.h.

**12.73.2.102 CFE\_PLATFORM\_SB\_FILTER\_MASK2** #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 [CFE\\_EVS\\_FIRST\\_4\\_STOP](#)  
Definition at line 1225 of file example\_platform\_cfg.h.

**12.73.2.103 CFE\_PLATFORM\_SB\_FILTER\_MASK3** #define CFE\_PLATFORM\_SB\_FILTER\_MASK3 [CFE\\_EVS\\_FIRST\\_16\\_STOP](#)  
Definition at line 1228 of file example\_platform\_cfg.h.

**12.73.2.104 CFE\_PLATFORM\_SB\_FILTER\_MASK4** #define CFE\_PLATFORM\_SB\_FILTER\_MASK4 [CFE\\_EVS\\_FIRST\\_16\\_STOP](#)  
Definition at line 1231 of file example\_platform\_cfg.h.

**12.73.2.105 CFE\_PLATFORM\_SB\_FILTER\_MASK5** #define CFE\_PLATFORM\_SB\_FILTER\_MASK5 [CFE\\_EVS\\_NO\\_FILTER](#)  
Definition at line 1234 of file example\_platform\_cfg.h.

**12.73.2.106 CFE\_PLATFORM\_SB\_FILTER\_MASK6** #define CFE\_PLATFORM\_SB\_FILTER\_MASK6 [CFE\\_EVS\\_NO\\_FILTER](#)  
Definition at line 1237 of file example\_platform\_cfg.h.

**12.73.2.107 CFE\_PLATFORM\_SB\_FILTER\_MASK7** #define CFE\_PLATFORM\_SB\_FILTER\_MASK7 [CFE\\_EVS\\_NO\\_FILTER](#)  
Definition at line 1240 of file example\_platform\_cfg.h.

**12.73.2.108 CFE\_PLATFORM\_SB\_FILTER\_MASK8** #define CFE\_PLATFORM\_SB\_FILTER\_MASK8 [CFE\\_EVS\\_NO\\_FILTER](#)  
Definition at line 1243 of file example\_platform\_cfg.h.

**12.73.2.109 CFE\_PLATFORM\_SB\_FILTERED\_EVENT1** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_SB\_SEND\_NO\_SUBS\_EID

**Purpose** SB Event Filtering

Description:

This group of configuration parameters dictates what SB events will be filtered through SB. The filtering will begin after the SB task initializes and stay in effect until a cmd to SB changes it. This allows the operator to set limits on the number of event messages that are sent during system initialization. NOTE: Set all unused event values and mask values to zero

Limits

This filtering applies only to SB events. These parameters have a lower limit of 0 and an upper limit of 65535.

Definition at line 1221 of file example\_platform\_cfg.h.

**12.73.2.110 CFE\_PLATFORM\_SB\_FILTERED\_EVENT2** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_SB\_DUP\_SUBSCRIP\_EID

Definition at line 1224 of file example\_platform\_cfg.h.

**12.73.2.111 CFE\_PLATFORM\_SB\_FILTERED\_EVENT3** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT3 CFE\_SB\_MSGID\_LIM\_ERR\_EID

Definition at line 1227 of file example\_platform\_cfg.h.

**12.73.2.112 CFE\_PLATFORM\_SB\_FILTERED\_EVENT4** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT4 CFE\_SB\_Q\_FULL\_ERR\_EID

Definition at line 1230 of file example\_platform\_cfg.h.

**12.73.2.113 CFE\_PLATFORM\_SB\_FILTERED\_EVENT5** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT5 0

Definition at line 1233 of file example\_platform\_cfg.h.

**12.73.2.114 CFE\_PLATFORM\_SB\_FILTERED\_EVENT6** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT6 0

Definition at line 1236 of file example\_platform\_cfg.h.

**12.73.2.115 CFE\_PLATFORM\_SB\_FILTERED\_EVENT7** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT7 0

Definition at line 1239 of file example\_platform\_cfg.h.

**12.73.2.116 CFE\_PLATFORM\_SB\_FILTERED\_EVENT8** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT8 0

Definition at line 1242 of file example\_platform\_cfg.h.

**12.73.2.117 CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID** #define CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_←  
MSGID 0xFFFF

**Purpose** Highest Valid Message Id

**Description:**

The value of this constant dictates the range of valid message ID's, from 0 to CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID (inclusive).

Although this can be defined differently across platforms, each platform can only publish/subscribe to message ids within their allowable range. Typically this value is set the same across all mission platforms to avoid this complexity.

**Limits**

This parameter has a lower limit is 1, and an upper limit of 0xFFFFFFFF.

When using the direct message map implementation for software bus routing, this value is used to size the map where a value of 0xFFFF results in a 16 KBytes map and 0xFFFF is 128 KBytes.

When using the hash implementation for software bus routing, a multiple of the CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS is used to size the message map. In that case the range selected here does not impact message map memory use, so it's reasonable to use up to the full range supported by the message ID implementation.

Definition at line 1159 of file example\_platform\_cfg.h.

**12.73.2.118 CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE (CFE\_MISSION\_SB\_MAX\_SB\_M  
+ 128)

Definition at line 1272 of file example\_platform\_cfg.h.

**12.73.2.119 CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT** #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16

**Purpose** Maximum Number of unique local destinations a single MsgId can have

**Description:**

Dictates the maximum number of unique local destinations a single MsgId can have.

**Limits**

This parameter has a lower limit of 1. There are no restrictions on the upper limit however, the maximum number of destinations per packet is system dependent and should be verified. Destination number values that are checked against this configuration are defined by a 16 bit data word.

Definition at line 1097 of file example\_platform\_cfg.h.

**12.73.2.120 CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS** #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256

**Purpose** Maximum Number of Unique Message IDs SB Routing Table can hold

**Description:**

Dictates the maximum number of unique MsgIds the SB routing table will hold. This constant has a direct effect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

**Limits**

This must be a power of two if software bus message routing hash implementation is being used. Lower than 64 will cause unit test failures, and telemetry reporting is impacted below 32. There is no hard upper limit, but impacts memory footprint. For software bus message routing search implementation the number of msg ids subscribed to impacts performance.

Definition at line 1064 of file example\_platform\_cfg.h.

**12.73.2.121 CFE\_PLATFORM\_SB\_MAX\_PIPES** #define CFE\_PLATFORM\_SB\_MAX\_PIPES 64

**Purpose** Maximum Number of Unique Pipes SB Routing Table can hold

**Description:**

Dictates the maximum number of unique Pipes the SB routing table will hold. This constant has a direct effect on the size of SB's tables and arrays. Keeping this count as low as possible will save memory. To see the run-time, high-water mark and the current utilization figures regarding this parameter, send an SB command to 'Send Statistics Pkt'.

**Limits**

This parameter has a lower limit of 1. This parameter must also be less than or equal to OS\_MAX\_QUEUES.

Definition at line 1081 of file example\_platform\_cfg.h.

**12.73.2.122 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 8

**Purpose** Define SB Memory Pool Block Sizes

**Description:**

Software Bus Memory Pool Block Sizes

**Limits**

These sizes MUST be increasing and MUST be an integral multiple of 4. The number of block sizes defined cannot exceed [CFE\\_PLATFORM\\_ES\\_POOL\\_MAX\\_BUCKETS](#)

Definition at line 1256 of file example\_platform\_cfg.h.

**12.73.2.123 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 16

Definition at line 1257 of file example\_platform\_cfg.h.

**12.73.2.124 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 20

Definition at line 1258 of file example\_platform\_cfg.h.

**12.73.2.125 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 36

Definition at line 1259 of file example\_platform\_cfg.h.

**12.73.2.126 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 64

Definition at line 1260 of file example\_platform\_cfg.h.

**12.73.2.127 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 96

Definition at line 1261 of file example\_platform\_cfg.h.

**12.73.2.128 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 128

Definition at line 1262 of file example\_platform\_cfg.h.

**12.73.2.129 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 160

Definition at line 1263 of file example\_platform\_cfg.h.

**12.73.2.130 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 256

Definition at line 1264 of file example\_platform\_cfg.h.

**12.73.2.131 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 512

Definition at line 1265 of file example\_platform\_cfg.h.

**12.73.2.132 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 1024

Definition at line 1266 of file example\_platform\_cfg.h.

**12.73.2.133 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 2048

Definition at line 1267 of file example\_platform\_cfg.h.

**12.73.2.134 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 4096

Definition at line 1268 of file example\_platform\_cfg.h.

**12.73.2.135 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 8192

Definition at line 1269 of file example\_platform\_cfg.h.

**12.73.2.136 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 16384

Definition at line 1270 of file example\_platform\_cfg.h.

**12.73.2.137 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 32768

Definition at line 1271 of file example\_platform\_cfg.h.

**12.73.2.138 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 1283 of file example\_platform\_cfg.h.

**12.73.2.139 CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_SB\_START\_TASK\_←  
STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define SB Task Stack Size

**Description:**

Defines the cFE\_SB Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1298 of file example\_platform\_cfg.h.

**12.73.2.140 CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES** #define CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_←  
BYTES 524288

**Purpose** Size of Table Services Table Memory Pool

**Description:**

Defines the TOTAL size of the memory pool that cFE Table Services allocates from the system. The size must be large enough to provide memory for each registered table, the inactive buffers for double buffered tables and for the shared inactive buffers for single buffered tables.

**Limits**

The cFE does not place a limit on the size of this parameter.

Definition at line 1345 of file example\_platform\_cfg.h.

**12.73.2.141 CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE** #define CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE "/ram/cfe\_tbl\_reg.log"

**Purpose** Default Filename for a Table Registry Dump

**Description:**

Defines the file name used to store the table registry when no filename is specified in the dump registry command.

**Limits**

The length of each string, including the NULL terminator cannot exceed the [OS\\_MAX\\_PATH\\_LEN](#) value.

Definition at line 1459 of file example\_platform\_cfg.h.

**12.73.2.142 CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES 32

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

**Description:**

Defines the maximum number of critical tables supported by this processor's Table Services.

**Limits**

This number must be less than 32767. It should be recognized that this parameter determines the size of the Critical Table Registry which is maintained in the Critical Data Store. An excessively high number will waste Critical Data Store memory. Therefore, this number must not exceed the value defined in [CFE\\_ES\\_CDS\\_MAX\\_CRITICAL\\_TABLES](#).

Definition at line 1400 of file example\_platform\_cfg.h.

**12.73.2.143 CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE** #define CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE 16384

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

**Description:**

Defines the maximum allowed size (in bytes) of a double buffered table.

**Limits**

The cFE does not place a limit on the size of this parameter but it must be less than half of [CFE\\_PLATFORM\\_TBL\\_BUF\\_MEMORY\\_BLOCK\\_SIZE](#).

Definition at line 1357 of file example\_platform\_cfg.h.

**12.73.2.144 CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256

**Purpose** Maximum Number of Table Handles

**Description:**

Defines the maximum number of Table Handles.

**Limits**

This number must be less than 32767. This number must be at least as big as the number of tables (**CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES**) and should be set higher if tables are shared between applications.

Definition at line 1413 of file example\_platform\_cfg.h.

**12.73.2.145 CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128

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

**Description:**

Defines the maximum number of tables supported by this processor's Table Services.

**Limits**

This number must be less than 32767. It should be recognized that this parameter determines the size of the Table Registry. An excessively high number will waste memory.

Definition at line 1386 of file example\_platform\_cfg.h.

**12.73.2.146 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 1446 of file example\_platform\_cfg.h.

```
12.73.2.147 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 1428 of file example\_platform\_cfg.h.

```
12.73.2.148 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\\_BYTES](#).

Definition at line 1373 of file example\_platform\_cfg.h.

```
12.73.2.149 CFE_PLATFORM_TBL_START_TASK_PRIORITY #define CFE_PLATFORM_TBL_START_TASK_←  
PRIORITY 70
```

**Purpose** Define TBL Task Priority

Description:

Defines the cFE\_TBL Task priority.

Limits

Not Applicable

Definition at line 1314 of file example\_platform\_cfg.h.

**12.73.2.150 CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TBL\_START\_TASK\_←  
STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define TBL Task Stack Size

**Description:**

Defines the cFE\_TBL Task Stack Size

**Limits**

There is a lower limit of 2048 on this configuration parameter. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1329 of file example\_platform\_cfg.h.

**12.73.2.151 CFE\_PLATFORM\_TBL\_U32FROM4CHARS** #define CFE\_PLATFORM\_TBL\_U32FROM4CHARS (←  
\_C1,  
\_C2,  
\_C3,  
\_C4) ((uint32) (\_C1) << 24 | (uint32) (\_C2) << 16 | (uint32) (\_C3) << 8 | (uint32) (←  
\_C4))

Definition at line 1481 of file example\_platform\_cfg.h.

**12.73.2.152 CFE\_PLATFORM\_TBL\_VALID\_PRID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 (1)

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

**Description:**

Defines the processor ID values used for validating the processor ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

**Limits**

This value can be any 32 bit unsigned integer.

Definition at line 1530 of file example\_platform\_cfg.h.

**12.73.2.153 CFE\_PLATFORM\_TBL\_VALID\_PRID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CH←  
'b', 'c', 'd'))

Definition at line 1531 of file example\_platform\_cfg.h.

**12.73.2.154 CFE\_PLATFORM\_TBL\_VALID\_PRID\_3** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 0

Definition at line 1532 of file example\_platform\_cfg.h.

**12.73.2.155 CFE\_PLATFORM\_TBL\_VALID\_PRID\_4** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 0

Definition at line 1533 of file example\_platform\_cfg.h.

**12.73.2.156 CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT** `#define CFE_PLATFORM_TBL_VALID_PRID_COUNT 0`

**Purpose** Number of Processor ID's specified for validation

**Description:**

Defines the number of specified processor ID values that are verified during table loads. If the number is zero then no validation of the processor ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of processor ID's defined below are compared to the processor ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified processor ID values.

**Limits**

This number must be greater than or equal to zero and less than or equal to 4.

Definition at line 1516 of file example\_platform\_cfg.h.

**12.73.2.157 CFE\_PLATFORM\_TBL\_VALID\_SCID\_1** `#define CFE_PLATFORM_TBL_VALID_SCID_1 (0x42)`

**Purpose** Spacecraft ID values used for table load validation

**Description:**

Defines the spacecraft ID values used for validating the spacecraft ID field in the table file header. To be valid, the spacecraft ID specified in the table file header must match one of the values defined here.

**Limits**

This value can be any 32 bit unsigned integer.

Definition at line 1496 of file example\_platform\_cfg.h.

**12.73.2.158 CFE\_PLATFORM\_TBL\_VALID\_SCID\_2** `#define CFE_PLATFORM_TBL_VALID_SCID_2 (CFE_PLATFORM_TBL_U32FROM4CH  
'b', 'c', 'd'))`

Definition at line 1497 of file example\_platform\_cfg.h.

**12.73.2.159 CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT** `#define CFE_PLATFORM_TBL_VALID_SCID_COUNT 0`

**Purpose** Number of Spacecraft ID's specified for validation

**Description:**

Defines the number of specified spacecraft ID values that are verified during table loads. If the number is zero then no validation of the spacecraft ID field in the table file header is performed when tables are loaded. Non-zero values indicate how many values from the list of spacecraft ID's defined below are compared to the spacecraft ID field in the table file header. The ELF2CFETBL tool may be used to create table files with specified spacecraft ID values.

**Limits**

This number must be greater than or equal to zero and less than or equal to 2.

Definition at line 1478 of file example\_platform\_cfg.h.

**12.73.2.160 CFE\_PLATFORM\_TIME\_CFG\_CLIENT** #define CFE\_PLATFORM\_TIME\_CFG\_CLIENT false  
Definition at line 1553 of file example\_platform\_cfg.h.

**12.73.2.161 CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY** #define CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY 8

**Purpose** Define Periodic Time to Update Local Clock Tone Latch

**Description:**

Define Periodic Time to Update Local Clock Tone Latch. Applies only when in flywheel mode. This define dictates the period at which the simulated 'last tone' time is updated. Units are seconds.

**Limits**

Not Applicable

Definition at line 1710 of file example\_platform\_cfg.h.

**12.73.2.162 CFE\_PLATFORM\_TIME\_CFG\_SERVER** #define CFE\_PLATFORM\_TIME\_CFG\_SERVER true

**Purpose** Time Server or Time Client Selection

**Description:**

This configuration parameter selects whether the Time task functions as a time "server" or "client". A time server generates the "time at the tone" packet which is received by time clients.

**Limits**

Enable one, and only one by defining either CFE\_PLATFORM\_TIME\_CFG\_SERVER or CFE\_PLATFORM\_TIME\_CFG\_CLIENT AS true. The other must be defined as false.

Definition at line 1552 of file example\_platform\_cfg.h.

**12.73.2.163 CFE\_PLATFORM\_TIME\_CFG\_SIGNAL** #define CFE\_PLATFORM\_TIME\_CFG\_SIGNAL false

**Purpose** Include or Exclude the Primary/Redundant Tone Selection Cmd

**Description:**

Depending on the specific hardware system configuration, it may be possible to switch between a primary and redundant tone signal. If supported by hardware, this definition will enable command interfaces to select the active tone signal. Both Time Clients and Time Servers support this feature. Note: Set the CFE\_PLATFORM\_TIME\_CFG\_SIGNAL define to true to enable tone signal commands.

**Limits**

Not Applicable

Definition at line 1600 of file example\_platform\_cfg.h.

**12.73.2.164 CFE\_PLATFORM\_TIME\_CFG\_SOURCE** #define CFE\_PLATFORM\_TIME\_CFG\_SOURCE false

**Purpose** Include or Exclude the Internal/External Time Source Selection Cmd

Description:

By default, Time Servers maintain time using an internal MET which may be a h/w register or software counter, depending on available hardware. The following definition enables command interfaces to switch between an internal MET, or external time data received from one of several supported external time sources. Only a Time Server may be configured to use external time data. Note: Set the CFE\_PLATFORM\_TIME\_CFG\_SOURCE define to true to include the Time Source Selection Command (command allows selection between the internal or external time source). Then choose the external source with the CFE\_TIME\_CFG\_SRC\_??? define.

Limits

Only applies if CFE\_PLATFORM\_TIME\_CFG\_SERVER is set to true.

Definition at line 1620 of file example\_platform\_cfg.h.

**12.73.2.165 CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS** #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS false

Definition at line 1637 of file example\_platform\_cfg.h.

**12.73.2.166 CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET** #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET false

**Purpose** Choose the External Time Source for Server only

Description:

If CFE\_PLATFORM\_TIME\_CFG\_SOURCE is set to true, then one of the following external time source types must also be set to true. Do not set any of the external time source types to true unless CFE\_PLATFORM\_TIME\_CFG\_SOURCE is set to true.

Limits

1. If CFE\_PLATFORM\_TIME\_CFG\_SOURCE is set to true then one and only one of the following three external time sources can and must be set true: CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET, CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS, CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME
2. Only applies if CFE\_PLATFORM\_TIME\_CFG\_SERVER is set to true.

Definition at line 1636 of file example\_platform\_cfg.h.

**12.73.2.167 CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME** #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME false

Definition at line 1638 of file example\_platform\_cfg.h.

**12.73.2.168 CFE\_PLATFORM\_TIME\_CFG\_START\_FLY** #define CFE\_PLATFORM\_TIME\_CFG\_START\_FLY 2

**Purpose** Define Time to Start Flywheel Since Last Tone

**Description:**

Define time to enter flywheel mode (in seconds since last tone data update) Units are microseconds as measured with the local clock.

**Limits**

Not Applicable

Definition at line 1697 of file example\_platform\_cfg.h.

**12.73.2.169 CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT** `#define CFE_PLATFORM_TIME_CFG_TONE_LIMIT 20000`**Purpose** Define Timing Limits From One Tone To The Next**Description:**

Defines limits to the timing of the 1Hz tone signal. A tone signal is valid only if it arrives within one second (plus or minus the tone limit) from the previous tone signal.Units are microseconds as measured with the local clock.

**Limits**

Not Applicable

Definition at line 1685 of file example\_platform\_cfg.h.

**12.73.2.170 CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL** `#define CFE_PLATFORM_TIME_CFG_VIRTUAL true`**Purpose** Time Tone In Big-Endian Order**Description:**

If this configuration parameter is defined, the CFE time server will publish time tones with payloads in big-endian order, and time clients will expect the tones to be in big-endian order. This is useful for mixed-endian environments. This will become obsolete once EDS is available and the CFE time tone message is defined.

**Purpose** Local MET or Virtual MET Selection for Time Servers**Description:**

Depending on the specific hardware system configuration, it may be possible for Time Servers to read the "local" MET from a h/w register rather than having to track the MET as the count of tone signal interrupts (virtual MET)

Time Clients must be defined as using a virtual MET. Also, a Time Server cannot be defined as having both a h/w MET and an external time source (they both cannot synchronize to the same tone).

Note: "disable" this define (set to false) only for Time Servers with local hardware that supports a h/w MET that is synchronized to the tone signal !!!

**Limits**

Only applies if [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) is set to true.

Definition at line 1585 of file example\_platform\_cfg.h.

**12.73.2.171 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS** #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS 0

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

**Description:**

If [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) is set to true and one of the external time sources is also set to true, then the delta time limits for range checking is used.

When a new time value is received from an external source, the value is compared against the "expected" time value. If the delta exceeds the following defined amount, then the new time data will be ignored. This range checking is only performed after the clock state has been commanded to "valid". Until then, external time data is accepted unconditionally.

**Limits**

Applies only if both [CFE\\_PLATFORM\\_TIME\\_CFG\\_SERVER](#) and [CFE\\_PLATFORM\\_TIME\\_CFG\\_SOURCE](#) are set to true.

Definition at line 1657 of file example\_platform\_cfg.h.

**12.73.2.172 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS** #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS 500000

Definition at line 1658 of file example\_platform\_cfg.h.

**12.73.2.173 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS** #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS 27

**Purpose** Define the Local Clock Rollover Value in seconds and subseconds

**Description:**

Specifies the capability of the local clock. Indicates the time at which the local clock rolls over.

**Limits**

Not Applicable

Definition at line 1670 of file example\_platform\_cfg.h.

**12.73.2.174 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS** #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS 0

Definition at line 1671 of file example\_platform\_cfg.h.

**12.73.2.175 CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY 25

PRIORITY 25

Definition at line 1727 of file example\_platform\_cfg.h.

**12.73.2.176 CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE 8192

Definition at line 1746 of file example\_platform\_cfg.h.

**12.73.2.177 CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY 60

**Purpose** Define TIME Task Priorities

**Description:**

Defines the cFE\_TIME Task priority. Defines the cFE\_TIME Tone Task priority. Defines the cFE\_TIME 1HZ Task priority.

**Limits**

There is a lower limit of zero and an upper limit of 255 on these configuration parameters. Remember that the meaning of each task priority is inverted – a "lower" number has a "higher" priority.

Definition at line 1725 of file example\_platform\_cfg.h.

**12.73.2.178 CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

**Purpose** Define TIME Task Stack Sizes

**Description:**

Defines the cFE\_TIME Main Task Stack Size Defines the cFE\_TIME Tone Task Stack Size Defines the cFE\_TIME 1HZ Task Stack Size

**Limits**

There is a lower limit of 2048 on these configuration parameters. There are no restrictions on the upper limit however, the maximum stack size is system dependent and should be verified. Most operating systems provide tools for measuring the amount of stack used by a task during operation. It is always a good idea to verify that no more than 1/2 of the stack is used.

Definition at line 1744 of file example\_platform\_cfg.h.

**12.73.2.179 CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY 25

Definition at line 1726 of file example\_platform\_cfg.h.

**12.73.2.180 CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096

Definition at line 1745 of file example\_platform\_cfg.h.

- 12.74 [cfe/docs/src/cfe\\_api.dox File Reference](#)
- 12.75 [cfe/docs/src/cfe\\_es.dox File Reference](#)
- 12.76 [cfe/docs/src/cfe\\_evs.dox File Reference](#)
- 12.77 [cfe/docs/src/cfe\\_frontpage.dox File Reference](#)
- 12.78 [cfe/docs/src/cfe\\_glossary.dox File Reference](#)
- 12.79 [cfe/docs/src/cfe\\_sb.dox File Reference](#)
- 12.80 [cfe/docs/src/cfe\\_tbl.dox File Reference](#)
- 12.81 [cfe/docs/src/cfe\\_time.dox File Reference](#)
- 12.82 [cfe/docs/src/cfe\\_xref.dox File Reference](#)
- 12.83 [cfe/docs/src/cfs\\_versions.dox File Reference](#)
- 12.84 [cfe/modules/config/fsw/inc/cfe\\_config\\_external.h File Reference](#)

```
#include "cfe_config_api_typedefs.h"
```

## Functions

- void [CFE\\_Config\\_SetupPlatformConfigInfo](#) (void)

### 12.84.1 Detailed Description

Initialization for CFE configuration registry, external items

These are generated from the build system, they are not directly set

### 12.84.2 Function Documentation

**12.84.2.1 CFE\_Config\_SetupPlatformConfigInfo()** void CFE\_Config\_SetupPlatformConfigInfo ( void )

## 12.85 cfe/modules/config/fsw/inc/cfe\_config\_init.h File Reference

```
#include "cfe_config_api_typedefs.h"
#include "cfe_config_ids.h"
```

## Functions

- void [CFE\\_Config\\_SetupBasicBuildInfo](#) (void)
- int32 [CFE\\_Config\\_Init](#) (void)

### 12.85.1 Detailed Description

Function declarations for items implemented in init.c

## 12.85.2 Function Documentation

**12.85.2.1 CFE\_Config\_Init()** `int32 CFE_Config_Init ( void )`

**12.85.2.2 CFE\_Config\_SetupBasicBuildInfo()** `void CFE_Config_SetupBasicBuildInfo ( void )`

## 12.86 cfe/modules/config/fsw/inc/cfe\_config\_lookup.h File Reference

```
#include "cfe_config_api_typedefs.h"
#include "cfe_config_table.h"
```

### Functions

- `CFE_Config_ValueEntry_t * CFE_Config_LocateConfigRecordByID (CFE_ConfigId_t ConfigId)`  
*Gets the value record associated with a config ID.*

### 12.86.1 Detailed Description

Function declarations for items implemented in lookup.c

### 12.86.2 Function Documentation

**12.86.2.1 CFE\_Config\_LocateConfigRecordByID()** `CFE_Config_ValueEntry_t* CFE_Config_LocateConfigRecordByID ( CFE_ConfigId_t ConfigId )`  
Gets the value record associated with a config ID.

## 12.87 cfe/modules/config/fsw/inc/cfe\_config\_nametable.h File Reference

```
#include "cfe_configid_offset.h"
```

### Data Structures

- struct `CFE_Config_IdNameEntry`

### Typedefs

- typedef struct `CFE_Config_IdNameEntry` `CFE_Config_IdNameEntry_t`

### Variables

- const `CFE_Config_IdNameEntry_t CFE_CONFIGID_NAMETABLE []`

### 12.87.1 Detailed Description

This file contains the CFE configuration registry global data definitions.

## 12.87.2 Typedef Documentation

### 12.87.2.1 CFE\_Config\_IdNameEntry\_t

```
typedef struct CFE_Config_IdNameEntry CFE_Config_IdNameEntry_t
```

## 12.87.3 Variable Documentation

### 12.87.3.1 CFE\_CONFIGID\_NAMETABLE

```
const CFE_Config_IdNameEntry_t CFE_CONFIGID_NAMETABLE[ ]  
[extern]
```

## 12.88 cfe/modules/config/fsw/inc/cfe\_config\_set.h File Reference

```
#include "cfe_config_api_typedefs.h"
```

### Functions

- void [CFE\\_Config\\_SetValue](#) (CFE\_ConfigId\_t ConfigId, uint32 Value)
- void [CFE\\_Config\\_SetObjPointer](#) (CFE\_ConfigId\_t ConfigId, const void \*Ptr)
- void [CFE\\_Config\\_SetString](#) (CFE\_ConfigId\_t ConfigId, const char \*Ptr)
- void [CFE\\_Config\\_SetArrayValue](#) (CFE\_ConfigId\_t ConfigId, const CFE\_Config\_ArrayValue\_t \*ArrayPtr)

### 12.88.1 Detailed Description

Function declarations for items implemented in set.c

### 12.88.2 Function Documentation

#### 12.88.2.1 CFE\_Config\_SetArrayValue()

```
void CFE_Config_SetArrayValue(  
    CFE_ConfigId_t ConfigId,  
    const CFE_Config_ArrayValue_t * ArrayPtr )
```

#### 12.88.2.2 CFE\_Config\_SetObjPointer()

```
void CFE_Config_SetObjPointer(   
    CFE_ConfigId_t ConfigId,  
    const void * Ptr )
```

#### 12.88.2.3 CFE\_Config\_SetString()

```
void CFE_Config_SetString(   
    CFE_ConfigId_t ConfigId,  
    const char * Ptr )
```

#### 12.88.2.4 CFE\_Config\_SetValue()

```
void CFE_Config_SetValue(   
    CFE_ConfigId_t ConfigId,  
    uint32 Value )
```

## 12.89 cfe/modules/config/fsw/inc/cfe\_config\_table.h File Reference

```
#include "common_types.h"
#include "cfe_config_ids.h"
```

### Data Structures

- union [CFE\\_Config\\_ValueBuffer](#)
- struct [CFE\\_Config\\_ValueEntry](#)

### Typedefs

- typedef enum [CFE\\_ConfigType](#) [CFE\\_ConfigType\\_t](#)
- typedef union [CFE\\_Config\\_ValueBuffer](#) [CFE\\_Config\\_ValueBuffer\\_t](#)
- typedef struct [CFE\\_Config\\_ValueEntry](#) [CFE\\_Config\\_ValueEntry\\_t](#)

### Enumerations

- enum [CFE\\_ConfigType](#) {  
    [CFE\\_ConfigType\\_UNDEFINED](#) , [CFE\\_ConfigType\\_VALUE](#) , [CFE\\_ConfigType\\_STRING](#) , [CFE\\_ConfigType\\_POINTER](#)  
    ,  
    [CFE\\_ConfigType\\_ARRAY](#) }

#### 12.89.1 Detailed Description

This file contains the CFE configuration registry global data definitions.

#### 12.89.2 Typedef Documentation

##### 12.89.2.1 [CFE\\_Config\\_ValueBuffer\\_t](#) [typedef union CFE\\_Config\\_ValueBuffer CFE\\_Config\\_ValueBuffer\\_t](#)

##### 12.89.2.2 [CFE\\_Config\\_ValueEntry\\_t](#) [typedef struct CFE\\_Config\\_ValueEntry CFE\\_Config\\_ValueEntry\\_t](#)

##### 12.89.2.3 [CFE\\_ConfigType\\_t](#) [typedef enum CFE\\_ConfigType CFE\\_ConfigType\\_t](#)

#### 12.89.3 Enumeration Type Documentation

##### 12.89.3.1 [CFE\\_ConfigType](#) [enum CFE\\_ConfigType](#)

###### Enumerator

<a href="#">CFE_ConfigType_UNDEFINED</a>	
<a href="#">CFE_ConfigType_VALUE</a>	Value is an unsigned int
<a href="#">CFE_ConfigType_STRING</a>	Value is a string pointer
<a href="#">CFE_ConfigType_POINTER</a>	Value is a non-string object pointer
<a href="#">CFE_ConfigType_ARRAY</a>	Value is a combination of length and pointer

Definition at line 34 of file cfe\_config\_table.h.

## 12.90 cfe/modules/core\_api/config/default\_cfe\_core\_api\_basemsgid\_values.h File Reference

### Macros

- `#define CFE_PLATFORM_BASE_MIDVAL(x) DEFAULT_CFE_PLATFORM_##x##_MID_BASE`  
*Platform-specific MSGID base value.*
- `#define CFE_GLOBAL_BASE_MIDVAL(x) DEFAULT_GLOBAL_##x##_MID_BASE`

#### 12.90.1 Detailed Description

This header file contains the platform-specific base msg ID values and logic to convert a topic ID to a message ID value. This file is just a stand-in to permit CFE to build initially. It is expected that the user will provide the preferred values in a custom header.

Conventionally, CFS binaries are compiled for a specific CPU instance, with the MSGIDs for that CPU compiled directly into the binaries. This example file implements that paradigm. There should be a separate definition of this file for every CFS instance in the mission.

The intent of this design is to permit use of a separate, single header that defines all of the msgid values for all cpus, for example:

```
#define MY_CPU1_CMD_MID_BASE 0x1A00
#define MY_CPU2_CMD_MID_BASE 0x1B00
#define MY_CPU3_CMD_MID_BASE 0x1C00
....
```

Then this file can include that combined base msgid header, and define `CFE_PLATFORM_BASE_MIDVAL()` to cherry-pick one of the values within that set. For example on CPU2 this can be defined as:

```
#define CFE_PLATFORM_BASE_MIDVAL(x) MY_CPU2_##x##_MID_BASE
```

#### 12.90.2 Macro Definition Documentation

##### 12.90.2.1 CFE\_GLOBAL\_BASE\_MIDVAL `#define CFE_GLOBAL_BASE_MIDVAL(`

```
    x ) DEFAULT_GLOBAL_##x##_MID_BASE
```

Definition at line 62 of file default\_cfe\_core\_api\_basemsgid\_values.h.

##### 12.90.2.2 CFE\_PLATFORM\_BASE\_MIDVAL `#define CFE_PLATFORM_BASE_MIDVAL(`

```
    x ) DEFAULT_CFE_PLATFORM_##x##_MID_BASE
```

Platform-specific MSGID base value.

Definition at line 61 of file default\_cfe\_core\_api\_basemsgid\_values.h.

## 12.91 cfe/modules/core\_api/config/default\_cfe\_core\_api\_interface\_cfg\_values.h File Reference

### Macros

- `#define CFE_MISSION_CORE_API_CFGVAL(x) DEFAULT_CFE_MISSION_CORE_API_##x`

#### 12.91.1 Detailed Description

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

#### 12.91.2 Macro Definition Documentation

**12.91.2.1 CFE\_MISSION\_CORE\_API\_CFGVAL** #define CFE\_MISSION\_CORE\_API\_CFGVAL( x ) DEFAULT\_CFE\_MISSION\_CORE\_API\_##x

Definition at line 31 of file default\_cfe\_core\_api\_interface\_cfg\_values.h.

## 12.92 cfe/modules/core\_api/config/default\_cfe\_core\_api\_msgid\_mapping.h File Reference

```
#include "cfe_core_api_base_msgid_values.h"
```

### Macros

- #define CFE\_PLATFORM\_CMD\_TOPICID\_TO\_MIDV(topic) (CFE\_PLATFORM\_BASE\_MIDVAL(CMD) | (topic))  
*Convert a command topic ID to a MsgID value.*
- #define DEFAULT\_CFE\_PLATFORM\_CMD\_MID\_BASE 0x1800
- #define CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV(topic) (CFE\_PLATFORM\_BASE\_MIDVAL(TLM) | (topic))  
*Convert a telemetry topic ID to a MsgID value.*
- #define DEFAULT\_CFE\_PLATFORM\_TLM\_MID\_BASE 0x0800
- #define CFE\_GLOBAL\_CMD\_TOPICID\_TO\_MIDV(topic) (CFE\_GLOBAL\_BASE\_MIDVAL(CMD) | (topic))  
*Convert a "global" command topic ID to a MsgID value.*
- #define DEFAULT\_GLOBAL\_CMD\_MID\_BASE 0x1860
- #define CFE\_GLOBAL\_TLM\_TOPICID\_TO\_MIDV(topic) (CFE\_GLOBAL\_BASE\_MIDVAL(TLM) | (topic))  
*Convert a "global" telemetry topic ID to a MsgID value.*
- #define DEFAULT\_GLOBAL\_TLM\_MID\_BASE 0x0860

### 12.92.1 Detailed Description

This header file contains the logic to convert a topic ID to a message ID value.

In an conventional deployment, the macro simply combines the base MID with the topic ID to produce the MID value. It is simple bitwise "OR" operation.

This logic is intended to be customizable. By overriding this file, these macros can be replaced with whatever logic is desired. However, the same logic must be used across all instances. In order to simply tune the base value used for a given instance, only the cfs\_base\_msgids.h file needs to be overridden.

### 12.92.2 Macro Definition Documentation

**12.92.2.1 CFE\_GLOBAL\_CMD\_TOPICID\_TO\_MIDV** #define CFE\_GLOBAL\_CMD\_TOPICID\_TO\_MIDV( topic ) (CFE\_GLOBAL\_BASE\_MIDVAL(CMD) | (topic))

Convert a "global" command topic ID to a MsgID value.

A global command is one that is not specific to an individual instance of CFE, but rather intended to be broadcast to all CFE instances at the same time.

This is otherwise identical to [CFE\\_PLATFORM\\_CMD\\_TOPICID\\_TO\\_MIDV](#)

Definition at line 74 of file default\_cfe\_core\_api\_msgid\_mapping.h.

**12.92.2.2 CFE\_GLOBAL\_TLM\_TOPICID\_TO\_MIDV** #define CFE\_GLOBAL\_TLM\_TOPICID\_TO\_MIDV( topic ) (CFE\_GLOBAL\_BASE\_MIDVAL(TLM) | (topic))

Convert a "global" telemetry topic ID to a MsgID value.

A global telemetry is one that is not specific to an individual instance of CFE, but rather intended to be broadcast to all CFE instances at the same time.

This is otherwise identical to [CFE\\_PLATFORM\\_TLM\\_TOPICID\\_TO\\_MIDV](#)

Definition at line 85 of file default\_cfe\_core\_api\_msgid\_mapping.h.

```
12.92.2.3 CFE_PLATFORM_CMD_TOPICID_TO_MIDV #define CFE_PLATFORM_CMD_TOPICID_TO_MIDV(  
    topic) (CFE_PLATFORM_BASE_MIDVAL(CMD) | (topic))
```

Convert a command topic ID to a MsgID value.

This defines the logic to convert a topic ID value into a message ID value. This operates on integer values and should resolve at compile time such that it can be used in e.g. switch/case statements.

#### Note

The result of this conversion is a simple integer, thus also needs to go through [CFE\\_SB\\_ValueToMsgId\(\)](#) to obtain a properly-typed [CFE\\_SB\\_MsgId\\_t](#) for interacting with SB APIs.

Definition at line 49 of file default\_cfe\_core\_api\_msgid\_mapping.h.

```
12.92.2.4 CFE_PLATFORM_TLM_TOPICID_TO_MIDV #define CFE_PLATFORM_TLM_TOPICID_TO_MIDV(  
    topic) (CFE_PLATFORM_BASE_MIDVAL(TLM) | (topic))
```

Convert a telemetry topic ID to a MsgID value.

This defines the logic to convert a topic ID value into a message ID value. This operates on integer values and should resolve at compile time such that it can be used in e.g. switch/case statements.

#### Note

The result of this conversion is a simple integer, thus also needs to go through [CFE\\_SB\\_ValueToMsgId\(\)](#) to obtain a properly-typed [CFE\\_SB\\_MsgId\\_t](#) for interacting with SB APIs.

Definition at line 63 of file default\_cfe\_core\_api\_msgid\_mapping.h.

```
12.92.2.5 DEFAULT_CFE_PLATFORM_CMD_MID_BASE #define DEFAULT_CFE_PLATFORM_CMD_MID_BASE 0x1800
```

Definition at line 50 of file default\_cfe\_core\_api\_msgid\_mapping.h.

```
12.92.2.6 DEFAULT_CFE_PLATFORM_TLM_MID_BASE #define DEFAULT_CFE_PLATFORM_TLM_MID_BASE 0x0800
```

Definition at line 64 of file default\_cfe\_core\_api\_msgid\_mapping.h.

```
12.92.2.7 DEFAULT_GLOBAL_CMD_MID_BASE #define DEFAULT_GLOBAL_CMD_MID_BASE 0x1860
```

Definition at line 75 of file default\_cfe\_core\_api\_msgid\_mapping.h.

```
12.92.2.8 DEFAULT_GLOBAL_TLM_MID_BASE #define DEFAULT_GLOBAL_TLM_MID_BASE 0x0860
```

Definition at line 86 of file default\_cfe\_core\_api\_msgid\_mapping.h.

## 12.93 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"
```

### 12.93.1 Detailed Description

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

## 12.94 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"
```

### 12.94.1 Detailed Description

Purpose: This header file contains the Message Id's for messages used by the cFE core.

## 12.95 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"
```

### 12.95.1 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.

## 12.96 cfe/modules/core\_api/fsw/inc/cfe\_config.h File Reference

```
#include "common_types.h"
#include "cfe_config_api_typedefs.h"
#include "cfe_config_ids.h"
```

### Functions

- `uint32 CFE_Config_GetValue (CFE_ConfigId_t ConfigId)`  
*Obtain an integer value correlating to an CFE configuration ID.*
- `const void * CFE_Config_GetObjPointer (CFE_ConfigId_t ConfigId)`  
*Obtain a pointer value correlating to an CFE configuration ID.*

- `CFE_Config_ArrayValue_t CFE_Config_GetArrayValue (CFE_ConfigId_t ConfigId)`  
*Obtain an array correlating to an CFE configuration ID.*
- `const char * CFE_Config_GetString (CFE_ConfigId_t ConfigId)`  
*Obtain a string value correlating to an CFE configuration ID.*
- `const char * CFE_Config_getName (CFE_ConfigId_t ConfigId)`  
*Obtain the name of a CFE configuration ID.*
- `CFE_ConfigId_t CFE_Config_getIdByName (const char *Name)`  
*Obtain the ID value associated with a configuration name.*
- `void CFE_Config_IterateAll (void *Arg, CFE_Config_Callback_t Callback)`  
*Iterate all known name/ID value pairs.*
- `void CFE_Config_GetVersionString (char *Buf, size_t Size, const char *Component, const char *SrcVersion, const char *CodeName, const char *LastOffcRel)`  
*Obtain the version string for a cFS component or app.*

### 12.96.1 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

### 12.96.2 Function Documentation

#### 12.96.2.1 CFE\_Config\_GetArrayValue() `CFE_Config_ArrayValue_t CFE_Config_GetArrayValue (` `CFE_ConfigId_t ConfigId )`

Obtain an array correlating to an CFE configuration ID.

Retrieves the CFE\_Config\_ArrayValue\_t value associated with the specified key. This combines an array length (number of elements) and a pointer to the first element.

If no value has been set, or the key is not valid, this returns 0 / NULL.

##### Parameters

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

##### Returns

Value associated with key

##### Return values

NULL	if key is not defined or not set
------	----------------------------------

#### 12.96.2.2 CFE\_Config\_getIdByName() `CFE_ConfigId_t CFE_Config_getIdByName (` `const char * Name )`

Obtain the ID value associated with a configuration name.

**Parameters**

in	<i>Name</i>	The name of the ID to look up
----	-------------	-------------------------------

**Returns**

ID associated with name

**Return values**

<i>CFE_CONFIGID_UNDEFINED</i>	if the name did not correspond to a key
-------------------------------	---

**12.96.2.3 CFE\_Config\_GetName()** `const char* CFE_Config_GetName ( CFE_ConfigId_t ConfigId )`

Obtain the name of a CFE configuration ID.

Retrieves the printable name associated with the specified key.

**Note**

This function does not return NULL.

If the ID is not valid/known, then the implementation returns the special string '[unknown]' rather than NULL, so this function may be more easily used in printf() style calls.

**Parameters**

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

**Returns**

Name associated with key

**12.96.2.4 CFE\_Config\_GetObjPointer()** `const void* CFE_Config_GetObjPointer ( CFE_ConfigId_t ConfigId )`

Obtain a pointer value correlating to an CFE configuration ID.

Retrieves the pointer value associated with the specified key.

If no value has been set, or the key is not valid, this returns NULL.

**Parameters**

in	<i>ConfigId</i>	Configuration ID/Key to look up
----	-----------------	---------------------------------

**Returns**

Value associated with key

## Return values

<code>NULL</code>	if key is not defined or not set
-------------------	----------------------------------

**12.96.2.5 CFE\_Config\_GetString()** `const char* CFE_Config_GetString ( CFE_ConfigId_t ConfigId )`

Obtain a string value correlating to an CFE configuration ID.

Retrieves the string value associated with the specified key.

If no value has been set, or the key is not valid, this returns the special string "UNDEFINED"

## Note

This function does not return NULL, so it can be used directly in printf-style calls.

## Parameters

<code>in</code>	<code>ConfigId</code>	Configuration ID/Key to look up
-----------------	-----------------------	---------------------------------

## Returns

String value associated with key

**12.96.2.6 CFE\_Config\_GetValue()** `uint32 CFE_Config_GetValue ( CFE_ConfigId_t ConfigId )`

Obtain an integer value correlating to an CFE configuration ID.

Retrieves the integer value associated with the specified key.

If no value has been set, or the key is not valid, this returns 0.

## Parameters

<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
----------------	----------------------------------

**12.96.2.7 CFE\_Config\_GetVersionString()** `void CFE_Config_GetVersionString (`

```
char * Buf,
size_t Size,
const char * Component,
```

```
const char * SrcVersion,
const char * CodeName,
const char * LastOffcRel )
```

Obtain the version string for a cFS component or app.

Assembles a standardized version string associated with the specified component/app.

#### Parameters

in	<i>Buf</i>	Buffer to place version string in. Will be populated with standard version string containing the provided parameters (i.e.: "cFE DEVELOPMENT BUILD equuleus-rc1+dev0 (Codename equuleus), Last Official Release: cFE 6.7.0"
in	<i>Size</i>	Size of the provided buffer
in	<i>Component</i>	Component for which to get version string (i.e. "cFE")
in	<i>SrcVersion</i>	Source version identifier (i.e. "equuleus-rc1+dev0")
in	<i>CodeName</i>	Code name for the build (i.e. "equuleus")
in	<i>LastOffcRel</i>	Last official release (i.e. "6.7.0")

### 12.96.2.8 CFE\_Config\_IterateAll()

```
void CFE_Config_IterateAll (
    void * Arg,
    CFE_Config_Callback_t Callback )
```

Iterate all known name/ID value pairs.

#### Parameters

in	<i>Arg</i>	User-supplied opaque argument to pass to callback
in	<i>Callback</i>	User-supplied callback function to invoke for each ID

## 12.97 cfe/modules/core\_api/fsw/inc/cfe\_config\_api\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_resourceid_api_typedefs.h"
```

### Data Structures

- struct **CFE\_Config\_ArrayValue**

*Wrapper type for array configuration.*

### Macros

- #define **CFE\_CONFIGID\_C**(val) ((**CFE\_ConfigId\_t**)CFE\_RESOURCEID\_WRAP(val))
- #define **CFE\_CONFIGID\_UNDEFINED** CFE\_CONFIGID\_C(CFE\_RESOURCEID\_UNDEFINED)

### Typedefs

- typedef **CFE\_RESOURCEID\_BASE\_TYPE** **CFE\_ConfigId\_t**  
*A type for Configuration IDs.*
- typedef void(\* **CFE\_Config\_Callback\_t**) (void \*Arg, **CFE\_ConfigId\_t** Id, const char \*Name)
- typedef struct **CFE\_Config\_ArrayValue** **CFE\_Config\_ArrayValue\_t**

*Wrapper type for array configuration.*

### 12.97.1 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

### 12.97.2 Macro Definition Documentation

**12.97.2.1 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.

**12.97.2.2 CFE\_CONFIGID\_UNDEFINED** `#define CFE_CONFIGID_UNDEFINED CFE_CONFIGID_C(CFE_RESOURCEID_UNDEFINED)`

Definition at line 49 of file cfe\_config\_api\_typedefs.h.

### 12.97.3 Typedef Documentation

**12.97.3.1 CFE\_Config\_ArrayValue\_t** `typedef struct CFE_Config_ArrayValue CFE_Config_ArrayValue_t`  
Wrapper type for array configuration.

This is a pair containing a size and pointer that is get/set via a single config table entry

**12.97.3.2 CFE\_Config\_Callback\_t** `typedef void(* CFE_Config_Callback_t)(void *Arg, CFE_ConfigId_t`  
`Id, const char *Name)`

Definition at line 51 of file cfe\_config\_api\_typedefs.h.

**12.97.3.3 CFE\_ConfigId\_t** `typedef CFE_RESOURCEID_BASE_TYPE CFE_ConfigId_t`  
A type for Configuration IDs.

This is the type that is used for any API accepting or returning a configuration key ID

Definition at line 46 of file cfe\_config\_api\_typedefs.h.

## 12.98 cfe/modules/core\_api/fsw/inc/cfe\_core\_api\_base\_msgids.h File Reference

`#include "cfe_core_api_msgid_mapping.h"`

### 12.98.1 Detailed Description

This header file contains the logic to convert a topic ID to a message ID value.

**Note**

Traditionally in CFS the MsgID values were translated using an offset added to a base value that was specific to that CFS instance. The name of this file reflects that heritage and is preserved for backward compatibility with existing apps.

Although this file is called "base\_msgids.h" the translation can take on any form, even a runtime lookup. The file name of "cfe\_core\_api\_msgid\_mapping.h" better conveys the intent that this is not limited to a base+offset paradigm. This file is now just an alias/wrapper around the msgid\_mapping header file to allow apps and libraries to continue building.

To customize the mapping, override "cfe\_core\_api\_msgid\_mapping.h"

## 12.99 cfe/modules/core\_api/fsw/inc/cfe\_core\_api\_interface\_cfg.h File Reference

```
#include "cfe_core_api_interface_cfg_values.h"
```

**Macros**

- #define CFE\_MISSION\_MAX\_PATH\_LEN CFE\_MISSION\_CORE\_API\_CFGVAL(MAX\_PATH\_LEN)
- #define DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_PATH\_LEN 64
- #define CFE\_MISSION\_MAX\_FILE\_LEN CFE\_MISSION\_CORE\_API\_CFGVAL(MAX\_FILE\_LEN)
- #define DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_FILE\_LEN 20
- #define CFE\_MISSION\_MAX\_API\_LEN CFE\_MISSION\_CORE\_API\_CFGVAL(MAX\_API\_LEN)
- #define DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_API\_LEN 20
- #define CFE\_MISSION\_MAX\_NUM\_FILES CFE\_MISSION\_CORE\_API\_CFGVAL(MAX\_NUM\_FILES)
- #define DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_NUM\_FILES 50

### 12.99.1 Detailed Description

Purpose: This header file contains the mission configuration parameters and typedefs with mission scope.

### 12.99.2 Macro Definition Documentation

#### 12.99.2.1 CFE\_MISSION\_MAX\_API\_LEN #define CFE\_MISSION\_MAX\_API\_LEN CFE\_MISSION\_CORE\_API\_CFGVAL (MAX↔\_API\_LEN)

**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 112 of file cfe\_core\_api\_interface\_cfg.h.

**12.99.2.2 CFE\_MISSION\_MAX\_FILE\_LEN** `#define CFE_MISSION_MAX_FILE_LEN CFE_MISSION_CORE_API_CFGVAL(MAX←_FILE_LEN)`

**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 85 of file cfe\_core\_api\_interface\_cfg.h.

**12.99.2.3 CFE\_MISSION\_MAX\_NUM\_FILES** `#define CFE_MISSION_MAX_NUM_FILES CFE_MISSION_CORE_API_CFGVAL(MAX←_NUM_FILES)`

**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 135 of file cfe\_core\_api\_interface\_cfg.h.

**12.99.2.4 CFE\_MISSION\_MAX\_PATH\_LEN** #define CFE\_MISSION\_MAX\_PATH\_LEN CFE\_MISSION\_CORE\_API\_CFGVAL(MAX←\_PATH\_LEN)

**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 57 of file cfe\_core\_api\_interface\_cfg.h.

**12.99.2.5 DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_API\_LEN** #define DEFAULT\_CFE\_MISSION\_CORE\_API←\_MAX\_API\_LEN 20

Definition at line 113 of file cfe\_core\_api\_interface\_cfg.h.

**12.99.2.6 DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_FILE\_LEN** #define DEFAULT\_CFE\_MISSION\_CORE\_API←\_MAX\_FILE\_LEN 20

Definition at line 86 of file cfe\_core\_api\_interface\_cfg.h.

**12.99.2.7 DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_NUM\_FILES** #define DEFAULT\_CFE\_MISSION\_CORE←\_API\_MAX\_NUM\_FILES 50

Definition at line 136 of file cfe\_core\_api\_interface\_cfg.h.

**12.99.2.8 DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_PATH\_LEN** #define DEFAULT\_CFE\_MISSION\_CORE\_API←\_MAX\_PATH\_LEN 64

Definition at line 58 of file cfe\_core\_api\_interface\_cfg.h.

## 12.100 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))

### 12.100.1 Detailed Description

Purpose: Define macros to enforce big-endian/network byte order for 16 and 32 bit integers

### 12.100.2 Macro Definition Documentation

**12.100.2.1 CFE\_MAKE\_BIG16** #define CFE\_MAKE\_BIG16(  
 $n$ ) (((n)&0x00FF) << 8) | (((n)&0xFF00) >> 8))

Definition at line 64 of file cfe\_endian.h.

**12.100.2.2 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.

## 12.101 cfe/modules/core\_api/fsw/inc/cfe\_error.h File Reference

```
#include "osapi.h"
```

### Macros

- #define CFE\_STATUS\_C(X) ((CFE\_Status\_t)(X))  
*cFE Status macro for literal*
- #define CFE\_STATUS\_STRING\_LENGTH 11  
*cFE Status converted to string length limit*
- #define CFE\_SEVERITY\_BITMASK ((CFE\_Status\_t)0xc0000000)  
*Error Severity Bitmask.*
- #define CFE\_SEVERITY\_SUCCESS ((CFE\_Status\_t)0x00000000)  
*Severity Success.*
- #define CFE\_SEVERITY\_INFO ((CFE\_Status\_t)0x40000000)  
*Severity Info.*
- #define CFE\_SEVERITY\_ERROR ((CFE\_Status\_t)0xc0000000)  
*Severity Error.*
- #define CFE\_SERVICE\_BITMASK ((CFE\_Status\_t)0x0e000000)  
*Error Service Bitmask.*
- #define CFE\_EVENTS\_SERVICE ((CFE\_Status\_t)0x02000000)  
*Event Service.*
- #define CFE\_EXECUTIVE\_SERVICE ((CFE\_Status\_t)0x04000000)  
*Executive Service.*
- #define CFE\_FILE\_SERVICE ((CFE\_Status\_t)0x06000000)  
*File Service.*
- #define CFE\_GENERIC\_SERVICE ((CFE\_Status\_t)0x08000000)  
*Generic Service.*
- #define CFE\_SOFTWARE\_BUS\_SERVICE ((CFE\_Status\_t)0x0a000000)  
*Software Bus Service.*
- #define CFE\_TABLE\_SERVICE ((CFE\_Status\_t)0x0c000000)

*Table Service.*

- #define CFE\_TIME\_SERVICE ((CFE\_Status\_t)0x0e000000)  
*Time Service.*
- #define CFE\_SUCCESS ((CFE\_Status\_t)0)  
*Successful execution.*
- #define CFE\_STATUS\_NO\_COUNTER\_INCREMENT ((CFE\_Status\_t)0x48000001)  
*No Counter Increment.*
- #define CFE\_STATUS\_WRONG\_MSG\_LENGTH ((CFE\_Status\_t)0xc8000002)  
*Wrong Message Length.*
- #define CFE\_STATUS\_UNKNOWN\_MSG\_ID ((CFE\_Status\_t)0xc8000003)  
*Unknown Message ID.*
- #define CFE\_STATUS\_BAD\_COMMAND\_CODE ((CFE\_Status\_t)0xc8000004)  
*Bad Command Code.*
- #define CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL ((CFE\_Status\_t)0xc8000005)  
*External failure.*
- #define CFE\_STATUS\_REQUEST\_ALREADY\_PENDING ((int32)0xc8000006)  
*Request already pending.*
- #define CFE\_STATUS\_VALIDATION\_FAILURE ((int32)0xc8000007)  
*Request or input value failed basic structural validation.*
- #define CFE\_STATUS\_RANGE\_ERROR ((int32)0xc8000008)  
*Request or input value is out of range.*
- #define CFE\_STATUS\_INCORRECT\_STATE ((int32)0xc8000009)  
*Cannot process request at this time.*
- #define CFE\_STATUS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc800ffff)  
*Not Implemented.*
- #define CFE\_EVS\_UNKNOWN\_FILTER ((CFE\_Status\_t)0xc2000001)  
*Unknown Filter.*
- #define CFE\_EVS\_APP\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000002)  
*Application Not Registered.*
- #define CFE\_EVS\_APP\_ILLEGAL\_APP\_ID ((CFE\_Status\_t)0xc2000003)  
*Illegal Application ID.*
- #define CFE\_EVS\_APP\_FILTER\_OVERLOAD ((CFE\_Status\_t)0xc2000004)  
*Application Filter Overload.*
- #define CFE\_EVS\_RESET\_AREA\_POINTER ((CFE\_Status\_t)0xc2000005)  
*Reset Area Pointer Failure.*
- #define CFE\_EVS\_EVT\_NOT\_REGISTERED ((CFE\_Status\_t)0xc2000006)  
*Event Not Registered.*
- #define CFE\_EVS\_FILE\_WRITE\_ERROR ((CFE\_Status\_t)0xc2000007)  
*File Write Error.*
- #define CFE\_EVS\_INVALID\_PARAMETER ((CFE\_Status\_t)0xc2000008)  
*Invalid Pointer.*
- #define CFE\_EVS\_APP\_SQUELCHED ((CFE\_Status\_t)0xc2000009)  
*Event squelched.*
- #define CFE\_EVS\_NOT\_IMPLEMENTED ((CFE\_Status\_t)0xc200ffff)  
*Not Implemented.*
- #define CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID ((CFE\_Status\_t)0xc4000001)  
*Resource ID is not valid.*

- #define CFE\_ES\_ERR\_NAME\_NOT\_FOUND ((CFE\_Status\_t)0xc4000002)  
*Resource Name Error.*
- #define CFE\_ES\_ERR\_APP\_CREATE ((CFE\_Status\_t)0xc4000004)  
*Application Create Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_CREATE ((CFE\_Status\_t)0xc4000005)  
*Child Task Create Error.*
- #define CFE\_ES\_ERR\_SYS\_LOG\_FULL ((CFE\_Status\_t)0xc4000006)  
*System Log Full.*
- #define CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE ((CFE\_Status\_t)0xc4000008)  
*Memory Block Size Error.*
- #define CFE\_ES\_ERR\_LOAD\_LIB ((CFE\_Status\_t)0xc4000009)  
*Load Library Error.*
- #define CFE\_ES\_BAD\_ARGUMENT ((CFE\_Status\_t)0xc400000a)  
*Bad Argument.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER ((CFE\_Status\_t)0xc400000b)  
*Child Task Register Error.*
- #define CFE\_ES\_CDS\_ALREADY\_EXISTS ((CFE\_Status\_t)0x4400000d)  
*CDS Already Exists.*
- #define CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY ((CFE\_Status\_t)0xc400000e)  
*CDS Insufficient Memory.*
- #define CFE\_ES\_CDS\_INVALID\_NAME ((CFE\_Status\_t)0xc400000f)  
*CDS Invalid Name.*
- #define CFE\_ES\_CDS\_INVALID\_SIZE ((CFE\_Status\_t)0xc4000010)  
*CDS Invalid Size.*
- #define CFE\_ES\_CDS\_INVALID ((CFE\_Status\_t)0xc4000012)  
*CDS Invalid.*
- #define CFE\_ES\_CDS\_ACCESS\_ERROR ((CFE\_Status\_t)0xc4000013)  
*CDS Access Error.*
- #define CFE\_ES\_FILE\_IO\_ERR ((CFE\_Status\_t)0xc4000014)  
*File IO Error.*
- #define CFE\_ES\_RST\_ACCESS\_ERR ((CFE\_Status\_t)0xc4000015)  
*Reset Area Access Error.*
- #define CFE\_ES\_ERR\_APP\_REGISTER ((CFE\_Status\_t)0xc4000017)  
*Application Register Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE ((CFE\_Status\_t)0xc4000018)  
*Child Task Delete Error.*
- #define CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK ((CFE\_Status\_t)0xc4000019)  
*Child Task Delete Passed Main Task.*
- #define CFE\_ES\_CDS\_BLOCK\_CRC\_ERR ((CFE\_Status\_t)0xc400001A)  
*CDS Block CRC Error.*
- #define CFE\_ES\_MUT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001B)  
*Mutex Semaphore Delete Error.*
- #define CFE\_ES\_BIN\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001C)  
*Binary Semaphore Delete Error.*
- #define CFE\_ES\_COUNT\_SEM\_DELETE\_ERR ((CFE\_Status\_t)0xc400001D)  
*Counting Semaphore Delete Error.*
- #define CFE\_ES\_QUEUE\_DELETE\_ERR ((CFE\_Status\_t)0xc400001E)

- #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\_INFO\_NO\_DUMP\_PENDING ((CFE\_Status\_t)0x4c00002e)  
*No Dump Pending.*
- #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.*

### 12.101.1 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

### 12.101.2 Macro Definition Documentation

#### 12.101.2.1 CFE\_EVENTS\_SERVICE #define CFE\_EVENTS\_SERVICE ((CFE\_Status\_t)0x02000000)

Event Service.

Definition at line 126 of file cfe\_error.h.

**12.101.2.2 CFE\_EXECUTIVE\_SERVICE** #define CFE\_EXECUTIVE\_SERVICE ((CFE\_Status\_t)0x04000000)  
Executive Service.  
Definition at line 127 of file cfe\_error.h.

**12.101.2.3 CFE\_FILE\_SERVICE** #define CFE\_FILE\_SERVICE ((CFE\_Status\_t)0x06000000)  
File Service.  
Definition at line 128 of file cfe\_error.h.

**12.101.2.4 CFE\_GENERIC\_SERVICE** #define CFE\_GENERIC\_SERVICE ((CFE\_Status\_t)0x08000000)  
Generic Service.  
Definition at line 129 of file cfe\_error.h.

**12.101.2.5 CFE\_SERVICE\_BITMASK** #define CFE\_SERVICE\_BITMASK ((CFE\_Status\_t)0x0e000000)  
Error Service Bitmask.  
Definition at line 124 of file cfe\_error.h.

**12.101.2.6 CFE\_SEVERITY\_BITMASK** #define CFE\_SEVERITY\_BITMASK ((CFE\_Status\_t)0xc0000000)  
Error Severity Bitmask.  
Definition at line 115 of file cfe\_error.h.

**12.101.2.7 CFE\_SEVERITY\_ERROR** #define CFE\_SEVERITY\_ERROR ((CFE\_Status\_t)0xc0000000)  
Severity Error.  
Definition at line 119 of file cfe\_error.h.

**12.101.2.8 CFE\_SEVERITY\_INFO** #define CFE\_SEVERITY\_INFO ((CFE\_Status\_t)0x40000000)  
Severity Info.  
Definition at line 118 of file cfe\_error.h.

**12.101.2.9 CFE\_SEVERITY\_SUCCESS** #define CFE\_SEVERITY\_SUCCESS ((CFE\_Status\_t)0x00000000)  
Severity Success.  
Definition at line 117 of file cfe\_error.h.

**12.101.2.10 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.

**12.101.2.11 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.

**12.101.2.12 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.

**12.101.2.13 CFE\_TABLE\_SERVICE** #define CFE\_TABLE\_SERVICE ((CFE\_Status\_t)0x0c000000)  
 Table Service.  
 Definition at line 131 of file cfe\_error.h.

**12.101.2.14 CFE\_TIME\_SERVICE** #define CFE\_TIME\_SERVICE ((CFE\_Status\_t)0x0e000000)  
 Time Service.  
 Definition at line 132 of file cfe\_error.h.

### 12.101.3 Typedef Documentation

**12.101.3.1 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.

**12.101.3.2 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.

### 12.101.4 Function Documentation

**12.101.4.1 CFE\_ES\_StatusToString()** char\* CFE\_ES\_StatusToString (

CFE_Status_t	status,
	CFE_StatusString_t * status_string )

Convert status to a string.

#### Parameters

in	status	Status value to convert
out	status_string	Buffer to store status converted to string

#### Returns

Passed in string pointer

### 12.102 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_PerfLogEntry(id) (CFE_ES_PerfLogAdd(id, 0))`  
*Entry marker for use with Software Performance Analysis Tool.*
- `#define CFE_ES_PerfLogExit(id) (CFE_ES_PerfLogAdd(id, 1))`  
*Exit marker for use with Software Performance Analysis Tool.*

## Functions

- `CFE_Status_t CFE_ES_AppID_ToIndex (CFE_ES_AppId_t AppID, uint32 *Idx)`  
*Obtain an index value correlating to an ES Application ID.*
- `int32 CFE_ES_LibID_ToIndex (CFE_ES_LibId_t LibId, uint32 *Idx)`  
*Obtain an index value correlating to an ES Library ID.*
- `CFE_Status_t CFE_ES_TaskID_ToIndex (CFE_ES_TaskId_t TaskID, uint32 *Idx)`  
*Obtain an index value correlating to an ES Task ID.*
- `CFE_Status_t CFE_ES_CounterID_ToIndex (CFE_ES_CounterId_t CounterId, uint32 *Idx)`  
*Obtain an index value correlating to an ES Counter ID.*
- `void CFE_ES_Main (uint32 StartType, uint32 StartSubtype, uint32 ModelId, const char *StartFilePath)`  
*cFE Main Entry Point used by Board Support Package to start cFE*
- `CFE_Status_t CFE_ES_ResetCFE (uint32 ResetType)`  
*Reset the cFE Core and all cFE Applications.*
- `CFE_Status_t CFE_ES_RestartApp (CFE_ES_AppId_t AppID)`  
*Restart a single cFE Application.*
- `CFE_Status_t CFE_ES_ReloadApp (CFE_ES_AppId_t AppID, const char *AppFileName)`  
*Reload a single cFE Application.*
- `CFE_Status_t CFE_ES_DeleteApp (CFE_ES_AppId_t AppID)`  
*Delete a cFE Application.*
- `void CFE_ES_ExitApp (uint32 ExitStatus)`  
*Exit a cFE Application.*
- `bool CFE_ES_RunLoop (uint32 *RunStatus)`  
*Check for Exit, Restart, or Reload commands.*
- `CFE_Status_t CFE_ES_WaitForSystemState (uint32 MinSystemState, uint32 TimeOutMilliseconds)`  
*Allow an Application to Wait for a minimum global system state.*
- `void CFE_ES_WaitForStartupSync (uint32 TimeOutMilliseconds)`  
*Allow an Application to Wait for the "OPERATIONAL" global system state.*
- `void CFE_ES_IncrementTaskCounter (void)`  
*Increments the execution counter for the calling task.*
- `int32 CFE_ES_GetResetType (uint32 *ResetSubtypePtr)`  
*Return the most recent Reset Type.*
- `CFE_Status_t CFE_ES_GetAppID (CFE_ES_AppId_t *AppIdPtr)`  
*Get an Application ID for the calling Application.*
- `CFE_Status_t CFE_ES_GetTaskID (CFE_ES_TaskId_t *TaskIdPtr)`  
*Get the task ID of the calling context.*
- `CFE_Status_t CFE_ES_GetAppIDByName (CFE_ES_AppId_t *AppIdPtr, const char *AppName)`  
*Get an Application ID associated with a specified Application name.*
- `CFE_Status_t CFE_ES_GetLibIDByName (CFE_ES_LibId_t *LibIdPtr, const char *LibName)`  
*Get a Library ID associated with a specified Library name.*

- **CFE\_Status\_t CFE\_ES\_GetAppName** (char \*AppName, **CFE\_ES\_AppId\_t** AppId, size\_t BufferLength)  
*Get an Application name for a specified Application ID.*
- **CFE\_Status\_t CFE\_ES\_GetLibName** (char \*LibName, **CFE\_ES\_LibId\_t** LibId, size\_t BufferLength)  
*Get a Library name for a specified Library ID.*
- **CFE\_Status\_t CFE\_ES\_GetAppInfo** (**CFE\_ES\_AppInfo\_t** \*AppInfo, **CFE\_ES\_AppId\_t** AppId)  
*Get Application Information given a specified App ID.*
- **CFE\_Status\_t CFE\_ES\_GetTaskInfo** (**CFE\_ES\_TaskInfo\_t** \*TaskInfo, **CFE\_ES\_TaskId\_t** TaskId)  
*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.*
- **CFE\_Status\_t CFE\_ES\_PoolCreateEx\_WithAlignment** (*CFE\_ES\_MemHandle\_t* \*PoolID, *void* \*MemPtr, *size\_t* Size, *uint16* NumBlockSizes, *const size\_t* \*BlockSizes, *bool* UseMutex, *size\_t* Alignment)
  - Implements CFE\_ES\_PoolCreateEx with added param for alignment added for coverage purposes*
- **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.*

### 12.102.1 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:

### 12.102.2 Macro Definition Documentation

#### 12.102.2.1 OS\_PRINTF #define OS\_PRINTF(

```
m,  
n )
```

Definition at line 50 of file cfe\_es.h.

## 12.103 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 OSAL change \*/  
*Indicates that the stack for the child task should be dynamically allocated.*

## Typedefs

- `typedef void(* CFE_ES_TaskEntryFuncPtr_t) (void)`  
*Required Prototype of Task Main Functions.*
- `typedef int32(* CFE_ES_LibraryEntryFuncPtr_t) (CFE_ES_LibId_t LibId)`  
*Required Prototype of Library Initialization Functions.*
- `typedef CFE_ES_TaskEntryFuncPtr_t CFE_ES_ChildTaskMainFuncPtr_t`  
*Compatible typedef for ES child task entry point.*
- `typedef void * CFE_ES_StackPointer_t`  
*Type for the stack pointer of tasks.*
- `typedef enum CFE_ES_CrcType_Enum CFE_ES_CrcType_Enum_t`  
*Checksum/CRC algorithm identifiers.*
- `typedef union CFE_ES_PoolAlign CFE_ES_PoolAlign_t`  
*Pool Alignment.*
- `typedef void * CFE_ES_MemPoolBuf_t`  
*Pointer type used for memory pool API.*

## Enumerations

- `enum CFE_ES_CrcType_Enum {  
 CFE_ES_CrcType_NONE = 0, CFE_ES_CrcType_16_CRC = 1, CFE_ES_CrcType_MAX = 2, CFE_ES_CrcType_CRC_16  
 = CFE_ES_CrcType_16_CRC,  
 CFE_ES_CrcType_CRC_8 = CFE_ES_CrcType_MAX + 1, CFE_ES_CrcType_CRC_32 = CFE_ES_CrcType_←  
 MAX + 2 }  
  
Checksum/CRC algorithm identifiers.`

### 12.103.1 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:

### 12.103.2 Macro Definition Documentation

#### 12.103.2.1 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.

#### 12.103.2.2 CFE\_ES\_APPID\_C #define CFE\_ES\_APPID\_C( val ) ((CFE\_ES\_AppId\_t)CFE\_RESOURCEID\_WRAP(val))

Definition at line 208 of file cfe\_es\_api\_typedefs.h.

#### 12.103.2.3 CFE\_ES\_APPID\_UNDEFINED #define CFE\_ES\_APPID\_UNDEFINED CFE\_ES\_APPID\_C(CFE\_RESOURCEID\_UNDEFINED)

Definition at line 220 of file cfe\_es\_api\_typedefs.h.

**12.103.2.4 CFE\_ES\_CDS\_BAD\_HANDLE** #define CFE\_ES\_CDS\_BAD\_HANDLE CFE\_ES\_CDHANDLE\_C(CFE\_RESOURCEID\_UNDEFINED)  
Definition at line 225 of file cfe\_es\_api\_typedefs.h.

**12.103.2.5 CFE\_ES\_CDHANDLE\_C** #define CFE\_ES\_CDHANDLE\_C(  
    val ) ((CFE\_ES\_CDHandle\_t)CFE\_RESOURCEID\_WRAP(val))

Definition at line 213 of file cfe\_es\_api\_typedefs.h.

**12.103.2.6 CFE\_ES\_COUNTERID\_C** #define CFE\_ES\_COUNTERID\_C(  
    val ) ((CFE\_ES\_CounterId\_t)CFE\_RESOURCEID\_WRAP(val))

Definition at line 211 of file cfe\_es\_api\_typedefs.h.

**12.103.2.7 CFE\_ES\_COUNTERID\_UNDEFINED** #define CFE\_ES\_COUNTERID\_UNDEFINED CFE\_ES\_COUNTERID\_C(CFE\_RESOURCEID\_UNDEFINED)  
Definition at line 223 of file cfe\_es\_api\_typedefs.h.

**12.103.2.8 CFE\_ES\_LIBID\_C** #define CFE\_ES\_LIBID\_C(  
    val ) ((CFE\_ES\_LibId\_t)CFE\_RESOURCEID\_WRAP(val))

Definition at line 210 of file cfe\_es\_api\_typedefs.h.

**12.103.2.9 CFE\_ES\_LIBID\_UNDEFINED** #define CFE\_ES\_LIBID\_UNDEFINED CFE\_ES\_LIBID\_C(CFE\_RESOURCEID\_UNDEFINED)  
Definition at line 222 of file cfe\_es\_api\_typedefs.h.

**12.103.2.10 CFE\_ES\_MEMHANDLE\_C** #define CFE\_ES\_MEMHANDLE\_C(  
    val ) ((CFE\_ES\_MemHandle\_t)CFE\_RESOURCEID\_WRAP(val))

Definition at line 212 of file cfe\_es\_api\_typedefs.h.

**12.103.2.11 CFE\_ES\_MEMHANDLE\_UNDEFINED** #define CFE\_ES\_MEMHANDLE\_UNDEFINED CFE\_ES\_MEMHANDLE\_C(CFE\_RESOURCEID\_UNDEFINED)  
Definition at line 224 of file cfe\_es\_api\_typedefs.h.

**12.103.2.12 CFE\_ES\_MEMPOOLBUF\_C** #define CFE\_ES\_MEMPOOLBUF\_C(  
    x ) ((CFE\_ES\_MemPoolBuf\_t)(x))

Conversion macro to create buffer pointer from another type.

In cases where the actual buffer pointer is computed, this macro aids in converting the computed address (typically an OSAL "cpuaddr" type) into a buffer pointer.

#### Note

Any address calculation needs to take machine alignment requirements into account.

Definition at line 193 of file cfe\_es\_api\_typedefs.h.

**12.103.2.13 CFE\_ES\_NO\_MUTEX** #define CFE\_ES\_NO\_MUTEX false

Indicates that the memory pool selection will not use a semaphore.

Definition at line 240 of file cfe\_es\_api\_typedefs.h.

**12.103.2.14 CFE\_ES\_STATIC\_POOL\_TYPE** #define CFE\_ES\_STATIC\_POOL\_TYPE ( size )

**Value:**

```
union
{
    CFE_ES_PoolAlign_t Align;
    uint8              Data[size];
}
```

Static Pool Type.

A macro to help instantiate static memory pools that are correctly aligned. This resolves to a union type that contains a member called "Data" that will be correctly aligned to be a memory pool and sized according to the argument.

Definition at line 160 of file cfe\_es\_api\_typedefs.h.

**12.103.2.15 CFE\_ES\_TASK\_STACK\_ALLOCATE** #define CFE\_ES\_TASK\_STACK\_ALLOCATE NULL /\* aka OS\_←  
TASK\_STACK\_ALLOCATE in proposed OSAL change \*/

Indicates that the stack for the child task should be dynamically allocated.

This value may be supplied as the Stack Pointer argument to CFE\_ES\_ChildTaskCreate() to indicate that the stack should be dynamically allocated.

Definition at line 237 of file cfe\_es\_api\_typedefs.h.

**12.103.2.16 CFE\_ES\_TASKID\_C** #define CFE\_ES\_TASKID\_C( val ) ((CFE\_ES\_TaskId\_t)CFE\_RESOURCEID\_WRAP(val))

Definition at line 209 of file cfe\_es\_api\_typedefs.h.

**12.103.2.17 CFE\_ES\_TASKID\_UNDEFINED** #define CFE\_ES\_TASKID\_UNDEFINED CFE\_ES\_TASKID\_C(CFE\_RESOURCEID\_UNDEFINED)

Definition at line 221 of file cfe\_es\_api\_typedefs.h.

**12.103.2.18 CFE\_ES\_USE\_MUTEX** #define CFE\_ES\_USE\_MUTEX true

Indicates that the memory pool selection will use a semaphore.

Definition at line 241 of file cfe\_es\_api\_typedefs.h.

### 12.103.3 Typedef Documentation

**12.103.3.1 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.

**12.103.3.2 CFE\_ES\_CrcType\_Enum\_t** typedef enum CFE\_ES\_CrcType\_Enum CFE\_ES\_CrcType\_Enum\_t

CHECKSUM/CRC algorithm identifiers.

Currently only CFE\_ES\_CrcType\_16\_ARC is supported.

All defined CRC algorithms should include a check value, which is the accepted result of computing the CRC across the fixed string "123456789"

**12.103.3.3 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.

#### 12.103.3.4 CFE\_ES\_MemPoolBuf\_t `typedef void* CFE_ES_MemPoolBuf_t`

Pointer type used for memory pool API.

This is used in the Get/Put API calls to refer to a pool buffer.

This pointer is expected to be type cast to the real object type after getting a new buffer. Using void\* allows this type conversion to occur easily.

##### Note

Older versions of CFE implemented the API using a uint32\*, which required explicit type casting everywhere it was called. Although the API type is now void\* to make usage easier, the pool buffers are aligned to machine requirements - typically 64 bits.

Definition at line 181 of file cfe\_es\_api\_typedefs.h.

#### 12.103.3.5 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.

#### 12.103.3.6 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.

#### 12.103.3.7 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.

### 12.103.4 Enumeration Type Documentation

#### 12.103.4.1 CFE\_ES\_CrcType\_Enum `enum CFE_ES_CrcType_Enum`

Checksum/CRC algorithm identifiers.

Currently only CFE\_ES\_CrcType\_16\_ARC is supported.

All defined CRC algorithms should include a check value, which is the accepted result of computing the CRC across the fixed string "123456789"

##### Enumerator

<code>CFE_ES_CrcType_NONE</code>	Reserved placeholder value. No computation is performed, always returns 0 as a result.
----------------------------------	--

**Enumerator**

CFE_ES_CrcType_16_ARC	Implementation of CRC-16/ARC.  Polynomial: 0x8005 Initialization: 0x0000 Reflect Input/Output: true Check value: 0xbb3d XorOut: 0x0000
CFE_ES_CrcType_MAX	Placeholder for end of normal enumeration list This should reflect the number of algorithms defined.
CFE_ES_CrcType_CRC_16	CRC-16 historically implied CRC-16/ARC CRC-8 historically defined but not implemented, value must not be 0
CFE_ES_CrcType_CRC_8	CRC-32 historically defined but not implemented, value must not be 0
CFE_ES_CrcType_CRC_32	

Definition at line 94 of file cfe\_es\_api\_typedefs.h.

### 12.104 cfe/modules/core\_api/fsw/inc/cfe\_evs.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_evs_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
#include "cfe_time_api_typedefs.h"
```

#### Macros

- #define CFE\_EVS\_Send(E, T, ...) CFE\_EVS\_SendEvent((E), CFE\_EVS\_EventType\_##T, \_\_VA\_ARGS\_\_)
- #define CFE\_EVS\_SendDbg(E, ...) CFE\_EVS\_Send(E, DEBUG, \_\_VA\_ARGS\_\_)
- #define CFE\_EVS\_SendInfo(E, ...) CFE\_EVS\_Send(E, INFORMATION, \_\_VA\_ARGS\_\_)
- #define CFE\_EVS\_SendErr(E, ...) CFE\_EVS\_Send(E, ERROR, \_\_VA\_ARGS\_\_)
- #define CFE\_EVS\_SendCrit(E, ...) CFE\_EVS\_Send(E, CRITICAL, \_\_VA\_ARGS\_\_)

#### Functions

- CFE\_Status\_t CFE\_EVS\_Register (const void \*Filters, uint16 NumEventFilters, uint16 FilterScheme)
 

*Register an application for receiving event services.*
- CFE\_Status\_t CFE\_EVS\_SendEvent (uint16 EventID, CFE\_EVS\_EventType\_Enum\_t EventType, const char \*Spec,...) OS\_PRINTF(3)
 

*Generate a software event.*
- CFE\_Status\_t CFE\_Status\_t CFE\_EVS\_SendEventWithAppID (uint16 EventID, CFE\_EVS\_EventType\_Enum\_t EventType, CFE\_ES\_AppId\_t AppID, const char \*Spec,...) OS\_PRINTF(4)
 

*Generate a software event given the specified Application ID.*
- CFE\_Status\_t CFE\_Status\_t CFE\_Status\_t CFE\_EVS\_SendTimedEvent (CFE\_TIME\_SysTime\_t Time, uint16 EventID, CFE\_EVS\_EventType\_Enum\_t EventType, const char \*Spec,...) OS\_PRINTF(4)
 

*Generate a software event with a specific time tag.*
- CFE\_Status\_t CFE\_EVS\_ResetFilter (uint16 EventID)

*Resets the calling application's event filter for a single event ID.*

- `CFE_Status_t CFE_EVS_ResetAllFilters (void)`

*Resets all of the calling application's event filters.*

#### 12.104.1 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

#### 12.104.2 Macro Definition Documentation

##### 12.104.2.1 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.

##### 12.104.2.2 CFE\_EVS\_SendCrit #define CFE\_EVS\_SendCrit(

```
E,  
... ) CFE_EVS_Send(E, CRITICAL, __VA_ARGS__)
```

Definition at line 50 of file cfe\_evs.h.

##### 12.104.2.3 CFE\_EVS\_SendDbg #define CFE\_EVS\_SendDbg(

```
E,  
... ) CFE_EVS_Send(E, DEBUG, __VA_ARGS__)
```

Definition at line 47 of file cfe\_evs.h.

##### 12.104.2.4 CFE\_EVS\_SendErr #define CFE\_EVS\_SendErr(

```
E,  
... ) CFE_EVS_Send(E, ERROR, __VA_ARGS__)
```

Definition at line 49 of file cfe\_evs.h.

##### 12.104.2.5 CFE\_EVS\_SendInfo #define CFE\_EVS\_SendInfo(

```
E,  
... ) CFE_EVS_Send(E, INFORMATION, __VA_ARGS__)
```

Definition at line 48 of file cfe\_evs.h.

#### 12.105 cfe/modules/core\_api/fsw/inc/cfe\_evs\_api\_typedefs.h File Reference

```
#include "common_types.h"  
#include "cfe_evs_extern_typedefs.h"
```

## Data Structures

- struct [CFE\\_EVS\\_BinFilter](#)

*Event message filter definition structure.*

## Macros

### Common Event Filter Mask Values

*Message is sent if (previous event count) & MASK == 0*

- #define [CFE\\_EVS\\_NO\\_FILTER](#) 0x0000  
*Stops any filtering. All messages are sent.*
- #define [CFE\\_EVS\\_FIRST\\_ONE\\_STOP](#) 0xFFFF  
*Sends the first event. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_TWO\\_STOP](#) 0xFFFE  
*Sends the first 2 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_4\\_STOP](#) 0xFFFC  
*Sends the first 4 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_8\\_STOP](#) 0xFFF8  
*Sends the first 8 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_16\\_STOP](#) 0FFF0  
*Sends the first 16 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_32\\_STOP](#) 0FFE0  
*Sends the first 32 events. All remaining messages are filtered.*
- #define [CFE\\_EVS\\_FIRST\\_64\\_STOP](#) 0FFC0  
*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.*

### 12.105.1 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

### 12.105.2 Macro Definition Documentation

#### 12.105.2.1 [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`.

**12.105.2.2 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.

**12.105.2.3 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.

**12.105.2.4 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.

**12.105.2.5 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.

**12.105.2.6 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.

**12.105.2.7 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.

**12.105.2.8 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.

**12.105.2.9 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.

**12.105.2.10 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.

**12.105.2.11 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.

## 12.105.3 Typedef Documentation

**12.105.3.1 CFE\_EVS\_BinFilter\_t** `typedef struct CFE_EVS_BinFilter CFE_EVS_BinFilter_t`  
Event message filter definition structure.

## 12.106 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.*

### 12.106.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

## 12.107 cfe/modules/core\_api/fsw/inc/cfe\_fs\_api\_typedefs.h File Reference

```
#include "common_types.h"
#include "osconfig.h"
#include "cfe_mission_cfg.h"
#include "cfe_fs_extern_typedefs.h"
```

## Data Structures

- struct [CFE\\_FS\\_FileWriteMetaData](#)

*External Metadata/State object associated with background file writes.*

## Typedefs

- typedef bool(\* [CFE\\_FS\\_FileWriteGetData\\_t](#)) (void \*Meta, [uint32](#) RecordNum, void \*\*Buffer, size\_t \*BufSize)
- typedef void(\* [CFE\\_FS\\_FileWriteOnEvent\\_t](#)) (void \*Meta, [CFE\\_FS\\_FileWriteEvent\\_t](#) Event, [int32](#) Status, [uint32](#) RecordNum, size\_t BlockSize, size\_t Position)
- typedef struct [CFE\\_FS\\_FileWriteMetaData](#) [CFE\\_FS\\_FileWriteMetaData\\_t](#)

*External Metadata/State object associated with background file writes.*

## Enumerations

- enum [CFE\\_FS\\_FileCategory\\_t](#) {
 [CFE\\_FS\\_FileCategory\\_UNKNOWN](#) , [CFE\\_FS\\_FileCategory\\_DYNAMIC\\_MODULE](#) , [CFE\\_FS\\_FileCategory\\_BINARY\\_DATA\\_DUMP](#) ,
 [CFE\\_FS\\_FileCategory\\_TEXT\\_LOG](#) ,
 [CFE\\_FS\\_FileCategory\\_SCRIPT](#) , [CFE\\_FS\\_FileCategory\\_TEMP](#) , [CFE\\_FS\\_FileCategory\\_MAX](#) }
- Generalized file types/categories known to FS.*
- enum [CFE\\_FS\\_FileWriteEvent\\_t](#) {
 [CFE\\_FS\\_FileWriteEvent\\_UNDEFINED](#) , [CFE\\_FS\\_FileWriteEvent\\_COMPLETE](#) , [CFE\\_FS\\_FileWriteEvent\\_CREATE\\_ERROR](#) ,
 [CFE\\_FS\\_FileWriteEvent\\_HEADER\\_WRITE\\_ERROR](#) ,
 [CFE\\_FS\\_FileWriteEvent\\_RECORD\\_WRITE\\_ERROR](#) , [CFE\\_FS\\_FileWriteEvent\\_MAX](#) }

### 12.107.1 Detailed Description

Purpose: cFE File Services (FS) library API header file

Author: S.Walling/Microtel

### 12.107.2 Typedef Documentation

**12.107.2.1 CFE\_FS\_FileWriteGetData\_t** [typedef bool\(\\* CFE\\_FS\\_FileWriteGetData\\_t\)](#) ([void \\*Meta](#), [uint32](#) RecordNum, [void \\*\\*Buffer](#), [size\\_t \\*BufSize](#))  
 Data Getter routine provided by requester  
 Outputs a data block. Should return true if the file is complete (last record/EOF), otherwise return false.

#### Parameters

<i>in, out</i>	<i>Meta</i>	Pointer to the metadata object
<i>in</i>	<i>RecordNum</i>	Incrementing record number counter
<i>out</i>	<i>Buffer</i>	Pointer to buffer data block, should be set by implementation
<i>out</i>	<i>BufSize</i>	Pointer to buffer data size, should be set by implementation

#### Returns

End of file status

#### Return values

<i>true</i>	if at last data record, and output file should be closed
<i>false</i>	if not at last record, more data records to write

**Note**

The implementation of this function must always set the "Buffer" and "BufSize" outputs. If no data is available, they may be set to NULL and 0, respectively.

Definition at line 98 of file cfe\_fs\_api\_typedefs.h.

### **12.107.2.2 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).

### **12.107.2.3 CFE\_FS\_FileWriteOnEvent\_t** `typedef void(* CFE_FS_FileWriteOnEvent_t) (void *Meta, CFE_FS_FileWriteEvent`

`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**

<i>in,out</i>	<i>Meta</i>	Pointer to the metadata object
<i>in</i>	<i>Event</i>	Generalized type of event to report (not actual event ID)
<i>in</i>	<i>Status</i>	Generalized status code (may be from OSAL or CFE)
<i>in</i>	<i>RecordNum</i>	Record number counter at which event occurred
<i>in</i>	<i>BlockSize</i>	Size of record being processed when event occurred (if applicable)
<i>in</i>	<i>Position</i>	File position/size when event occurred

Definition at line 114 of file cfe\_fs\_api\_typedefs.h.

**12.107.3 Enumeration Type Documentation**

### **12.107.3.1 CFE\_FS\_FileCategory\_t** `enum CFE_FS_FileCategory_t`

Generalized file types/categories known to FS.

This defines different categories of files, where they may reside in different default locations of the virtualized file system. This is different from, and should not be confused with, the "SubType" field in the FS header. This value is only used at runtime for FS APIs and should not actually appear in any output file or message.

**Enumerator**

<code>CFE_FS_FileCategory_UNKNOWN</code>	Placeholder, unknown file category
<code>CFE_FS_FileCategory_DYNAMIC_MODULE</code>	Dynamically loadable apps/libraries (e.g. .so, .o, .dll, etc)
<code>CFE_FS_FileCategory_BINARY_DATA_DUMP</code>	Binary log file generated by various data dump commands
<code>CFE_FS_FileCategory_TEXT_LOG</code>	Text-based log file generated by various commands
<code>CFE_FS_FileCategory_SCRIPT</code>	Text-based Script files (e.g. ES startup script)
<code>CFE_FS_FileCategory_TEMP</code>	Temporary/Ephemeral files
<code>CFE_FS_FileCategory_MAX</code>	Placeholder, keep last

Definition at line 49 of file cfe\_fs\_api\_typedefs.h.

### 12.107.3.2 CFE\_FS\_FileWriteEvent\_t enum CFE\_FS\_FileWriteEvent\_t

Enumerator

CFE_FS_FileWriteEvent_UNDEFINED	
CFE_FS_FileWriteEvent_COMPLETE	File is completed successfully
CFE_FS_FileWriteEvent_CREATE_ERROR	Unable to create/open file
CFE_FS_FileWriteEvent_HEADER_WRITE_ERROR	Unable to write FS header
CFE_FS_FileWriteEvent_RECORD_WRITE_ERROR	Unable to write data record
CFE_FS_FileWriteEvent_MAX	

Definition at line 69 of file cfe\_fs\_api\_typedefs.h.

## 12.108 cfe/modules/core\_api/fsw/inc/cfe\_msg.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_msg_hdr.h"
#include "cfe_msg_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
#include "cfe_sb_api_typedefs.h"
#include "cfe_time_api_typedefs.h"
```

### Functions

- **CFE\_Status\_t CFE\_MSG\_Init (CFE\_MSG\_Message\_t \*MsgPtr, CFE\_SB\_MsgId\_t MsgId, CFE\_MSG\_Size\_t Size)**

*Initialize a message.*
- **CFE\_Status\_t CFE\_MSG\_GetSize (const CFE\_MSG\_Message\_t \*MsgPtr, CFE\_MSG\_Size\_t \*Size)**

*Gets the total size of a message.*
- **CFE\_Status\_t CFE\_MSG\_SetSize (CFE\_MSG\_Message\_t \*MsgPtr, CFE\_MSG\_Size\_t Size)**

*Sets the total size of a message.*
- **CFE\_Status\_t CFE\_MSG.GetType (const CFE\_MSG\_Message\_t \*MsgPtr, CFE\_MSG\_Type\_t \*Type)**

*Gets the message type.*
- **CFE\_Status\_t CFE\_MSG\_SetType (CFE\_MSG\_Message\_t \*MsgPtr, CFE\_MSG\_Type\_t Type)**

*Sets the message type.*
- **CFE\_Status\_t CFE\_MSG\_GetHeaderVersion (const CFE\_MSG\_Message\_t \*MsgPtr, CFE\_MSG\_HeaderVersion\_t \*Version)**

*Gets the message header version.*
- **CFE\_Status\_t CFE\_MSG\_SetHeaderVersion (CFE\_MSG\_Message\_t \*MsgPtr, CFE\_MSG\_HeaderVersion\_t Version)**

*Sets the message header version.*
- **CFE\_Status\_t CFE\_MSG\_GetHasSecondaryHeader (const CFE\_MSG\_Message\_t \*MsgPtr, bool \*HasSecondary)**

*Gets the message secondary header boolean.*
- **CFE\_Status\_t CFE\_MSG\_SetHasSecondaryHeader (CFE\_MSG\_Message\_t \*MsgPtr, bool HasSecondary)**

*Sets the message secondary header boolean.*
- **CFE\_Status\_t CFE\_MSG\_GetApld (const CFE\_MSG\_Message\_t \*MsgPtr, CFE\_MSG\_Apld\_t \*Apld)**

*Gets the message application ID.*

- `CFE_Status_t CFE_MSG_SetApld (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Apld_t Apld)`  
*Sets the message application ID.*
- `CFE_Status_t CFE_MSG_GetSegmentationFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t *SegFlag)`  
*Gets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_SetSegmentationFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SegmentationFlag_t SegFlag)`  
*Sets the message segmentation flag.*
- `CFE_Status_t CFE_MSG_GetSequenceCount (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t *SeqCnt)`  
*Gets the message sequence count.*
- `CFE_Status_t CFE_MSG_SetSequenceCount (CFE_MSG_Message_t *MsgPtr, CFE_MSG_SequenceCount_t SeqCnt)`  
*Sets the message sequence count.*
- `CFE_MSG_SequenceCount_t CFE_MSG_GetNextSequenceCount (CFE_MSG_SequenceCount_t SeqCnt)`  
*Gets the next sequence count value (rolls over if appropriate)*
- `CFE_Status_t CFE_MSG_GetEDSVersion (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t *Version)`  
*Gets the message EDS version.*
- `CFE_Status_t CFE_MSG_SetEDSVersion (CFE_MSG_Message_t *MsgPtr, CFE_MSG_EDSVersion_t Version)`  
*Sets the message EDS version.*
- `CFE_Status_t CFE_MSG_GetEndian (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t *Endian)`  
*Gets the message endian.*
- `CFE_Status_t CFE_MSG_SetEndian (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Endian_t Endian)`  
*Sets the message endian.*
- `CFE_Status_t CFE_MSG_GetPlaybackFlag (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t *PlayFlag)`  
*Gets the message playback flag.*
- `CFE_Status_t CFE_MSG_SetPlaybackFlag (CFE_MSG_Message_t *MsgPtr, CFE_MSG_PlaybackFlag_t PlayFlag)`  
*Sets the message playback flag.*
- `CFE_Status_t CFE_MSG_GetSubsystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t *Subsystem)`  
*Gets the message subsystem.*
- `CFE_Status_t CFE_MSG_SetSubsystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_Subsystem_t Subsystem)`  
*Sets the message subsystem.*
- `CFE_Status_t CFE_MSG_GetSystem (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t *System)`  
*Gets the message system.*
- `CFE_Status_t CFE_MSG_SetSystem (CFE_MSG_Message_t *MsgPtr, CFE_MSG_System_t System)`  
*Sets the message system.*
- `CFE_Status_t CFE_MSG_GenerateChecksum (CFE_MSG_Message_t *MsgPtr)`  
*Calculates and sets the checksum of a message.*
- `CFE_Status_t CFE_MSG_ValidateChecksum (const CFE_MSG_Message_t *MsgPtr, bool *isValid)`  
*Validates the checksum of a message.*
- `CFE_Status_t CFE_MSG_SetFcnCode (CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t FcnCode)`  
*Sets the function code field in a message.*
- `CFE_Status_t CFE_MSG_GetFcnCode (const CFE_MSG_Message_t *MsgPtr, CFE_MSG_FcnCode_t *FcnCode)`  
*Code)*

- Gets the function code field from a message.
- CFE\_Status\_t CFE\_MSG\_GetMsgTime (const CFE\_MSG\_Message\_t \*MsgPtr, CFE\_TIME\_SysTime\_t \*Time)
  - Gets the time field from a message.
- CFE\_Status\_t CFE\_MSG\_SetMsgTime (CFE\_MSG\_Message\_t \*MsgPtr, CFE\_TIME\_SysTime\_t NewTime)
  - Sets the time field in a message.
- CFE\_Status\_t CFE\_MSG\_GetMsgId (const CFE\_MSG\_Message\_t \*MsgPtr, CFE\_SB\_MsgId\_t \*MsgId)
  - Gets the message id from a message.
- CFE\_Status\_t CFE\_MSG\_SetMsgId (CFE\_MSG\_Message\_t \*MsgPtr, CFE\_SB\_MsgId\_t MsgId)
  - Sets the message id bits in a message.
- CFE\_Status\_t CFE\_MSG\_GetTypeFromMsgId (CFE\_SB\_MsgId\_t MsgId, CFE\_MSG\_Type\_t \*Type)
  - Gets message type using message ID.
- CFE\_Status\_t CFE\_MSG\_OriginationAction (CFE\_MSG\_Message\_t \*MsgPtr, size\_t BufferSize, bool \*IsAcceptable)
  - Perform any necessary actions on a newly-created message, prior to sending.
- CFE\_Status\_t CFE\_MSG\_VerificationAction (const CFE\_MSG\_Message\_t \*MsgPtr, size\_t BufferSize, bool \*IsAcceptable)
  - Checks message integrity/acceptability.

### 12.108.1 Detailed Description

Message access APIs

## 12.109 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 struct CFE_MSG_Message CFE_MSG_Message_t`  
*cFS generic base message*
- `typedef struct CFE_MSG_CommandHeader CFE_MSG_CommandHeader_t`  
*cFS command header*
- `typedef struct CFE_MSG_TelemetryHeader CFE_MSG_TelemetryHeader_t`  
*cFS telemetry header*

## Enumerations

- `enum CFE_MSG_Type { CFE_MSG_Type_Invalid , CFE_MSG_Type_Cmd , CFE_MSG_Type_Tlm }`  
*Message type.*
- `enum CFE_MSG_SegmentationFlag { CFE_MSG_SegFlag_Invalid , CFE_MSG_SegFlag_Continue , CFE_MSG_SegFlag_First , CFE_MSG_SegFlag_Last , CFE_MSG_SegFlag_Unsegmented }`  
*Segmentation flags.*
- `enum CFE_MSG_Endian { CFE_MSG_Endian_Invalid , CFE_MSG_Endian_Big , CFE_MSG_Endian_Little }`  
*Endian flag.*
- `enum CFE_MSG_PlaybackFlag { CFE_MSG_PlayFlag_Invalid , CFE_MSG_PlayFlag_Original , CFE_MSG_PlayFlag_Playback }`  
*Playback flag.*

### 12.109.1 Detailed Description

Typedefs for Message API

- Separate from API so these can be adjusted for custom implementations

### 12.109.2 Macro Definition Documentation

#### 12.109.2.1 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.

**12.109.2.2 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.

**12.109.2.3 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.

### 12.109.3 Typedef Documentation

**12.109.3.1 CFE\_MSG\_Apld\_t** typedef uint16 CFE\_MSG\_Apld\_t  
Message application ID.  
Definition at line 50 of file cfe\_msg\_api\_typedefs.h.

**12.109.3.2 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.

**12.109.3.3 CFE\_MSG\_CommandHeader\_t** typedef struct CFE\_MSG\_CommandHeader CFE\_MSG\_CommandHeader\_t  
cFS command header  
Definition at line 54 of file cfe\_msg\_api\_typedefs.h.

**12.109.3.4 CFE\_MSG\_EDSVersion\_t** typedef uint16 CFE\_MSG\_EDSVersion\_t  
Message EDS version.  
Definition at line 52 of file cfe\_msg\_api\_typedefs.h.

**12.109.3.5 CFE\_MSG\_Endian\_t** typedef enum CFE\_MSG\_Endian CFE\_MSG\_Endian\_t  
Endian flag.

**12.109.3.6 CFE\_MSG\_FcnCode\_t** typedef uint16 CFE\_MSG\_FcnCode\_t  
Message function code.  
Definition at line 48 of file cfe\_msg\_api\_typedefs.h.

**12.109.3.7 CFE\_MSG\_HeaderVersion\_t** typedef uint16 CFE\_MSG\_HeaderVersion\_t  
Message header version.  
Definition at line 49 of file cfe\_msg\_api\_typedefs.h.

**12.109.3.8 CFE\_MSG\_Message\_t** typedef struct CFE\_MSG\_Message CFE\_MSG\_Message\_t  
cFS generic base message  
Definition at line 54 of file cfe\_msg\_api\_typedefs.h.

**12.109.3.9 CFE\_MSG\_PlaybackFlag\_t** `typedef enum CFE_MSG_PlaybackFlag CFE_MSG_PlaybackFlag_t`  
Playback flag.

**12.109.3.10 CFE\_MSG\_SegmentationFlag\_t** `typedef enum CFE_MSG_SegmentationFlag CFE_MSG_SegmentationFlag_t`  
Segmentation flags.

**12.109.3.11 CFE\_MSG\_SequenceCount\_t** `typedef uint16 CFE_MSG_SequenceCount_t`  
Message sequence count.  
Definition at line 51 of file `cfe_msg_api_typedefs.h`.

**12.109.3.12 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`.

**12.109.3.13 CFE\_MSG\_Subsystem\_t** `typedef uint16 CFE_MSG_Subsystem_t`  
Message subsystem.  
Definition at line 53 of file `cfe_msg_api_typedefs.h`.

**12.109.3.14 CFE\_MSG\_System\_t** `typedef uint16 CFE_MSG_System_t`  
Message system.  
Definition at line 54 of file `cfe_msg_api_typedefs.h`.

**12.109.3.15 CFE\_MSG\_TelemetryHeader\_t** `typedef struct CFE_MSG_TelemetryHeader CFE_MSG_TelemetryHeader_t`  
cFS telemetry header  
Definition at line 54 of file `cfe_msg_api_typedefs.h`.

**12.109.3.16 CFE\_MSG\_Type\_t** `typedef enum CFE_MSG_Type CFE_MSG_Type_t`  
Message type.

## 12.109.4 Enumeration Type Documentation

**12.109.4.1 CFE\_MSG\_Endian** `enum CFE_MSG_Endian`  
Endian flag.

Enumerator

<code>CFE_MSG_Endian_Invalid</code>	Invalid endian setting.
<code>CFE_MSG_Endian_Big</code>	Big endian.
<code>CFE_MSG_Endian_Little</code>	Little endian.

Definition at line 75 of file `cfe_msg_api_typedefs.h`.

#### 12.109.4.2 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`.

#### 12.109.4.3 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`.

#### 12.109.4.4 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`.

## 12.110 cfe/modules/core\_api/fsw/inc/cfe\_resourceid.h File Reference

```
#include "cfe_resourceid_api_typedefs.h"
```

### Typedefs

- `typedef CFE_ResourceId_t(* CFE_ResourceId_IncrementFunc_t)` (`CFE_ResourceId_t, void *`)  
*Serial number increment function.*
- `typedef bool(* CFE_ResourceId_CheckFunc_t)` (`CFE_ResourceId_t`)  
*Serial number availability check function.*

## Functions

- `uint32 CFE_Resourceld_GetBase (CFE_Resourceld_t Resourceld)`  
*Get the Base value (type/category) from a resource ID value.*
- `uint32 CFE_Resourceld_GetSerial (CFE_Resourceld_t Resourceld)`  
*Get the Serial Number (sequential ID) from a resource ID value.*
- `CFE_Resourceld_t CFE_Resourceld_FindNext (CFE_Resourceld_t StartId, uint32 TableSize, CFE_Resourceld_CheckFunc_t CheckFunc)`  
*Locate the next resource ID that maps to an available table entry.*
- `CFE_Resourceld_t CFE_Resourceld_FindNextEx (CFE_Resourceld_t StartId, CFE_Resourceld_IncrementFunc_t IncrFunc, void *IncrArg, CFE_Resourceld_CheckFunc_t CheckFunc)`  
*Locate the next resource ID that maps to an available table entry.*
- `int32 CFE_Resourceld_ToIndex (CFE_Resourceld_t Id, uint32 BaseValue, uint32 TableSize, uint32 *Idx)`  
*Internal routine to aid in converting an ES resource ID to an array index.*

## Resource ID test/conversion macros and inline functions

- `#define CFE_RESOURCEID_TO ULONG(id) CFE_Resourceld_ToInteger(CFE_RESOURCEID_UNWRAP(id))`  
*Convert a derived (app-specific) ID directly into an "unsigned long".*
- `#define CFE_RESOURCEID_TEST_DEFINED(id) CFE_Resourceld_IsDefined(CFE_RESOURCEID_UNWRAP(id))`  
*Determine if a derived (app-specific) ID is defined or not.*
- `#define CFE_RESOURCEID_TEST_EQUAL(id1, id2) CFE_Resourceld_Equal(CFE_RESOURCEID_UNWRAP(id1), CFE_RESOURCEID_UNWRAP(id2))`  
*Determine if two derived (app-specific) IDs are equal.*
- `static unsigned long CFE_Resourceld_ToInteger (CFE_Resourceld_t id)`  
*Convert a resource ID to an integer.*
- `static CFE_Resourceld_t CFE_Resourceld_FromInteger (unsigned long Value)`  
*Convert an integer to a resource ID.*
- `static bool CFE_Resourceld_Equal (CFE_Resourceld_t id1, CFE_Resourceld_t id2)`  
*Compare two Resource ID values for equality.*
- `static bool CFE_Resourceld_IsDefined (CFE_Resourceld_t id)`  
*Check if a resource ID value is defined.*

### 12.110.1 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)

### 12.110.2 Macro Definition Documentation

```
12.110.2.1 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 97 of file cfe\_resourceid.h.

```
12.110.2.2 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 105 of file cfe\_resourceid.h.

```
12.110.2.3 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 89 of file cfe\_resourceid.h.

### 12.110.3 Typedef Documentation

```
12.110.3.1 CFE_Resourceld_CheckFunc_t typedef bool(* CFE_ResourceId_CheckFunc_t)(CFE_ResourceId_t)
```

Serial number availability check function.

Checks if the slot associated with a pending serial number is in use or not. Used with [CFE\\_Resourceld\\_FindNext\(\)](#) to find the next available serial number.

#### Return values

<i>true</i>	if the slot is already in use (unavailable)
<i>false</i>	if the slot is not in use (available)

Definition at line 70 of file cfe\_resourceid.h.

```
12.110.3.2 CFE_Resourceld_IncrementFunc_t typedef CFE_ResourceId_t(* CFE_ResourceId_IncrementFunc_t)(CFE_ResourceId_t, void *)
```

Serial number increment function.

A helper function responsible for incrementing the serial number when iterating over all available resource slots. The default implementation of this function will treat all slots as equal, and simply increment to the next serial number. An alternative function can be used with [CFE\\_Resourceld\\_FindNextEx\(\)](#) if there are special requirements for slot assignments/relationships.

#### Returns

Next serial number to check/test

#### Return values

<code>CFE_RESOURCEID_UNDEFINED</code>	if no more IDs are available to test
---------------------------------------	--------------------------------------

Definition at line 58 of file cfe\_resourceid.h.

### 12.110.4 Function Documentation

**12.110.4.1 CFE\_Resourceld\_Equal()** `static bool CFE_ResourceId_Equal (`  
 `CFE_ResourceId_t id1,`  
 `CFE_ResourceId_t id2 ) [inline], [static]`

Compare two Resource ID values for equality.

#### Parameters

in	<i>id1</i>	Resource ID to check
in	<i>id2</i>	Resource ID to check

#### Returns

true if id1 and id2 are equal, false otherwise.

Definition at line 160 of file cfe\_resourceid.h.

Referenced by [CFE\\_Resourceld\\_IsDefined\(\)](#).

**12.110.4.2 CFE\_Resourceld\_FindNext()** `CFE_ResourceId_t CFE_ResourceId_FindNext (`  
 `CFE_ResourceId_t StartId,`  
 `uint32 TableSize,`  
 `CFE_ResourceId_CheckFunc_t CheckFunc )`

Locate the next resource ID that maps to an available 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 that maps to a table entry that is available for use. That is, it does not alias any valid/in-use ID when converted to an array index.

This is the simple form of the API that iterates over all slots equally in a round-robin fashion, and works for most use cases.

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**

<code>CFE_RESOURCEID_UNDEFINED</code>	if no open slots or bad arguments.
---------------------------------------	------------------------------------

**12.110.4.3 CFE\_ResourceId\_FindNextEx()** `CFE_ResourceId_t CFE_ResourceId_FindNextEx (`

```
    CFE_ResourceId_t StartId,
    CFE_ResourceId_IncrementFunc_t IncrFunc,
    void * IncrArg,
    CFE_ResourceId_CheckFunc_t CheckFunc )
```

Locate the next resource ID that maps to an available table entry.

An extended form of [CFE\\_ResourceId\\_FindNext\(\)](#) that allows more control over the slots that are checked. This can be used if slots are not all equivalent and thus the simple round-robin approach is insufficient. The additional increment function should return the next ID to test/check, given the previous ID.

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>IncrFunc</i>	function to increment the id to the next value
in	<i>IncrArg</i>	opaque argument that is passed through to the increment function
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**

<code>CFE_RESOURCEID_UNDEFINED</code>	if no open slots or bad arguments.
---------------------------------------	------------------------------------

**12.110.4.4 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 148 of file cfe\_resourceid.h.

**12.110.4.5 CFE\_ResourceId\_GetBase()** `uint32 CFE_ResourceId_GetBase ( CFE_ResourceId_t ResourceId )`

Get the Base value (type/category) from a resource ID value.

This masks out the ID serial number to obtain the base value, which is different for each resource type.

**Note**

The value is NOT shifted or otherwise adjusted.

**Parameters**

in	<i>Resource← Id</i>	the resource ID to decode
----	-------------------------	---------------------------

**Returns**

The base value associated with that ID

**12.110.4.6 CFE\_ResourceId\_GetSerial()** `uint32 CFE_ResourceId_GetSerial ( CFE_ResourceId_t ResourceId )`

Get the Serial Number (sequential ID) from a resource ID value.

This masks out the ID base value to obtain the serial number, which is different for each entity created.

**Parameters**

in	<i>Resource← Id</i>	the resource ID to decode
----	-------------------------	---------------------------

**Returns**

The serial number associated with that ID

**12.110.4.7 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
----	-----------	----------------------

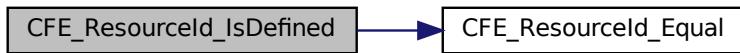
**Returns**

True if the ID may refer to a defined entity, false if invalid/undefined.

Definition at line 178 of file cfe\_resourceid.h.

References CFE\_ResourceId\_Equal(), and CFE\_RESOURCEID\_UNDEFINED.

Here is the call graph for this function:

**12.110.4.8 CFE\_ResourceId\_ToIndex()** `int32 CFE_ResourceId_ToIndex (`

```

    CFE_ResourceId_t Id,
    uint32 BaseValue,
    uint32 TableSize,
    uint32 * Idx )
```

Internal routine to aid in converting an ES resource ID to an array index.

**Parameters**

in	<i>Id</i>	The resource ID
in	<i>BaseValue</i>	The respective ID base value corresponding to the ID type
in	<i>TableSize</i>	The actual size of the internal table (MAX index value + 1)
out	<i>Idx</i>	The output index

**Returns**

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

**Return values**

<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.

**12.110.4.9 CFE\_ResourceId\_ToInteger()** `static unsigned long CFE_ResourceId_ToInteger (`

```

    CFE_ResourceId_t id ) [inline], [static]
```

Convert a resource ID to an integer.

This is primarily intended for logging purposes, such as writing to debug console, event messages, or log files, using printf-like APIs.

For compatibility with C library APIs, this returns an "unsigned long" type and should be used with the "%lx" format specifier in a printf format string.

#### Note

No assumptions should be made about the actual integer value, such as its base/range. It may be printed, but should not be modified or tested/compared using other arithmetic ops, and should never be used as the index to an array or table. See the related function [CFE\\_Resourceld\\_ToIndex\(\)](#) for cases where a zero-based array/table index is needed.

#### See also

[CFE\\_Resourceld\\_FromInteger\(\)](#)

#### Parameters

in	<i>id</i>	Resource ID to convert
----	-----------	------------------------

#### Returns

Integer value corresponding to ID

Definition at line 129 of file cfe\_resourceid.h.

## 12.111 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\_Resourceld\_t)CFE\_RESOURCEID\_WRAP(0))  
*A resource ID value that represents an undefined/unused resource.*
- #define [CFE\\_RESOURCEID\\_RESERVED](#) ((CFE\_Resourceld\_t)CFE\_RESOURCEID\_WRAP(0xFFFFFFFF))  
*A resource ID value that represents a reserved entry.*

### 12.111.1 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)

## 12.111.2 Macro Definition Documentation

**12.111.2.1 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.

**12.111.2.2 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.

## 12.112 cfe/modules/core\_api/fsw/inc/cfe\_sb.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_sb_api_typedefs.h"
#include "cfe_es_api_typedefs.h"
```

### Macros

- #define **CFE\_BIT**(x) (1 << (x))
   
*Places a one at bit positions 0 - 31.*
- #define **CFE\_SET**(i, x) ((i) |= **CFE\_BIT**(x))
   
*Sets bit x of i.*
- #define **CFE\_CLR**(i, x) ((i) &= ~**CFE\_BIT**(x))
   
*Clears bit x of i.*
- #define **CFE\_TST**(i, x) (((i)&**CFE\_BIT**(x)) != 0)
   
*true(non zero) if bit x of i is set*

### Functions

- **CFE\_Status\_t CFE\_SB\_CreatePipe** (CFE\_SB\_Pipeld\_t \*PipeldPtr, uint16 Depth, const char \*PipeName)
   
*Creates a new software bus pipe.*
- **CFE\_Status\_t CFE\_SB\_DeletePipe** (CFE\_SB\_Pipeld\_t Pipeld)
   
*Delete a software bus pipe.*
- **CFE\_Status\_t CFE\_SB\_Pipeld\_ToIndex** (CFE\_SB\_Pipeld\_t PipeID, uint32 \*Idx)
   
*Obtain an index value correlating to an SB Pipe ID.*
- **CFE\_Status\_t CFE\_SB\_SetPipeOpts** (CFE\_SB\_Pipeld\_t Pipeld, uint8 Opts)
   
*Set options on a pipe.*
- **CFE\_Status\_t CFE\_SB\_GetPipeOpts** (CFE\_SB\_Pipeld\_t Pipeld, uint8 \*OptsPtr)
   
*Get options on a pipe.*

- `CFE_Status_t CFE_SB_GetPipeName (char *PipeNameBuf, size_t PipeNameSize, CFE_SB_Pipeld_t Pipeld)`  
*Get the pipe name for a given id.*
- `CFE_Status_t CFE_SB_GetPipeldByName (CFE_SB_Pipeld_t *PipeldPtr, const char *PipeName)`  
*Get pipe id by pipe name.*
- `CFE_Status_t CFE_SB_SubscribeEx (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld, CFE_SB_Qos_t Quality, uint16 MsgLim)`  
*Subscribe to a message on the software bus.*
- `CFE_Status_t CFE_SB_Subscribe (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`  
*Subscribe to a message on the software bus with default parameters.*
- `CFE_Status_t CFE_SB_SubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld, uint16 MsgLim)`  
*Subscribe to a message while keeping the request local to a cpu.*
- `CFE_Status_t CFE_SB_Unsubscribe (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`  
*Remove a subscription to a message on the software bus.*
- `CFE_Status_t CFE_SB_UnsubscribeLocal (CFE_SB_MsgId_t MsgId, CFE_SB_Pipeld_t Pipeld)`  
*Remove a subscription to a message on the software bus on the current CPU.*
- `CFE_Status_t CFE_SB_TransmitMsg (const CFE_MSG_Message_t *MsgPtr, bool IsOrigination)`  
*Transmit a message.*
- `CFE_Status_t CFE_SB_ReceiveBuffer (CFE_SB_Buffer_t **BufPtr, CFE_SB_Pipeld_t Pipeld, int32 TimeOut)`  
*Receive a message from a software bus pipe.*
- `CFE_SB_Buffer_t * CFE_SB_AllocateMessageBuffer (size_t MsgSize)`  
*Get a buffer pointer to use for "zero copy" SB sends.*
- `CFE_Status_t CFE_SB_ReleaseMessageBuffer (CFE_SB_Buffer_t *BufPtr)`  
*Release an unused "zero copy" buffer pointer.*
- `CFE_Status_t CFE_SB_TransmitBuffer (CFE_SB_Buffer_t *BufPtr, bool IsOrigination)`  
*Transmit a buffer.*
- `void CFE_SB_SetUserDataLength (CFE_MSG_Message_t *MsgPtr, size_t DataLength)`  
*Sets the length of user data in a software bus message.*
- `void CFE_SB_TimeStampMsg (CFE_MSG_Message_t *MsgPtr)`  
*Sets the time field in a software bus message with the current spacecraft time.*
- `int32 CFE_SB_MessageStringSet (char *DestStringPtr, const char *SourceStringPtr, size_t DestMaxSize, size_t SourceMaxSize)`  
*Copies a string into a software bus message.*
- `void * CFE_SB_GetUserData (CFE_MSG_Message_t *MsgPtr)`  
*Get a pointer to the user data portion of a software bus message.*
- `size_t CFE_SB_GetUserDataLength (const CFE_MSG_Message_t *MsgPtr)`  
*Gets the length of user data in a software bus message.*
- `int32 CFE_SB_MessageStringGet (char *DestStringPtr, const char *SourceStringPtr, const char *DefaultString, size_t DestMaxSize, size_t SourceMaxSize)`  
*Copies a string out of a software bus message.*
- `bool CFE_SB_IsValidMsgId (CFE_SB_MsgId_t MsgId)`  
*Identifies whether a given `CFE_SB_MsgId_t` is valid.*
- `static bool CFE_SB_MsgId_Equal (CFE_SB_MsgId_t MsgId1, CFE_SB_MsgId_t MsgId2)`  
*Identifies whether two `CFE_SB_MsgId_t` values are equal.*
- `static CFE_SB_MsgId_Atom_t CFE_SB_MsgIdToValue (CFE_SB_MsgId_t MsgId)`  
*Converts a `CFE_SB_MsgId_t` to a normal integer.*
- `static CFE_SB_MsgId_t CFE_SB_ValueToMsgId (CFE_SB_MsgId_Atom_t MsgIdValue)`  
*Converts a normal integer into a `CFE_SB_MsgId_t`.*

- **CFE\_SB\_MsgId\_Atom\_t CFE\_SB\_CmdTopicIdToMsgId** (`uint16 TopicId, uint16 InstanceNum`)  
*Converts a topic ID and instance number combination into a MsgID value integer.*
- **CFE\_SB\_MsgId\_Atom\_t CFE\_SB\_TlmTopicIdToMsgId** (`uint16 TopicId, uint16 InstanceNum`)  
*Converts a topic ID and instance number combination into a MsgID value integer.*
- **CFE\_SB\_MsgId\_Atom\_t CFE\_SB\_GlobalCmdTopicIdToMsgId** (`uint16 TopicId`)  
*Converts a topic ID to a MsgID value integer for Global commands.*
- **CFE\_SB\_MsgId\_Atom\_t CFE\_SB\_GlobalTlmTopicIdToMsgId** (`uint16 TopicId`)  
*Converts a topic ID to a MsgID value integer for Global telemetry.*
- **CFE\_SB\_MsgId\_Atom\_t CFE\_SB\_LocalCmdTopicIdToMsgId** (`uint16 TopicId`)  
*Converts a topic ID to a MsgID value integer for local commands.*
- **CFE\_SB\_MsgId\_Atom\_t CFE\_SB\_LocalTlmTopicIdToMsgId** (`uint16 TopicId`)  
*Converts a topic ID to a MsgID value integer for local telemetry.*

### 12.112.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.  
Author: R.McGraw/SSI

### 12.112.2 Macro Definition Documentation

**12.112.2.1 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.

**12.112.2.2 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.

**12.112.2.3 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.

**12.112.2.4 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.

## 12.113 cfe/modules/core\_api/fsw/inc/cfe\_sb\_api\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_sb_extern_typedefs.h"
#include "cfe_msg_api_typedefs.h"
#include "cfe_resourceid_api_typedefs.h"
#include "cfe_msg_hdr.h"
```

### Data Structures

- union [CFE\\_SB\\_Msg](#)  
*Software Bus generic message.*

### Macros

- #define [CFE\\_SB\\_POLL](#) 0  
*Option used with [CFE\\_SB\\_ReceiveBuffer](#) to request immediate pipe status.*
- #define [CFE\\_SB\\_PEND\\_FOREVER](#) -1  
*Option used with [CFE\\_SB\\_ReceiveBuffer](#) to force a wait for next message.*
- #define [CFE\\_SB\\_SUBSCRIPTION](#) 0  
*Subtype specifier used in [CFE\\_SB\\_SingleSubscriptionTlm\\_t](#) by SBN App.*
- #define [CFE\\_SB\\_UNSUBSCRIPTION](#) 1  
*Subtype specified used in [CFE\\_SB\\_SingleSubscriptionTlm\\_t](#) by SBN App.*
- #define [CFE\\_SB\\_MSGID\\_WRAP\\_VALUE](#)(val)  
*Translation macro to convert from MsgId integer values to opaque/abstract API values.*
- #define [CFE\\_SB\\_MSGID\\_C](#)(val) (([CFE\\_SB\\_MsgId\\_t](#))[CFE\\_SB\\_MSGID\\_WRAP\\_VALUE](#)(val))  
*Translation macro to convert to MsgId integer values from a literal.*
- #define [CFE\\_SB\\_MSGID\\_UNWRAP\\_VALUE](#)(mid) ((mid).Value)  
*Translation macro to convert to MsgId integer values from opaque/abstract API values.*
- #define [CFE\\_SB\\_MSGID\\_RESERVED](#) [CFE\\_SB\\_MSGID\\_WRAP\\_VALUE](#)(0)  
*Reserved value for [CFE\\_SB\\_MsgId\\_t](#) that will not match any valid MsgId.*
- #define [CFE\\_SB\\_INVALID\\_MSG\\_ID](#) [CFE\\_SB\\_MSGID\\_C](#)(0)  
*A literal of the [CFE\\_SB\\_MsgId\\_t](#) type representing an invalid ID.*
- #define [CFE\\_SB\\_PIPEID\\_C](#)(val) (([CFE\\_SB\\_Pipeld\\_t](#))[CFE\\_RESOURCEID\\_WRAP](#)(val))  
*Cast/Convert a generic [CFE\\_ResourceId\\_t](#) to a [CFE\\_SB\\_Pipeld\\_t](#).*
- #define [CFE\\_SB\\_INVALID\\_PIPE](#) [CFE\\_SB\\_PIPEID\\_C](#)([CFE\\_RESOURCEID\\_UNDEFINED](#))  
*A [CFE\\_SB\\_Pipeld\\_t](#) value which is always invalid.*
- #define [CFE\\_SB\\_PIPEOPTS\\_IGNOREMINE](#) 0x00000001  
*Messages sent by the app that owns this pipe will not be sent to this pipe.*
- #define [CFE\\_SB\\_DEFAULT\\_QOS](#) (([CFE\\_SB\\_Qos\\_t](#)) {0})  
*Default Qos macro.*

### Typedefs

- typedef union [CFE\\_SB\\_Msg](#) [CFE\\_SB\\_Buffer\\_t](#)  
*Software Bus generic message.*

#### 12.113.1 Detailed Description

Purpose: This header file contains all definitions for the cFE Software Bus Application Programmer's Interface.  
Author: R.McGraw/SSI

## 12.113.2 Macro Definition Documentation

### 12.113.2.1 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.

### 12.113.2.2 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.

### 12.113.2.3 CFE\_SB\_INVALID\_PIPE #define CFE\_SB\_INVALID\_PIPE CFE\_SB\_PIPEID\_C(CFE\_RESOURCEID\_UNDEFINED)

A [CFE\\_SB\\_Pipeld\\_t](#) value which is always invalid.

This may be used as a safe initializer for [CFE\\_SB\\_Pipeld\\_t](#) values

Definition at line 125 of file cfe\_sb\_api\_typedefs.h.

### 12.113.2.4 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.

### 12.113.2.5 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.

**12.113.2.6 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.

**12.113.2.7 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.

**12.113.2.8 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.

**12.113.2.9 CFE\_SB\_PIPEID\_C** #define CFE\_SB\_PIPEID\_C(  
    *val* ) (([CFE\\_SB\\_PipeId\\_t](#))CFE\_RESOURCEID\_WRAP(*val*))

Cast/Convert a generic CFE\_Resourceld\_t to a CFE\_SB\_PipeId\_t.

Definition at line 118 of file cfe\_sb\_api\_typedefs.h.

**12.113.2.10 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.

**12.113.2.11 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.

**12.113.2.12 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.

### 12.113.3 Typedef Documentation

**12.113.3.1 CFE\_SB\_Buffer\_t** `typedef union CFE_SB_Msg CFE_SB_Buffer_t`  
 Software Bus generic message.

## 12.114 cfe/modules/core\_api/fsw/inc/cfe\_tbl.h File Reference

```
#include "common_types.h"
#include "cfe_error.h"
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_tbl_api_typedefs.h"
#include "cfe_sb_api_typedefs.h"
#include "cfe_resourceid.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.*

### Conversions for TBL resource IDs

*These inline functions enforce the type of the argument(s), such that comparing a table handle to some other type of resource ID will result in a compiler error.*

- static bool `CFE_TBL_HandleID_IsEqual (CFE_TBL_HandleId_t x, CFE_TBL_HandleId_t y)`
- static unsigned long `CFE_TBL_HandleID_AsInt (CFE_TBL_HandleId_t x)`
- static bool `CFE_TBL_HandleID_IsDefined (CFE_TBL_HandleId_t x)`
- `CFE_TBL_HandleId_t CFE_TBL_HandleToID (CFE_TBL_Handle_t TblHandle)`
- `CFE_TBL_Handle_t CFE_TBL_HandleFromID (CFE_TBL_HandleId_t TblId)`

#### 12.114.1 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:

#### 12.114.2 Function Documentation

##### 12.114.2.1 `CFE_TBL_HandleFromID()` `CFE_TBL_Handle_t CFE_TBL_HandleFromID ( CFE_TBL_HandleId_t TblId )`

Convert an ID to an API handle

##### 12.114.2.2 `CFE_TBL_HandleID_AsInt()` `static unsigned long CFE_TBL_HandleID_AsInt ( CFE_TBL_HandleId_t x ) [inline], [static]`

Get the integer representation of the ID

Definition at line 65 of file cfe\_tbl.h.

References CFE\_RESOURCEID\_TO ULONG.

##### 12.114.2.3 `CFE_TBL_HandleID_IsDefined()` `static bool CFE_TBL_HandleID_IsDefined ( CFE_TBL_HandleId_t x ) [inline], [static]`

Check if the ID is valid

Definition at line 73 of file cfe\_tbl.h.

References CFE\_RESOURCEID\_TEST\_DEFINED.

##### 12.114.2.4 `CFE_TBL_HandleID_IsEqual()` `static bool CFE_TBL_HandleID_IsEqual ( CFE_TBL_HandleId_t x, CFE_TBL_HandleId_t y ) [inline], [static]`

Check for equality between two IDs

Definition at line 57 of file cfe\_tbl.h.

References CFE\_RESOURCEID\_TEST\_EQUAL.

### 12.114.2.5 CFE\_TBL\_HandleToID() `CFE_TBL_HandleId_t` CFE\_TBL\_HandleToID ( `CFE_TBL_Handle_t TblHandle` )

Convert an API handle to an ID

## 12.115 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"
#include "cfe_resourceid_api_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`)(-1))  
*Bad table handle.*

- #define CFE\_TBL\_HANDLE\_EQ(x, y) ((x) == (y))
- #define CFE\_TBL\_HANDLE\_INT(x) ((unsigned long)(x))
- #define CFE\_TBL\_HANDLE\_IS\_VALID(x) ((x) != CFE\_TBL\_BAD\_TABLE\_HANDLE)

#### Constants for table registry IDs, using the CFE\_TBL\_RegId\_t type

- #define CFE\_TBL\_REGID\_C(x) ((CFE\_TBL\_RegId\_t)CFE\_RESOURCEID\_WRAP(x))
- #define CFE\_TBL\_REGID\_UNDEFINED CFE\_TBL\_REGID\_C(CFE\_RESOURCEID\_UNDEFINED)

#### Constants for table handle IDs, using the CFE\_TBL\_HandleId\_t type

- #define CFE\_TBL\_HANDLEID\_C(x) ((CFE\_TBL\_HandleId\_t)CFE\_RESOURCEID\_WRAP(x))
- #define CFE\_TBL\_HANDLEID\_UNDEFINED CFE\_TBL\_HANDLEID\_C(CFE\_RESOURCEID\_UNDEFINED)

### Typedefs

- typedef int32(\* CFE\_TBL\_CallbackFuncPtr\_t) (void \*TblPtr)  
*Table Callback Function.*
- typedef enum CFE\_TBL\_SrcEnum CFE\_TBL\_SrcEnum\_t  
*Table Source.*
- typedef struct CFE\_TBL\_Info CFE\_TBL\_Info\_t  
*Table Info.*
- typedef int16 CFE\_TBL\_Handle\_t

### Enumerations

- enum CFE\_TBL\_SrcEnum { CFE\_TBL\_SRC\_FILE = 0 , CFE\_TBL\_SRC\_ADDRESS }  
*Table Source.*

#### 12.115.1 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:

#### 12.115.2 Macro Definition Documentation

##### 12.115.2.1 CFE\_TBL\_BAD\_TABLE\_HANDLE #define CFE\_TBL\_BAD\_TABLE\_HANDLE ((CFE\_TBL\_Handle\_t) (-1)) Bad table handle.

Definition at line 173 of file cfe\_tbl\_api\_typedefs.h.

##### 12.115.2.2 CFE\_TBL\_HANDLE\_EQ #define CFE\_TBL\_HANDLE\_EQ(                   x,                   y ) ((x) == (y))

Definition at line 175 of file cfe\_tbl\_api\_typedefs.h.

**12.115.2.3 CFE\_TBL\_HANDLE\_INT** #define CFE\_TBL\_HANDLE\_INT(  
  x) ((unsigned long)(x))

Definition at line 176 of file cfe\_tbl\_api\_typedefs.h.

**12.115.2.4 CFE\_TBL\_HANDLE\_IS\_VALID** #define CFE\_TBL\_HANDLE\_IS\_VALID(  
  x) ((x) != CFE\_TBL\_BAD\_TABLE\_HANDLE)

Definition at line 177 of file cfe\_tbl\_api\_typedefs.h.

**12.115.2.5 CFE\_TBL\_HANDLEID\_C** #define CFE\_TBL\_HANDLEID\_C(  
  x) ((CFE\_TBL\_HandleId\_t)CFE\_RESOURCEID\_WRAP(x))

Definition at line 136 of file cfe\_tbl\_api\_typedefs.h.

**12.115.2.6 CFE\_TBL\_HANDLEID\_UNDEFINED** #define CFE\_TBL\_HANDLEID\_UNDEFINED CFE\_TBL\_HANDLEID\_C(CFE\_RESOURCEID\_UNDEFINED)

Definition at line 137 of file cfe\_tbl\_api\_typedefs.h.

**12.115.2.7 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 77 of file cfe\_tbl\_api\_typedefs.h.

**12.115.2.8 CFE\_TBL\_REGID\_C** #define CFE\_TBL\_REGID\_C(  
  x) ((CFE\_TBL\_RegId\_t)CFE\_RESOURCEID\_WRAP(x))

Definition at line 128 of file cfe\_tbl\_api\_typedefs.h.

**12.115.2.9 CFE\_TBL\_REGID\_UNDEFINED** #define CFE\_TBL\_REGID\_UNDEFINED CFE\_TBL\_REGID\_C(CFE\_RESOURCEID\_UNDEFINED)

Definition at line 129 of file cfe\_tbl\_api\_typedefs.h.

## 12.115.3 Typedef Documentation

**12.115.3.1 CFE\_TBL\_CallbackFuncPtr\_t** typedef int32 (\* CFE\_TBL\_CallbackFuncPtr\_t) (void \*TblPtr)

Table Callback Function.

Definition at line 82 of file cfe\_tbl\_api\_typedefs.h.

**12.115.3.2 CFE\_TBL\_Handle\_t** typedef int16 CFE\_TBL\_Handle\_t

Definition at line 170 of file cfe\_tbl\_api\_typedefs.h.

**12.115.3.3 CFE\_TBL\_Info\_t** typedef struct CFE\_TBL\_Info CFE\_TBL\_Info\_t

Table Info.

**12.115.3.4 CFE\_TBL\_SrcEnum\_t** `typedef enum CFE_TBL_SrcEnum CFE_TBL_SrcEnum_t`  
Table Source.

#### 12.115.4 Enumeration Type Documentation

**12.115.4.1 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 <code>Size</code> parameter.

Definition at line 85 of file `cfe_tbl_api_typedefs.h`.

### 12.116 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.*

#### 12.116.1 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:

## 12.116.2 Macro Definition Documentation

```
12.116.2.1 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.
```

## 12.116.3 Typedef Documentation

```
12.116.3.1 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.

## 12.117 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.*

- **CFE\_Status\_t 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.*

### 12.117.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

### 12.117.2 Macro Definition Documentation

#### 12.117.2.1 CFE\_TIME\_Copy #define CFE\_TIME\_Copy (

```
m,  
t )
```

##### Value:

```
{  
    (m)->Seconds      = (t)->Seconds;  
    (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.

## 12.118 cfe/modules/core\_api/fsw/inc/cfe\_time\_api\_typedefs.h File Reference

```
#include "common_types.h"  
#include "cfe_time_extern_typedefs.h"
```

### Macros

- #define **CFE\_TIME\_PRINTED\_STRING\_SIZE** 24  
*Required size of buffer to be passed into CFE\_TIME\_Print (includes null terminator)*
- #define **CFE\_TIME\_ZERO\_VALUE** ((CFE\_TIME\_SysTime\_t){0, 0})

### TypeDefs

- typedef enum **CFE\_TIME\_Compare** CFE\_TIME\_Compare\_t  
*Enumerated types identifying the relative relationships of two times.*
- typedef int32(\* **CFE\_TIME\_SynchCallbackPtr\_t**) (void)  
*Time Synchronization Callback Function Ptr Type.*

### Enumerations

- enum **CFE\_TIME\_Compare** { **CFE\_TIME\_A\_LT\_B** = -1 , **CFE\_TIME\_EQUAL** = 0 , **CFE\_TIME\_A\_GT\_B** = 1 }  
*Enumerated types identifying the relative relationships of two times.*

### 12.118.1 Detailed Description

Purpose: cFE Time Services (TIME) library API header file

Author: S.Walling/Microtel

Notes:

### 12.118.2 Macro Definition Documentation

#### 12.118.2.1 CFE\_TIME\_PRINTED\_STRING\_SIZE `#define CFE_TIME_PRINTED_STRING_SIZE 24`

Required size of buffer to be passed into [CFE\\_TIME\\_Print](#) (includes null terminator)

Definition at line 45 of file `cfe_time_api_typedefs.h`.

#### 12.118.2.2 CFE\_TIME\_ZERO\_VALUE `#define CFE_TIME_ZERO_VALUE ((CFE_TIME_SysTime_t){0, 0})`

A general-purpose initializer for `CFE_TIME_SysTime_t` values.

Represents "time zero" in the `CFE_TIME_SysTime_t` domain. This can be used as a general purpose initializer for instantiations of the `CFE_TIME_SysTime_t` type.

Definition at line 54 of file `cfe_time_api_typedefs.h`.

### 12.118.3 Typedef Documentation

#### 12.118.3.1 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.

#### 12.118.3.2 CFE\_TIME\_SynchCallbackPtr\_t `typedef int32 (* CFE_TIME_SynchCallbackPtr_t) (void)`

Time Synchronization Callback Function Ptr Type.

##### Description

Applications that wish to get direct notification of the receipt of the cFE Time Synchronization signal (typically a 1 Hz signal), must register a callback function with the following prototype via the [CFE\\_TIME\\_RegisterSynchCallback](#) API.

Definition at line 84 of file `cfe_time_api_typedefs.h`.

### 12.118.4 Enumeration Type Documentation

#### 12.118.4.1 CFE\_TIME\_Compare `enum CFE_TIME_Compare`

Enumerated types identifying the relative relationships of two times.

### Description

Since time fields contain numbers that are relative to an epoch time, then it is possible for a time value to be "negative". This can lead to some confusion about what relationship exists between two time values. To resolve this confusion, the cFE provides the API [CFE\\_TIME\\_Compare](#) which returns these enumerated values.

### Enumerator

CFE_TIME_A_LT_B	The first specified time is considered to be before the second specified time.
CFE_TIME_EQUAL	The two specified times are considered to be equal.
CFE_TIME_A_GT_B	The first specified time is considered to be after the second specified time.

Definition at line 69 of file `cfe_time_api_typedefs.h`.

## 12.119 cfe/modules/core\_api/fsw/inc/cfe\_version.h File Reference

### Macros

- `#define CFE_BUILD_NUMBER 0`

*Development: Number of development git commits since CFE\_BUILD\_BASELINE.*
- `#define CFE_BUILD_BASELINE "v7.0.0"`

*Development: Reference git tag for build number.*
- `#define CFE_BUILD_DEV_CYCLE "v7.0.0"`

*Development: Release name for current development cycle.*
- `#define CFE_BUILD_CODENAME "Draco"`

*: Development: Code name for the current build*
- `#define CFE_MAJOR_VERSION 7`

*Major version number.*
- `#define CFE_MINOR_VERSION 0`

*Minor version number.*
- `#define CFE_REVISION 0`

*Revision version number.*
- `#define CFE_LAST_OFFICIAL "v7.0.0"`

*Last official release.*
- `#define CFE_MISSION_REV 0x0`

*Mission revision.*
- `#define CFE_STR_HELPER(x) #x`

*Convert argument to string.*
- `#define CFE_STR(x) CFE_STR_HELPER(x)`

*Expand macro before conversion.*
- `#define CFE_SRC_VERSION CFE_BUILD_BASELINE "+dev" CFE_STR(CFE_BUILD_NUMBER)`

*Short Build Version String.*
- `#define CFE_CFG_MAX_VERSION_STR_LEN 256`

*Max Version String length.*

### 12.119.1 Detailed Description

Provide version identifiers for the cFE core. See [Version Numbers](#) for further details.

## 12.119.2 Macro Definition Documentation

### 12.119.2.1 CFE\_BUILD\_BASELINE `#define CFE_BUILD_BASELINE "v7.0.0"`

Development: Reference git tag for build number.

Definition at line 30 of file cfe\_version.h.

### 12.119.2.2 CFE\_BUILD\_CODENAME `#define CFE_BUILD_CODENAME "Draco"`

: Development: Code name for the current build

Definition at line 32 of file cfe\_version.h.

### 12.119.2.3 CFE\_BUILD\_DEV\_CYCLE `#define CFE_BUILD_DEV_CYCLE "v7.0.0"`

Development: Release name for current development cycle.

Definition at line 31 of file cfe\_version.h.

### 12.119.2.4 CFE\_BUILD\_NUMBER `#define CFE_BUILD_NUMBER 0`

Development: Number of development git commits since CFE\_BUILD\_BASELINE.

Definition at line 29 of file cfe\_version.h.

### 12.119.2.5 CFE\_CFG\_MAX\_VERSION\_STR\_LEN `#define CFE_CFG_MAX_VERSION_STR_LEN 256`

Max Version String length.

Maximum length that a CFE version string can be.

Definition at line 69 of file cfe\_version.h.

### 12.119.2.6 CFE\_LAST\_OFFICIAL `#define CFE_LAST_OFFICIAL "v7.0.0"`

Last official release.

Definition at line 42 of file cfe\_version.h.

### 12.119.2.7 CFE\_MAJOR\_VERSION `#define CFE_MAJOR_VERSION 7`

Major version number.

Definition at line 35 of file cfe\_version.h.

### 12.119.2.8 CFE\_MINOR\_VERSION `#define CFE_MINOR_VERSION 0`

Minor version number.

Definition at line 36 of file cfe\_version.h.

### 12.119.2.9 CFE\_MISSION\_REV `#define CFE_MISSION_REV 0x0`

Mission revision.

Values 1-254 are reserved for mission use to denote patches/customizations as needed. Value of 0 is reserved for official releases only Value of 255 (0xFF) is reserved for development and testing versions

Definition at line 51 of file cfe\_version.h.

**12.119.2.10 CFE\_REVISION** #define CFE\_REVISION 0  
 Revision version number.  
 Definition at line 37 of file cfe\_version.h.

**12.119.2.11 CFE\_SRC\_VERSION** #define CFE\_SRC\_VERSION CFE\_BUILD\_BASELINE "+dev" CFE\_STR(CFE\_BUILD\_NUMBER)  
 Short Build Version String.  
 Short string identifying the build, see [Version Numbers](#) for suggested format for development and official releases.  
 Definition at line 62 of file cfe\_version.h.

**12.119.2.12 CFE\_STR** #define CFE\_STR(  
 x ) CFE\_STR\_HELPER(x)  
 Expand macro before conversion.  
 Definition at line 54 of file cfe\_version.h.

**12.119.2.13 CFE\_STR\_HELPER** #define CFE\_STR\_HELPER(  
 x ) #x  
 Convert argument to string.  
 Definition at line 53 of file cfe\_version.h.

## 12.120 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"
#include "cfe_es_memaddress.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.*

### 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 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.*

## 12.120.1 Detailed Description

Declarations and prototypes for cfe\_es\_extern\_typedefs module

## 12.120.2 Typedef Documentation

### 12.120.2.1 [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 314 of file default\_cfe\_es\_extern\_typedefs.h.

### 12.120.2.2 [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.

### 12.120.2.3 [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 307 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.4 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 106 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.5 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.

**12.120.2.6 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 350 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.7 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.

**12.120.2.8 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 335 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.9 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 78 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.10 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 328 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.11 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 254 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.12 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 55 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.13 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 343 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.14 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](#)

**12.120.2.15 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 174 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.16 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 231 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.17 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 321 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.2.18 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.

**12.120.2.19 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 360 of file default\_cfe\_es\_extern\_typedefs.h.

## 12.120.3 Enumeration Type Documentation

**12.120.3.1 CFE\_ES\_AppState** `enum CFE_ES_AppState`

Label definitions associated with CFE\_ES\_AppState\_Enum\_t.

**Enumerator**

<code>CFE_ES_AppState_UNDEFINED</code>	Initial state before app thread is started.
<code>CFE_ES_AppState_EARLY_INIT</code>	App thread has started, app performing early initialization of its own data.
<code>CFE_ES_AppState_LATE_INIT</code>	Early/Local initialization is complete. First sync point.
<code>CFE_ES_AppState_RUNNING</code>	All initialization is complete. Second sync point.
<code>CFE_ES_AppState_WAITING</code>	Application is waiting on a Restart/Reload/Delete request.
<code>CFE_ES_AppState_STOPPED</code>	Application is stopped.
<code>CFE_ES_AppState_MAX</code>	Reserved entry, marker for the maximum state.

Definition at line 259 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.3.2 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 83 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.3.3 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 60 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.3.4 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 236 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.3.5 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 37 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.3.6 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.

**Enumerator**

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 111 of file default\_cfe\_es\_extern\_typedefs.h.

**12.120.3.7 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 179 of file default\_cfe\_es\_extern\_typedefs.h.

**12.121 cfe/modules/es/config/default\_cfe\_es\_fcncode\_values.h File Reference****Macros**

- #define CFE\_ES\_CCVAL(x) CFE\_ES\_FunctionCode\_##x

**Enumerations**

- enum CFE\_ES\_FunctionCode\_ {
 CFE\_ES\_FunctionCode\_NOOP = 0 , CFE\_ES\_FunctionCode\_RESET\_COUNTERS = 1 , CFE\_ES\_FunctionCode\_RESTART = 2 , CFE\_ES\_FunctionCode\_START\_APP = 4 ,
 CFE\_ES\_FunctionCode\_STOP\_APP = 5 , CFE\_ES\_FunctionCode\_RESTART\_APP = 6 , CFE\_ES\_FunctionCode\_RELOAD\_APP = 7 , CFE\_ES\_FunctionCode\_QUERY\_ONE = 8 ,
 CFE\_ES\_FunctionCode\_QUERY\_ALL = 9 , CFE\_ES\_FunctionCode\_CLEAR\_SYS\_LOG = 10 , CFE\_ES\_FunctionCode\_WRITE\_S = 11 , CFE\_ES\_FunctionCode\_CLEAR\_ER\_LOG = 12 ,
 CFE\_ES\_FunctionCode\_WRITE\_ER\_LOG = 13 , CFE\_ES\_FunctionCode\_START\_PERF\_DATA = 14 ,
 CFE\_ES\_FunctionCode\_STOP\_PERF\_DATA = 15 , CFE\_ES\_FunctionCode\_SET\_PERF\_FILTER\_MASK = 16 ,
 CFE\_ES\_FunctionCode\_SET\_PERF\_TRIGGER\_MASK = 17 , CFE\_ES\_FunctionCode\_OVER\_WRITE\_SYS\_LOG = 18 , CFE\_ES\_FunctionCode\_RESET\_PR\_COUNT = 19 , CFE\_ES\_FunctionCode\_SET\_MAX\_PR\_COUNT =
 }

```
20 ,
CFE_ES_FunctionCode_DELETE_CDS = 21 , CFE_ES_FunctionCode_SEND_MEM_POOL_STATS = 22 ,
CFE_ES_FunctionCode_DUMP_CDS_REGISTRY = 23 , CFE_ES_FunctionCode_QUERY_ALL_TASKS = 24 }
```

### 12.121.1 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.

### 12.121.2 Macro Definition Documentation

#### 12.121.2.1 CFE\_ES\_CCVAL #define CFE\_ES\_CCVAL (

```
    x ) CFE_ES_FunctionCode_##x
```

Definition at line 35 of file default\_cfe\_es\_fcncode\_values.h.

### 12.121.3 Enumeration Type Documentation

#### 12.121.3.1 CFE\_ES\_FunctionCode\_ enum CFE\_ES\_FunctionCode\_

##### Enumerator

CFE_ES_FunctionCode_NOOP
CFE_ES_FunctionCode_RESET_COUNTERS
CFE_ES_FunctionCode_RESTART
CFE_ES_FunctionCode_START_APP
CFE_ES_FunctionCode_STOP_APP
CFE_ES_FunctionCode_RESTART_APP
CFE_ES_FunctionCode_RELOAD_APP
CFE_ES_FunctionCode_QUERY_ONE
CFE_ES_FunctionCode_QUERY_ALL
CFE_ES_FunctionCode_CLEAR_SYS_LOG
CFE_ES_FunctionCode_WRITE_SYS_LOG
CFE_ES_FunctionCode_CLEAR_ER_LOG
CFE_ES_FunctionCode_WRITE_ER_LOG
CFE_ES_FunctionCode_START_PERF_DATA
CFE_ES_FunctionCode_STOP_PERF_DATA
CFE_ES_FunctionCode_SET_PERF_FILTER_MASK
CFE_ES_FunctionCode_SET_PERF_TRIGGER_MASK
CFE_ES_FunctionCode_OVER_WRITE_SYS_LOG
CFE_ES_FunctionCode_RESET_PR_COUNT
CFE_ES_FunctionCode_SET_MAX_PR_COUNT
CFE_ES_FunctionCode_DELETE_CDS
CFE_ES_FunctionCode_SEND_MEM_POOL_STATS
CFE_ES_FunctionCode_DUMP_CDS_REGISTRY
CFE_ES_FunctionCode_QUERY_ALL_TASKS

Definition at line 37 of file default\_cfe\_es\_fcncode\_values.h.

## 12.122 cfe/modules/es/config/default\_cfe\_es\_interface\_cfg\_values.h File Reference

### Macros

- #define CFE\_MISSION\_ES\_CFGVAL(x) DEFAULT\_CFE\_MISSION\_ES\_##x

#### 12.122.1 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.

#### 12.122.2 Macro Definition Documentation

##### 12.122.2.1 CFE\_MISSION\_ES\_CFGVAL #define CFE\_MISSION\_ES\_CFGVAL ( x ) DEFAULT\_CFE\_MISSION\_ES\_##x

Definition at line 36 of file default\_cfe\_es\_interface\_cfg\_values.h.

## 12.123 cfe/modules/es/config/default\_cfe\_es\_internal\_cfg\_values.h File Reference

### Macros

- #define CFE\_PLATFORM\_ES\_CFGVAL(x) DEFAULT\_CFE\_PLATFORM\_ES\_##x

#### 12.123.1 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.

#### 12.123.2 Macro Definition Documentation

##### 12.123.2.1 CFE\_PLATFORM\_ES\_CFGVAL #define CFE\_PLATFORM\_ES\_CFGVAL ( x ) DEFAULT\_CFE\_PLATFORM\_ES\_##x

Definition at line 36 of file default\_cfe\_es\_internal\_cfg\_values.h.

## 12.124 cfe/modules/es/config/default\_cfe\_es\_memaddress.h File Reference

```
#include "common_types.h"
```

## 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))`  
*Memory Address initializer wrapper.*
- `#define CFE_ES_MEMADDRESS_TO_PTR(x) ((void *)(cpuaddr)(x))`  
*Memory Address to pointer wrapper.*

## Typedefs

- `typedef uint64 CFE_ES_MemOffset_t`  
*Type used for memory sizes and offsets in commands and telemetry.*
- `typedef uint64 CFE_ES_MemAddress_t`  
*Type used for memory addresses in command and telemetry messages.*

### 12.124.1 Detailed Description

Example header file override that defines memory addresses and offsets to be a full 64 bit integer value. This is the simplest and most efficient approach to use 64 bit addressing, but may introduce unexpected padding in certain cases. On most systems a uint64 value needs to be aligned on a 64-bit boundary, and CFS has some TLM and CMD structures where this is not the case. These will be padded by the compiler and may cause interoperability issues. To use this implementation, clone this file as "cfe\_es\_memaddress.h" in your local defs dir.

### 12.124.2 Macro Definition Documentation

#### 12.124.2.1 CFE\_ES\_MEMADDRESS\_C `#define CFE_ES_MEMADDRESS_C(`   `x ) ( (CFE_ES_MemAddress_t) (cpuaddr) (x) )`

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 81 of file default\_cfe\_es\_memaddress.h.

#### 12.124.2.2 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 89 of file default\_cfe\_es\_memaddress.h.

#### 12.124.2.3 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 58 of file default\_cfe\_es\_memaddress.h.

**12.124.2.4 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 66 of file default\_cfe\_es\_memaddress.h.

### 12.124.3 Typedef Documentation

**12.124.3.1 CFE\_ES\_MemAddress\_t** typedef uint64 CFE\_ES\_MemAddress\_t

Type used for memory addresses in command and telemetry messages.

Use a full 64-bit integer value for memory offsets

Definition at line 73 of file default\_cfe\_es\_memaddress.h.

**12.124.3.2 CFE\_ES\_MemOffset\_t** typedef uint64 CFE\_ES\_MemOffset\_t

Type used for memory sizes and offsets in commands and telemetry.

Use a full 64-bit integer value for memory offsets

Definition at line 50 of file default\_cfe\_es\_memaddress.h.

## 12.125 cfe/modules/es/config/default\_cfe\_es\_mission\_cfg.h File Reference

```
#include "cfe_es_interface_cfg.h"
```

### 12.125.1 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.

## 12.126 cfe/modules/es/config/default\_cfe\_es\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_es_fcncodes.h"
#include "cfe_es_msgdefs.h"
#include "cfe_es_msgstruct.h"
```

### 12.126.1 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.

## 12.127 cfe/modules/es/config/default\_cfe\_es\_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_es_fcncodes.h"
```

### Data Structures

- struct [CFE\\_ES\\_RestartCmd\\_Payload](#)  
*Restart cFE Command Payload.*
- struct [CFE\\_ES\\_FileNameCmd\\_Payload](#)  
*Generic file name command payload.*
- struct [CFE\\_ES\\_OverWriteSysLogCmd\\_Payload](#)  
*Overwrite/Discard System Log Configuration Command Payload.*
- struct [CFE\\_ES\\_StartAppCmd\\_Payload](#)  
*Start Application Command Payload.*
- struct [CFE\\_ES\\_AppNameCmd\\_Payload](#)  
*Generic application name command payload.*
- struct [CFE\\_ES\\_AppReloadCmd\\_Payload](#)  
*Reload Application Command Payload.*
- struct [CFE\\_ES\\_SetMaxPRCountCmd\\_Payload](#)  
*Set Maximum Processor Reset Count Command Payload.*
- struct [CFE\\_ES\\_DeleteCDSCmd\\_Payload](#)  
*Delete Critical Data Store Command Payload.*
- struct [CFE\\_ES\\_StartPerfCmd\\_Payload](#)  
*Start Performance Analyzer Command Payload.*
- struct [CFE\\_ES\\_StopPerfCmd\\_Payload](#)  
*Stop Performance Analyzer Command Payload.*
- struct [CFE\\_ES\\_SetPerfFilterMaskCmd\\_Payload](#)  
*Set Performance Analyzer Filter Mask Command Payload.*
- struct [CFE\\_ES\\_SetPerfTrigMaskCmd\\_Payload](#)  
*Set Performance Analyzer Trigger Mask Command Payload.*
- struct [CFE\\_ES\\_SendMemPoolStatsCmd\\_Payload](#)  
*Send Memory Pool Statistics Command Payload.*
- struct [CFE\\_ES\\_DumpCDSRegistryCmd\\_Payload](#)  
*Dump CDS Registry Command Payload.*
- struct [CFE\\_ES\\_OneAppTlm\\_Payload](#)
- struct [CFE\\_ES\\_PoolStatsTlm\\_Payload](#)
- struct [CFE\\_ES\\_HousekeepingTlm\\_Payload](#)

### Typedefs

- typedef struct [CFE\\_ES\\_RestartCmd\\_Payload](#) [CFE\\_ES\\_RestartCmd\\_Payload\\_t](#)  
*Restart cFE Command Payload.*
- typedef struct [CFE\\_ES\\_FileNameCmd\\_Payload](#) [CFE\\_ES\\_FileNameCmd\\_Payload\\_t](#)  
*Generic file name command payload.*
- typedef struct [CFE\\_ES\\_OverWriteSysLogCmd\\_Payload](#) [CFE\\_ES\\_OverWriteSysLogCmd\\_Payload\\_t](#)  
*Overwrite/Discard System Log Configuration Command Payload.*

- `typedef struct CFE_ES_StartAppCmd_Payload CFE_ES_StartAppCmd_Payload_t`  
*Start Application Command Payload.*
- `typedef struct CFE_ES_AppNameCmd_Payload CFE_ES_AppNameCmd_Payload_t`  
*Generic application name command payload.*
- `typedef struct CFE_ES_AppReloadCmd_Payload CFE_ES_AppReloadCmd_Payload_t`  
*Reload Application Command Payload.*
- `typedef struct CFE_ES_SetMaxPRCountCmd_Payload CFE_ES_SetMaxPRCountCmd_Payload_t`  
*Set Maximum Processor Reset Count Command Payload.*
- `typedef struct CFE_ES_DeleteCDSCmd_Payload CFE_ES_DeleteCDSCmd_Payload_t`  
*Delete Critical Data Store Command Payload.*
- `typedef uint32 CFE_ES_PerfMode_Enum_t`
- `typedef struct CFE_ES_StartPerfCmd_Payload CFE_ES_StartPerfCmd_Payload_t`  
*Start Performance Analyzer Command Payload.*
- `typedef struct CFE_ES_StopPerfCmd_Payload CFE_ES_StopPerfCmd_Payload_t`  
*Stop Performance Analyzer Command Payload.*
- `typedef struct CFE_ES_SetPerfFilterMaskCmd_Payload CFE_ES_SetPerfFilterMaskCmd_Payload_t`  
*Set Performance Analyzer Filter Mask Command Payload.*
- `typedef struct CFE_ES_SetPerfTrigMaskCmd_Payload CFE_ES_SetPerfTrigMaskCmd_Payload_t`  
*Set Performance Analyzer Trigger Mask Command Payload.*
- `typedef struct CFE_ES_SendMemPoolStatsCmd_Payload CFE_ES_SendMemPoolStatsCmd_Payload_t`  
*Send Memory Pool Statistics Command Payload.*
- `typedef struct CFE_ES_DumpCDSRegistryCmd_Payload CFE_ES_DumpCDSRegistryCmd_Payload_t`  
*Dump CDS Registry Command Payload.*
- `typedef struct CFE_ES_OneAppTlm_Payload CFE_ES_OneAppTlm_Payload_t`
- `typedef struct CFE_ES_PoolStatsTlm_Payload CFE_ES_PoolStatsTlm_Payload_t`
- `typedef struct CFE_ES_HousekeepingTlm_Payload CFE_ES_HousekeepingTlm_Payload_t`

## Enumerations

- `enum CFE_ES_PerfMode { CFE_ES_PerfTrigger_START = 0, CFE_ES_PerfTrigger_CENTER, CFE_ES_PerfTrigger_END }`  
*Labels for values to use in `CFE_ES_StartPerfCmd_Payload.TriggerMode`.*

### 12.127.1 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

### 12.127.2 Typedef Documentation

#### 12.127.2.1 `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](#)

#### 12.127.2.2 `CFE_ES_AppReloadCmd_Payload_t` `typedef struct CFE_ES_AppReloadCmd_Payload CFE_ES_AppReloadCmd_Payload_t`

Reload Application Command Payload.

For command details, see [CFE\\_ES\\_RELOAD\\_APP\\_CC](#)

**12.127.2.3 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](#)

**12.127.2.4 CFE\_ES\_DumpCDSRegistryCmd\_Payload\_t** `typedef struct CFE_ES_DumpCDSRegistryCmd_Payload`  
`CFE_ES_DumpCDSRegistryCmd_Payload_t`  
Dump CDS Registry Command Payload.  
For command details, see [CFE\\_ES\\_DUMP\\_CDS\\_REGISTRY\\_CC](#)

**12.127.2.5 CFE\_ES\_FileNameCmd\_Payload\_t** `typedef struct CFE_ES_FileNameCmd_Payload CFE_ES_FileNameCmd_Payload_t`  
Generic file name command payload.  
This format is shared by several executive services commands. For command details, see [CFE\\_ES\\_QUERY\\_ALL\\_CC](#),  
[CFE\\_ES\\_QUERY\\_ALL\\_TASKS\\_CC](#), [CFE\\_ES\\_WRITE\\_SYS\\_LOG\\_CC](#), and [CFE\\_ES\\_WRITE\\_ER\\_LOG\\_CC](#)

**12.127.2.6 CFE\_ES\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_ES_HousekeepingTlm_Payload CFE_ES_HousekeepingTlm_Payload_t`  
**Name** Executive Services Housekeeping Packet

**12.127.2.7 CFE\_ES\_OneAppTlm\_Payload\_t** `typedef struct CFE_ES_OneAppTlm_Payload CFE_ES_OneAppTlm_Payload_t`  
**Name** Single Application Information Packet

**12.127.2.8 CFE\_ES\_OverWriteSysLogCmd\_Payload\_t** `typedef struct CFE_ES_OverWriteSysLogCmd_Payload`  
`CFE_ES_OverWriteSysLogCmd_Payload_t`  
Overwrite/Discard System Log Configuration Command Payload.  
For command details, see [CFE\\_ES\\_OVER\\_WRITE\\_SYS\\_LOG\\_CC](#)

**12.127.2.9 CFE\_ES\_PerfMode\_Elem\_t** `typedef uint32 CFE_ES_PerfMode_Elem_t`  
Definition at line 157 of file default\_cfe\_es\_msgdefs.h.

**12.127.2.10 CFE\_ES\_PoolStatsTlm\_Payload\_t** `typedef struct CFE_ES_PoolStatsTlm_Payload CFE_ES_PoolStatsTlm_Payload_t`  
**Name** Memory Pool Statistics Packet

**12.127.2.11 CFE\_ES\_RestartCmd\_Payload\_t** `typedef struct CFE_ES_RestartCmd_Payload CFE_ES_RestartCmd_Payload_t`  
Restart cFE Command Payload.  
For command details, see [CFE\\_ES\\_RESTART\\_CC](#)

**12.127.2.12 CFE\_ES\_SendMemPoolStatsCmd\_Payload\_t** `typedef struct CFE_ES_SendMemPoolStatsCmd_Payload`  
`CFE_ES_SendMemPoolStatsCmd_Payload_t`  
Send Memory Pool Statistics Command Payload.  
For command details, see [CFE\\_ES\\_SEND\\_MEM\\_POOL\\_STATS\\_CC](#)

**12.127.2.13 CFE\_ES\_SetMaxPRCountCmd\_Payload\_t** `typedef struct CFE_ES_SetMaxPRCountCmd_Payload`  
`CFE_ES_SetMaxPRCountCmd_Payload_t`  
Set Maximum Processor Reset Count Command Payload.  
For command details, see [CFE\\_ES\\_SET\\_MAX\\_PR\\_COUNT\\_CC](#)

**12.127.2.14 CFE\_ES\_SetPerfFilterMaskCmd\_Payload\_t** `typedef struct CFE_ES_SetPerfFilterMaskCmd_Payload CFE_ES_SetPerfFilterMaskCmd_Payload_t`

Set Performance Analyzer Filter Mask Command Payload.

For command details, see [CFE\\_ES\\_SET\\_PERF\\_FILTER\\_MASK\\_CC](#)

**12.127.2.15 CFE\_ES\_SetPerfTrigMaskCmd\_Payload\_t** `typedef struct CFE_ES_SetPerfTrigMaskCmd_Payload CFE_ES_SetPerfTrigMaskCmd_Payload_t`

Set Performance Analyzer Trigger Mask Command Payload.

For command details, see [CFE\\_ES\\_SET\\_PERF\\_TRIGGER\\_MASK\\_CC](#)

**12.127.2.16 CFE\_ES\_StartAppCmd\_Payload\_t** `typedef struct CFE_ES_StartAppCmd_Payload CFE_ES_StartAppCmd_Payload_t`

Start Application Command Payload.

For command details, see [CFE\\_ES\\_START\\_APP\\_CC](#)

**12.127.2.17 CFE\_ES\_StartPerfCmd\_Payload\_t** `typedef struct CFE_ES_StartPerfCmd_Payload CFE_ES_StartPerfCmd_Payload_t`

Start Performance Analyzer Command Payload.

For command details, see [CFE\\_ES\\_START\\_PERF\\_DATA\\_CC](#)

**12.127.2.18 CFE\_ES\_StopPerfCmd\_Payload\_t** `typedef struct CFE_ES_StopPerfCmd_Payload CFE_ES_StopPerfCmd_Payload_t`

Stop Performance Analyzer Command Payload.

For command details, see [CFE\\_ES\\_STOP\\_PERF\\_DATA\\_CC](#)

## 12.127.3 Enumeration Type Documentation

**12.127.3.1 CFE\_ES\_PerfMode** `enum CFE_ES_PerfMode`

Labels for values to use in `CFE_ES_StartPerfCmd_Payload.TriggerMode`.

See also

[CFE\\_ES\\_StartPerfCmd\\_Payload](#)

Enumerator

<code>CFE_ES_PerfTrigger_START</code>	
<code>CFE_ES_PerfTrigger_CENTER</code>	
<code>CFE_ES_PerfTrigger_END</code>	

Definition at line 150 of file default\_cfe\_es\_msgdefs.h.

## 12.128 cfe/modules/es/config/default\_cfe\_es\_msgid\_values.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_es_topicids.h"
```

### Macros

- `#define CFE_PLATFORM_ES_CMD_MIDVAL(x) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_←  
MISSION_ES_##x##_TOPICID)`
- `#define CFE_PLATFORM_ES_TLM_MIDVAL(x) CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION←  
_ES_##x##_TOPICID)`

### 12.128.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Message IDs

### 12.128.2 Macro Definition Documentation

**12.128.2.1 CFE\_PLATFORM\_ES\_CMD\_MIDVAL** #define CFE\_PLATFORM\_ES\_CMD\_MIDVAL(  
  x ) CFE\_PLATFORM\_CMD\_TOPICID\_TO\_MIDV(CFE\_MISSION\_ES\_##x##\_TOPICID)

Definition at line 29 of file default\_cfe\_es\_msgid\_values.h.

**12.128.2.2 CFE\_PLATFORM\_ES\_TLM\_MIDVAL** #define CFE\_PLATFORM\_ES\_TLM\_MIDVAL(  
  x ) CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV(CFE\_MISSION\_ES\_##x##\_TOPICID)

Definition at line 30 of file default\_cfe\_es\_msgid\_values.h.

## 12.129 cfe/modules/es/config/default\_cfe\_es\_msgids.h File Reference

```
#include "cfe_es_msgid_values.h"
```

### Macros

- #define CFE\_ES\_CMD\_MID CFE\_PLATFORM\_ES\_CMD\_MIDVAL(CMD)
- #define CFE\_ES\_SEND\_HK\_MID CFE\_PLATFORM\_ES\_CMD\_MIDVAL(SEND\_HK)
- #define CFE\_ES\_HK\_TLM\_MID CFE\_PLATFORM\_ES\_TLM\_MIDVAL(HK\_TLM)
- #define CFE\_ES\_APP\_TLM\_MID CFE\_PLATFORM\_ES\_TLM\_MIDVAL(APP\_TLM)
- #define CFE\_ES\_MEMSTATS\_TLM\_MID CFE\_PLATFORM\_ES\_TLM\_MIDVAL(MEMSTATS\_TLM)

### 12.129.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Message IDs

### 12.129.2 Macro Definition Documentation

**12.129.2.1 CFE\_ES\_APP\_TLM\_MID** #define CFE\_ES\_APP\_TLM\_MID CFE\_PLATFORM\_ES\_TLM\_MIDVAL(APP\_TLM)  
Definition at line 38 of file default\_cfe\_es\_msgids.h.

**12.129.2.2 CFE\_ES\_CMD\_MID** #define CFE\_ES\_CMD\_MID CFE\_PLATFORM\_ES\_CMD\_MIDVAL(CMD)  
Definition at line 31 of file default\_cfe\_es\_msgids.h.

**12.129.2.3 CFE\_ES\_HK\_TLM\_MID** #define CFE\_ES\_HK\_TLM\_MID CFE\_PLATFORM\_ES\_TLM\_MIDVAL(HK\_TLM)  
Definition at line 37 of file default\_cfe\_es\_msgids.h.

**12.129.2.4 CFE\_ES\_MEMSTATS\_TLM\_MID** #define CFE\_ES\_MEMSTATS\_TLM\_MID CFE\_PLATFORM\_ES\_TLM\_MIDVAL(MEMSTATS\_TLM)

Definition at line 39 of file default\_cfe\_es\_msgids.h.

**12.129.2.5 CFE\_ES\_SEND\_HK\_MID** #define CFE\_ES\_SEND\_HK\_MID CFE\_PLATFORM\_ES\_CMD\_MIDVAL (SEND\_HK)  
Definition at line 32 of file default\_cfe\_es\_msgids.h.

## 12.130 cfe/modules/es/config/default\_cfe\_es\_msgstruct.h File Reference

```
#include "cfe_es_msgdefs.h"
#include "cfe_msg_hdr.h"
#include "cfe_mission_cfg.h"
```

### Data Structures

- struct [CFE\\_ES\\_NoopCmd](#)
- struct [CFE\\_ES\\_ResetCountersCmd](#)
- struct [CFE\\_ES\\_ClearSysLogCmd](#)
- struct [CFE\\_ES\\_ClearERLogCmd](#)
- struct [CFE\\_ES\\_ResetPRCountCmd](#)
- struct [CFE\\_ES\\_SendHkCmd](#)
- struct [CFE\\_ES\\_RestartCmd](#)  
*Restart cFE Command.*
- struct [CFE\\_ES\\_FileNameCmd](#)  
*Generic file name command.*
- struct [CFE\\_ES\\_QueryAllCmd](#)
- struct [CFE\\_ES\\_QueryAllTasksCmd](#)
- struct [CFE\\_ES\\_WriteSysLogCmd](#)
- struct [CFE\\_ES\\_WriteERLogCmd](#)
- struct [CFE\\_ES\\_OverWriteSysLogCmd](#)  
*Overwrite/Discard System Log Configuration Command Payload.*
- struct [CFE\\_ES\\_StartApp](#)  
*Start Application Command.*
- struct [CFE\\_ES\\_StopAppCmd](#)
- struct [CFE\\_ES\\_RestartAppCmd](#)
- struct [CFE\\_ES\\_QueryOneCmd](#)
- struct [CFE\\_ES\\_ReloadAppCmd](#)  
*Reload Application Command.*
- struct [CFE\\_ES\\_SetMaxPRCountCmd](#)  
*Set Maximum Processor Reset Count Command.*
- struct [CFE\\_ES\\_DeleteCDSCmd](#)  
*Delete Critical Data Store Command.*
- struct [CFE\\_ES\\_StartPerfDataCmd](#)  
*Start Performance Analyzer Command.*
- struct [CFE\\_ES\\_StopPerfDataCmd](#)  
*Stop Performance Analyzer Command.*
- struct [CFE\\_ES\\_SetPerfFilterMaskCmd](#)  
*Set Performance Analyzer Filter Mask Command.*
- struct [CFE\\_ES\\_SetPerfTriggerMaskCmd](#)  
*Set Performance Analyzer Trigger Mask Command.*
- struct [CFE\\_ES\\_SendMemPoolStatsCmd](#)  
*Send Memory Pool Statistics Command.*
- struct [CFE\\_ES\\_DumpCDSRegistryCmd](#)

*Dump CDS Registry Command.*

- struct [CFE\\_ES\\_OneAppTlm](#)
- struct [CFE\\_ES\\_MemStatsTlm](#)
- struct [CFE\\_ES\\_HousekeepingTlm](#)

## Typedefs

- typedef struct [CFE\\_ES\\_NoopCmd](#) [CFE\\_ES\\_NoopCmd\\_t](#)
- typedef struct [CFE\\_ES\\_ResetCountersCmd](#) [CFE\\_ES\\_ResetCountersCmd\\_t](#)
- typedef struct [CFE\\_ES\\_ClearSysLogCmd](#) [CFE\\_ES\\_ClearSysLogCmd\\_t](#)
- typedef struct [CFE\\_ES\\_ClearERLogCmd](#) [CFE\\_ES\\_ClearERLogCmd\\_t](#)
- typedef struct [CFE\\_ES\\_ResetPRCountCmd](#) [CFE\\_ES\\_ResetPRCountCmd\\_t](#)
- typedef struct [CFE\\_ES\\_SendHkCmd](#) [CFE\\_ES\\_SendHkCmd\\_t](#)
- typedef struct [CFE\\_ES\\_RestartCmd](#) [CFE\\_ES\\_RestartCmd\\_t](#)

*Restart cFE Command.*

- typedef struct [CFE\\_ES\\_FileNameCmd](#) [CFE\\_ES\\_FileNameCmd\\_t](#)  
*Generic file name command.*
- typedef struct [CFE\\_ES\\_QueryAllCmd](#) [CFE\\_ES\\_QueryAllCmd\\_t](#)
- typedef struct [CFE\\_ES\\_QueryAllTasksCmd](#) [CFE\\_ES\\_QueryAllTasksCmd\\_t](#)
- typedef struct [CFE\\_ES\\_WriteSysLogCmd](#) [CFE\\_ES\\_WriteSysLogCmd\\_t](#)
- typedef struct [CFE\\_ES\\_WriteERLogCmd](#) [CFE\\_ES\\_WriteERLogCmd\\_t](#)
- typedef struct [CFE\\_ES\\_OverWriteSysLogCmd](#) [CFE\\_ES\\_OverWriteSysLogCmd\\_t](#)

*Overwrite/Discard System Log Configuration Command Payload.*

- typedef struct [CFE\\_ES\\_StartApp](#) [CFE\\_ES\\_StartAppCmd\\_t](#)

*Start Application Command.*

- typedef struct [CFE\\_ES\\_StopAppCmd](#) [CFE\\_ES\\_StopAppCmd\\_t](#)
- typedef struct [CFE\\_ES\\_RestartAppCmd](#) [CFE\\_ES\\_RestartAppCmd\\_t](#)
- typedef struct [CFE\\_ES\\_QueryOneCmd](#) [CFE\\_ES\\_QueryOneCmd\\_t](#)
- typedef struct [CFE\\_ES\\_ReloadAppCmd](#) [CFE\\_ES\\_ReloadAppCmd\\_t](#)

*Reload Application Command.*

- typedef struct [CFE\\_ES\\_SetMaxPRCountCmd](#) [CFE\\_ES\\_SetMaxPRCountCmd\\_t](#)

*Set Maximum Processor Reset Count Command.*

- typedef struct [CFE\\_ES\\_DeleteCDSCmd](#) [CFE\\_ES\\_DeleteCDSCmd\\_t](#)

*Delete Critical Data Store Command.*

- typedef struct [CFE\\_ES\\_StartPerfDataCmd](#) [CFE\\_ES\\_StartPerfDataCmd\\_t](#)

*Start Performance Analyzer Command.*

- typedef struct [CFE\\_ES\\_StopPerfDataCmd](#) [CFE\\_ES\\_StopPerfDataCmd\\_t](#)

*Stop Performance Analyzer Command.*

- typedef struct [CFE\\_ES\\_SetPerfFilterMaskCmd](#) [CFE\\_ES\\_SetPerfFilterMaskCmd\\_t](#)

*Set Performance Analyzer Filter Mask Command.*

- typedef struct [CFE\\_ES\\_SetPerfTriggerMaskCmd](#) [CFE\\_ES\\_SetPerfTriggerMaskCmd\\_t](#)

*Set Performance Analyzer Trigger Mask Command.*

- typedef struct [CFE\\_ES\\_SendMemPoolStatsCmd](#) [CFE\\_ES\\_SendMemPoolStatsCmd\\_t](#)

*Send Memory Pool Statistics Command.*

- typedef struct [CFE\\_ES\\_DumpCDSRegistryCmd](#) [CFE\\_ES\\_DumpCDSRegistryCmd\\_t](#)

*Dump CDS Registry Command.*

- typedef struct [CFE\\_ES\\_OneAppTlm](#) [CFE\\_ES\\_OneAppTlm\\_t](#)

- typedef struct [CFE\\_ES\\_MemStatsTlm](#) [CFE\\_ES\\_MemStatsTlm\\_t](#)

- typedef struct [CFE\\_ES\\_HousekeepingTlm](#) [CFE\\_ES\\_HousekeepingTlm\\_t](#)

### 12.130.1 Detailed Description

Purpose: cFE Executive Services (ES) Command and Telemetry packet definition file.

### 12.130.2 Typedef Documentation

**12.130.2.1 CFE\_ES\_ClearERLogCmd\_t** `typedef struct CFE_ES_ClearERLogCmd CFE_ES_ClearERLogCmd_t`

**12.130.2.2 CFE\_ES\_ClearSysLogCmd\_t** `typedef struct CFE_ES_ClearSysLogCmd CFE_ES_ClearSysLogCmd_t`

**12.130.2.3 CFE\_ES\_DeleteCDSCmd\_t** `typedef struct CFE_ES_DeleteCDSCmd CFE_ES_DeleteCDSCmd_t`  
Delete Critical Data Store Command.

**12.130.2.4 CFE\_ES\_DumpCDSRegistryCmd\_t** `typedef struct CFE_ES_DumpCDSRegistryCmd CFE_ES_DumpCDSRegistryCmd_t`  
Dump CDS Registry Command.

**12.130.2.5 CFE\_ES\_FileNameCmd\_t** `typedef struct CFE_ES_FileNameCmd CFE_ES_FileNameCmd_t`  
Generic file name command.

**12.130.2.6 CFE\_ES\_HousekeepingTlm\_t** `typedef struct CFE_ES_HousekeepingTlm CFE_ES_HousekeepingTlm_t`

**Name** Executive Services Housekeeping Packet

**12.130.2.7 CFE\_ES\_MemStatsTlm\_t** `typedef struct CFE_ES_MemStatsTlm CFE_ES_MemStatsTlm_t`

**Name** Memory Pool Statistics Packet

**12.130.2.8 CFE\_ES\_NoopCmd\_t** `typedef struct CFE_ES_NoopCmd CFE_ES_NoopCmd_t`

**12.130.2.9 CFE\_ES\_OneAppTlm\_t** `typedef struct CFE_ES_OneAppTlm CFE_ES_OneAppTlm_t`

**Name** Single Application Information Packet

**12.130.2.10 CFE\_ES\_OverWriteSysLogCmd\_t** `typedef struct CFE_ES_OverWriteSysLogCmd CFE_ES_OverWriteSysLogCmd_t`  
Overwrite/Discard System Log Configuration Command Payload.

**12.130.2.11 CFE\_ES\_QueryAllCmd\_t** `typedef struct CFE_ES_QueryAllCmd CFE_ES_QueryAllCmd_t`

**12.130.2.12 CFE\_ES\_QueryAllTasksCmd\_t** `typedef struct CFE_ES_QueryAllTasksCmd CFE_ES_QueryAllTasksCmd_t`

**12.130.2.13 CFE\_ES\_QueryOneCmd\_t** `typedef struct CFE_ES_QueryOneCmd CFE_ES_QueryOneCmd_t`

**12.130.2.14 CFE\_ES\_ReloadAppCmd\_t** `typedef struct CFE_ES_ReloadAppCmd CFE_ES_ReloadAppCmd_t`  
Reload Application Command.

**12.130.2.15 CFE\_ES\_ResetCountersCmd\_t** `typedef struct CFE_ES_ResetCountersCmd CFE_ES_ResetCountersCmd_t`

**12.130.2.16 CFE\_ES\_ResetPRCountCmd\_t** `typedef struct CFE_ES_ResetPRCountCmd CFE_ES_ResetPRCountCmd_t`

**12.130.2.17 CFE\_ES\_RestartAppCmd\_t** `typedef struct CFE_ES_RestartAppCmd CFE_ES_RestartAppCmd_t`

**12.130.2.18 CFE\_ES\_RestartCmd\_t** `typedef struct CFE_ES_RestartCmd CFE_ES_RestartCmd_t`  
Restart cFE Command.

**12.130.2.19 CFE\_ES\_SendHkCmd\_t** `typedef struct CFE_ES_SendHkCmd CFE_ES_SendHkCmd_t`

**12.130.2.20 CFE\_ES\_SendMemPoolStatsCmd\_t** `typedef struct CFE_ES_SendMemPoolStatsCmd CFE_ES_SendMemPoolStatsCmd_t`  
Send Memory Pool Statistics Command.

**12.130.2.21 CFE\_ES\_SetMaxPRCountCmd\_t** `typedef struct CFE_ES_SetMaxPRCountCmd CFE_ES_SetMaxPRCountCmd_t`  
Set Maximum Processor Reset Count Command.

**12.130.2.22 CFE\_ES\_SetPerfFilterMaskCmd\_t** `typedef struct CFE_ES_SetPerfFilterMaskCmd CFE_ES_SetPerfFilterMaskCmd_t`  
Set Performance Analyzer Filter Mask Command.

**12.130.2.23 CFE\_ES\_SetPerfTriggerMaskCmd\_t** `typedef struct CFE_ES_SetPerfTriggerMaskCmd CFE_ES_SetPerfTriggerMaskCmd_t`  
Set Performance Analyzer Trigger Mask Command.

**12.130.2.24 CFE\_ES\_StartAppCmd\_t** `typedef struct CFE_ES_StartApp CFE_ES_StartAppCmd_t`  
Start Application Command.

**12.130.2.25 CFE\_ES\_StartPerfDataCmd\_t** `typedef struct CFE_ES_StartPerfDataCmd CFE_ES_StartPerfDataCmd_t`  
Start Performance Analyzer Command.

**12.130.2.26 CFE\_ES\_StopAppCmd\_t** `typedef struct CFE_ES_StopAppCmd CFE_ES_StopAppCmd_t`

**12.130.2.27 CFE\_ES\_StopPerfDataCmd\_t** `typedef struct CFE_ES_StopPerfDataCmd CFE_ES_StopPerfDataCmd_t`  
Stop Performance Analyzer Command.

**12.130.2.28 CFE\_ES\_WriteERLogCmd\_t** `typedef struct CFE_ES_WriteERLogCmd CFE_ES_WriteERLogCmd_t`

**12.130.2.29 CFE\_ES\_WriteSysLogCmd\_t** `typedef struct CFE_ES_WriteSysLogCmd CFE_ES_WriteSysLogCmd_t`

## 12.131 cfe/modules/es/config/default\_cfe\_es\_platform\_cfg.h File Reference

```
#include "cfe_es_mission_cfg.h"
#include "cfe_es_internal_cfg.h"
```

### 12.131.1 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.

## 12.132 cfe/modules/es/config/default\_cfe\_es\_topicid\_values.h File Reference

### Macros

- `#define CFE_MISSION_ES_TIDVAL(x) DEFAULT_CFE_MISSION_ES_##x##_TOPICID`

### 12.132.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Topic IDs

### 12.132.2 Macro Definition Documentation

**12.132.2.1 CFE\_MISSION\_ES\_TIDVAL** `#define CFE_MISSION_ES_TIDVAL(
 x ) DEFAULT_CFE_MISSION_ES_##x##_TOPICID`

Definition at line 26 of file default\_cfe\_es\_topicid\_values.h.

## 12.133 cfe/modules/es/fsw/inc/cfe\_es\_eventids.h File Reference

### Macros

#### ES event IDs

- `#define CFE_ES_INIT_INF_EID 1`

- #define **CFE\_ES\_INITSTATS\_INF\_EID** 2  
*ES Initialization Statistics Information Event ID.*
- #define **CFE\_ES\_NOOP\_INF\_EID** 3  
*ES No-op Command Success Event ID.*
- #define **CFE\_ES\_RESET\_INF\_EID** 4  
*ES Reset Counters Command Success Event ID.*
- #define **CFE\_ES\_START\_INF\_EID** 6  
*ES Start Application Command Success Event ID.*
- #define **CFE\_ES\_STOP\_DBG\_EID** 7  
*ES Stop Application Command Request Success Event ID.*
- #define **CFE\_ES\_STOP\_INF\_EID** 8  
*ES Stop Application Completed Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_DBG\_EID** 9  
*ES Restart Application Command Request Success Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_INF\_EID** 10  
*ES Restart Application Completed Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_DBG\_EID** 11  
*ES Reload Application Command Request Success Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_INF\_EID** 12  
*ES Reload Application Complete Event ID.*
- #define **CFE\_ES\_EXIT\_APP\_INF\_EID** 13  
*ES Nominal Exit Application Complete Event ID.*
- #define **CFE\_ES\_ERREXIT\_APP\_INF\_EID** 14  
*ES Error Exit Application Complete Event ID.*
- #define **CFE\_ES\_ONE\_APP\_EID** 15  
*ES Query One Application Command Success Event ID.*
- #define **CFE\_ES\_ALL\_APPS\_EID** 16  
*ES Query All Applications Command Success Event ID.*
- #define **CFE\_ES\_SYSLOG1\_INF\_EID** 17  
*ES Clear System Log Command Success Event ID.*
- #define **CFE\_ES\_SYSLOG2\_EID** 18  
*ES Write System Log Command Success Event ID.*
- #define **CFE\_ES\_ERLOG1\_INF\_EID** 19  
*ES Clear Exception Reset Log Command Success Event ID.*
- #define **CFE\_ES\_ERLOG2\_EID** 20  
*ES Write Exception Reset Log Complete Event ID.*
- #define **CFE\_ES\_MID\_ERR\_EID** 21  
*ES Invalid Message ID Received Event ID.*
- #define **CFE\_ES\_CC1\_ERR\_EID** 22  
*ES Invalid Command Code Received Event ID.*
- #define **CFE\_ES\_LEN\_ERR\_EID** 23  
*ES Invalid Command Length Event ID.*
- #define **CFE\_ES\_BOOT\_ERR\_EID** 24  
*ES Restart Command Invalid Restart Type Event ID.*
- #define **CFE\_ES\_START\_ERR\_EID** 26  
*ES Start Application Command Application Creation Failed Event ID.*
- #define **CFE\_ES\_START\_INVALID\_FILENAME\_ERR\_EID** 27  
*ES Start Application Command Invalid Filename Event ID.*
- #define **CFE\_ES\_START\_INVALID\_ENTRY\_POINT\_ERR\_EID** 28  
*ES Start Application Command Entry Point NULL Event ID.*
- #define **CFE\_ES\_START\_NULL\_APP\_NAME\_ERR\_EID** 29  
*ES Start Application Command App Name NULL Event ID.*
- #define **CFE\_ES\_START\_PRIORITY\_ERR\_EID** 31  
*ES Start Application Command Priority Too Large Event ID.*

- #define **CFE\_ES\_START\_EXC\_ACTION\_ERR\_EID** 32  
*ES Start Application Command Exception Action Invalid Event ID.*
- #define **CFE\_ES\_ERREXIT\_APP\_ERR\_EID** 33  
*ES Error Exit Application Cleanup Failed Event ID.*
- #define **CFE\_ES\_STOP\_ERR1\_EID** 35  
*ES Stop Application Command Request Failed Event ID.*
- #define **CFE\_ES\_STOP\_ERR2\_EID** 36  
*ES Stop Application Command Get AppID By Name Failed Event ID.*
- #define **CFE\_ES\_STOP\_ERR3\_EID** 37  
*ES Stop Application Cleanup Failed Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_ERR1\_EID** 38  
*ES Restart Application Command Request Failed Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_ERR2\_EID** 39  
*ES Restart Application Command Get AppID By Name Failed Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_ERR3\_EID** 40  
*ES Restart Application Startup Failed Event ID.*
- #define **CFE\_ES\_RESTART\_APP\_ERR4\_EID** 41  
*ES Restart Application Cleanup Failed Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_ERR1\_EID** 42  
*ES Reload Application Command Request Failed Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_ERR2\_EID** 43  
*ES Reload Application Command Get AppID By Name Failed Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_ERR3\_EID** 44  
*ES Reload Application Startup Failed Event ID.*
- #define **CFE\_ES\_RELOAD\_APP\_ERR4\_EID** 45  
*ES Reload Application Cleanup Failed Event ID.*
- #define **CFE\_ES\_EXIT\_APP\_ERR\_EID** 46  
*ES Exit Application Cleanup Failed Event ID.*
- #define **CFE\_ES\_PCR\_ERR1\_EID** 47  
*ES Process Control Invalid Exception State Event ID.*
- #define **CFE\_ES\_PCR\_ERR2\_EID** 48  
*ES Process Control Unknown State Event ID.*
- #define **CFE\_ES\_ONE\_ERR\_EID** 49  
*ES Query One Application Data Command Transmit Message Failed Event ID.*
- #define **CFE\_ES\_ONE\_APPID\_ERR\_EID** 50  
*ES Query One Application Data Command Get AppID By Name Failed Event ID.*
- #define **CFE\_ES\_OSCREATE\_ERR\_EID** 51  
*ES Query All Application Data Command File Creation Failed Event ID.*
- #define **CFE\_ES\_WRHDR\_ERR\_EID** 52  
*ES Query All Application Data Command File Write Header Failed Event ID.*
- #define **CFE\_ES\_TASKWR\_ERR\_EID** 53  
*ES Query All Application Data Command File Write App Data Failed Event ID.*
- #define **CFE\_ES\_SYSLOG2\_ERR\_EID** 55  
*ES Write System Log Command Filename Parse or File Creation Failed Event ID.*
- #define **CFE\_ES\_ERLOG2\_ERR\_EID** 56  
*ES Write Exception Reset Log Command Request or File Creation Failed Event ID.*
- #define **CFE\_ES\_PERF\_STARTCMD\_EID** 57  
*ES Start Performance Analyzer Data Collection Command Success Event ID.*
- #define **CFE\_ES\_PERF\_STARTCMD\_ERR\_EID** 58  
*ES Start Performance Analyzer Data Collection Command Idle Check Failed Event ID.*
- #define **CFE\_ES\_PERF\_STARTCMD\_TRIG\_ERR\_EID** 59  
*ES Start Performance Analyzer Data Collection Command Invalid Trigger Event ID.*
- #define **CFE\_ES\_PERF\_STOPCMD\_EID** 60  
*ES Stop Performance Analyzer Data Collection Command Request Success Event ID.*
- #define **CFE\_ES\_PERF\_STOPCMD\_ERR2\_EID** 62

- `#define CFE_ES_PERF_FILTMSKCMD_EID 63`  
*ES Stop Performance Analyzer Data Collection Command Request Idle Check Failed Event ID.*
- `#define CFE_ES_PERF_FILTMSKERR_EID 64`  
*ES Set Performance Analyzer Filter Mask Command Success Event ID.*
- `#define CFE_ES_PERF_TRIGMSKCMD_EID 65`  
*ES Set Performance Analyzer Trigger Mask Command Invalid Index Event ID.*
- `#define CFE_ES_PERF_TRIGMSKERR_EID 66`  
*ES Set Performance Analyzer Trigger Mask Command Success Event ID.*
- `#define CFE_ES_PERF_LOG_ERR_EID 67`  
*ES Set Performance Analyzer Trigger Mask Command Invalid Mask Event ID.*
- `#define CFE_ES_PERF_DATAWRITTEN_EID 68`  
*ES Stop Performance Analyzer Data Collection Command Filename Parse or File Create Failed Event ID.*
- `#define CFE_ES_CDS_REGISTER_ERR_EID 69`  
*Performance Log Write Success Event ID.*
- `#define CFE_ES_SYSLOGMODE_EID 70`  
*ES Register CDS API Failed Event ID.*
- `#define CFE_ES_ERR_SYSLOGMODE_EID 71`  
*ES Set System Log Overwrite Mode Command Success 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 Filenamne 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 Filenamne 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.*

### 12.133.1 Detailed Description

cFE Executive Services Event IDs

### 12.133.2 Macro Definition Documentation

**12.133.2.1 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.

**12.133.2.2 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.

**12.133.2.3 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.

**12.133.2.4 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.

**12.133.2.5 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.

**12.133.2.6 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.

**12.133.2.7 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.

**12.133.2.8 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.

**12.133.2.9 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.

**12.133.2.10 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.

**12.133.2.11 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.

**12.133.2.12 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.

**12.133.2.13 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.

**12.133.2.14 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.

**12.133.2.15 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.

**12.133.2.16 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.

**12.133.2.17 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.

**12.133.2.18 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.

**12.133.2.19 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.

**12.133.2.20 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.

**12.133.2.21 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.

**12.133.2.22 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.

**12.133.2.23 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.

**12.133.2.24 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.

**12.133.2.25 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.

**12.133.2.26 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.

**12.133.2.27 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.

**12.133.2.28 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.

**12.133.2.29 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.

**12.133.2.30 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.

**12.133.2.31 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.

**12.133.2.32 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.

**12.133.2.33 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.

**12.133.2.34 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.

**12.133.2.35 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.

**12.133.2.36 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.

**12.133.2.37 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.

**12.133.2.38 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.

**12.133.2.39 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.

**12.133.2.40 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.

**12.133.2.41 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.

**12.133.2.42 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.

**12.133.2.43 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.

**12.133.2.44 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.

**12.133.2.45 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.

**12.133.2.46 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.

**12.133.2.47 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.

**12.133.2.48 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.

**12.133.2.49 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.

**12.133.2.50 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.

**12.133.2.51 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.

**12.133.2.52 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.

**12.133.2.53 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.

**12.133.2.54 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.

**12.133.2.55 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.

**12.133.2.56 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.

**12.133.2.57 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.

**12.133.2.58 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.

**12.133.2.59 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.

**12.133.2.60 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.

**12.133.2.61 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.

**12.133.2.62 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.

**12.133.2.63 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.

**12.133.2.64 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.

**12.133.2.65 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.

**12.133.2.66 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.

**12.133.2.67 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.

**12.133.2.68 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.

**12.133.2.69 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.

**12.133.2.70 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.

**12.133.2.71 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.

**12.133.2.72 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.

**12.133.2.73 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.

**12.133.2.74 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.

**12.133.2.75 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.

**12.133.2.76 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.

**12.133.2.77 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.

**12.133.2.78 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.

**12.133.2.79 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.

**12.133.2.80 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.

**12.133.2.81 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.

**12.133.2.82 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.

**12.133.2.83 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.

**12.133.2.84 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.

**12.133.2.85 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.

**12.133.2.86 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.

## 12.134 cfe/modules/es/fsw/inc/cfe\_es\_fcncodes.h File Reference

```
#include "cfe_es_fcncode_values.h"
```

### Macros

#### Executive Services Command Codes

- #define CFE\_ES\_NOOP\_CC CFE\_ES\_CCVAL(NOOP)
- #define CFE\_ES\_RESET\_COUNTERS\_CC CFE\_ES\_CCVAL(RESET\_COUNTERS)
- #define CFE\_ES\_RESTART\_CC CFE\_ES\_CCVAL(RESTART)
- #define CFE\_ES\_START\_APP\_CC CFE\_ES\_CCVAL(START\_APP)
- #define CFE\_ES\_STOP\_APP\_CC CFE\_ES\_CCVAL(STOP\_APP)
- #define CFE\_ES\_RESTART\_APP\_CC CFE\_ES\_CCVAL(RESTART\_APP)
- #define CFE\_ES\_RELOAD\_APP\_CC CFE\_ES\_CCVAL(RELOAD\_APP)
- #define CFE\_ES\_QUERY\_ONE\_CC CFE\_ES\_CCVAL(QUERY\_ONE)
- #define CFE\_ES\_QUERY\_ALL\_CC CFE\_ES\_CCVAL(QUERY\_ALL)
- #define CFE\_ES\_CLEAR\_SYS\_LOG\_CC CFE\_ES\_CCVAL(CLEAR\_SYS\_LOG)
- #define CFE\_ES\_WRITE\_SYS\_LOG\_CC CFE\_ES\_CCVAL(WRITE\_SYS\_LOG)
- #define CFE\_ES\_CLEAR\_ER\_LOG\_CC CFE\_ES\_CCVAL(CLEAR\_ER\_LOG)
- #define CFE\_ES\_WRITE\_ER\_LOG\_CC CFE\_ES\_CCVAL(WRITE\_ER\_LOG)
- #define CFE\_ES\_START\_PERF\_DATA\_CC CFE\_ES\_CCVAL(START\_PERF\_DATA)
- #define CFE\_ES\_STOP\_PERF\_DATA\_CC CFE\_ES\_CCVAL(STOP\_PERF\_DATA)
- #define CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC CFE\_ES\_CCVAL(SET\_PERF\_FILTER\_MASK)
- #define CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC CFE\_ES\_CCVAL(SET\_PERF\_TRIGGER\_MASK)
- #define CFE\_ES\_OVER\_WRITE\_SYS\_LOG\_CC CFE\_ES\_CCVAL(OVER\_WRITE\_SYS\_LOG)
- #define CFE\_ES\_RESET\_PR\_COUNT\_CC CFE\_ES\_CCVAL(RESET\_PR\_COUNT)
- #define CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC CFE\_ES\_CCVAL(SET\_MAX\_PR\_COUNT)
- #define CFE\_ES\_DELETE\_CDS\_CC CFE\_ES\_CCVAL(DELETE\_CDS)
- #define CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC CFE\_ES\_CCVAL(SEND\_MEM\_POOL\_STATS)
- #define CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC CFE\_ES\_CCVAL(DUMP\_CDS\_REGISTRY)
- #define CFE\_ES\_QUERY\_ALL\_TASKS\_CC CFE\_ES\_CCVAL(QUERY\_ALL\_TASKS)

### 12.134.1 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.

### 12.134.2 Macro Definition Documentation

#### 12.134.2.1 CFE\_ES\_CLEAR\_ER\_LOG\_CC #define CFE\_ES\_CLEAR\_ER\_LOG\_CC CFE\_ES\_CCVAL(CLEAR\_ER\_LOG)

**Name** Clears the contents of the Exception and Reset Log

#### Description

This command causes the contents of the Executive Services Exception and Reset Log to be cleared.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_ClearERLog

#### Command Structure

[CFE\\_ES\\_ClearERLogCmd\\_t](#)

#### Command Verification

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

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- The [CFE\\_ES\\_ERLOG1\\_INF\\_EID](#) informational event message will be generated.
- `$sc_$cpu_ES_ERLOGINDEX` - Index into Exception Reset Log goes to zero

#### Error Conditions

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

#### Criticality

This command is not dangerous. However, any previously logged data will be lost.

#### See also

[CFE\\_ES\\_CLEAR\\_SYS\\_LOG\\_CC](#), [CFE\\_ES\\_WRITE\\_SYS\\_LOG\\_CC](#), [CFE\\_ES\\_WRITE\\_ER\\_LOG\\_CC](#)

Definition at line 542 of file `cfe_es_fcncodes.h`.

### 12.134.2.2 CFE\_ES\_CLEAR\_SYS\_LOG\_CC #define CFE\_ES\_CLEAR\_SYS\_LOG\_CC [CFE\\_ES\\_CCVAL](#)(CLEAR\_SYS\_LOG)

**Name** Clear Executive Services System Log

#### Description

This command clears the contents of the Executive Services System Log.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_ClearSysLog

#### Command Structure

[CFE\\_ES\\_ClearSysLogCmd\\_t](#)

#### Command Verification

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

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- The [CFE\\_ES\\_SYSLOG1\\_INF\\_EID](#) informational event message will be generated.
- `$sc_$cpu_ES_SYSLOGBYTEUSED` - System Log Bytes Used will go to zero
- `$sc_$cpu_ES_SYSLOGENTRIES` - Number of System Log Entries will go to zero

### Error Conditions

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

### Criticality

This command is not dangerous. However, any previously logged data will be lost.

### See also

[CFE\\_ES\\_WRITE\\_SYS\\_LOG\\_CC](#), [CFE\\_ES\\_CLEAR\\_ER\\_LOG\\_CC](#), [CFE\\_ES\\_WRITE\\_ER\\_LOG\\_CC](#), [CFE\\_ES\\_OVER\\_WRITE\\_SYS](#)

Definition at line 465 of file cfe\_es\_fcncodes.h.

### 12.134.2.3 CFE\_ES\_DELETE\_CDS\_CC #define CFE\_ES\_DELETE\_CDS\_CC CFE\_ES\_CCVAL(DELETE\_CDS)

**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 911 of file cfe\_es\_fcncodes.h.

**12.134.2.4 CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC** #define CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC CFE\_ES\_CCVAL (DUMP←\_CDS\_REGISTRY)

**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 992 of file cfe\_es\_fcncodes.h.

**12.134.2.5 CFE\_ES\_NOOP\_CC** #define CFE\_ES\_NOOP\_CC CFE\_ES\_CCVAL (NOOP)

**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 75 of file [cfe\\_es\\_fncodes.h](#).

**12.134.2.6 CFE\_ES\_OVER\_WRITE\_SYS\_LOG\_CC** #define CFE\_ES\_OVER\_WRITE\_SYS\_LOG\_CC [CFE\\_ES\\_CCVAL](#) (OVER←\_WRITE\_SYS\_LOG)

**Name** Set Executive Services System Log Mode to Discard/Overwrite

### Description

This command allows the user to configure the Executive Services to either discard new System Log messages when it is full or to overwrite the oldest messages.

## Command Mnemonic(s) \$sc\_\$cpu\_ES\_OverwriteSysLogMode

### Command Structure

[CFE\\_ES\\_OverWriteSysLogCmd\\_t](#)

### Command Verification

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

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- `$sc_$cpu_ES_SYSLOGMODE` - Current System Log Mode should reflect the commanded value
- The [CFE\\_ES\\_SYSLOGMODE\\_EID](#) debug event message will be generated.

### Error Conditions

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

- The desired mode is neither [CFE\\_ES\\_LogMode\\_OVERWRITE](#) or [CFE\\_ES\\_LogMode\\_DISCARD](#)

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_ES_CMDEC` - command error counter will increment
- A command specific error event message is issued for all error cases

### Criticality

None. (It should be noted that "Overwrite" mode would allow a message identifying the cause of a problem to be lost by a subsequent flood of additional messages).

### See also

[CFE\\_ES\\_CLEAR\\_SYS\\_LOG\\_CC](#), [CFE\\_ES\\_WRITE\\_SYS\\_LOG\\_CC](#)

Definition at line 794 of file cfe\_es\_fcncodes.h.

## 12.134.2.7 CFE\_ES\_QUERY\_ALL\_CC #define CFE\_ES\_QUERY\_ALL\_CC [CFE\\_ES\\_CCVAL](#)(QUERY\_ALL)

**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 430 of file cfe\_es\_fncodes.h.

**12.134.2.8 CFE\_ES\_QUERY\_ALL\_TASKS\_CC** #define CFE\_ES\_QUERY\_ALL\_TASKS\_CC CFE\_ES\_CCVAL (QUERY\_↔  
ALL\_TASKS)

**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 1034 of file `cfe_es_fcncodes.h`.

#### 12.134.2.9 CFE\_ES\_QUERY\_ONE\_CC #define CFE\_ES\_QUERY\_ONE\_CC CFE\_ES\_CCVAL(QUERY\_ONE)

**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 388 of file `cfe_es_fcncodes.h`.

**12.134.2.10 CFE\_ES\_RELOAD\_APP\_CC** #define CFE\_ES\_RELOAD\_APP\_CC CFE\_ES\_CCVAL (RELOAD\_APP)

**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 352 of file cfe\_es\_fcncodes.h.

**12.134.2.11 CFE\_ES\_RESET\_COUNTERS\_CC** #define CFE\_ES\_RESET\_COUNTERS\_CC CFE\_ES\_CCVAL (RESET\_←  
COUNTERS)

**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 112 of file cfe\_es\_fcncodes.h.

**12.134.2.12 CFE\_ES\_RESET\_PR\_COUNT\_CC** #define CFE\_ES\_RESET\_PR\_COUNT\_CC CFE\_ES\_CCVAL (RESET\_←  
PR\_COUNT)

**Name** Resets the Processor Reset Counter to Zero

#### Description

This command allows the user to reset the Processor Reset Counter to zero. The Processor Reset Counter counts the number of Processor Resets that have occurred so as to identify when a Processor Reset should automatically be upgraded to a full Power-On Reset.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_ResetPRCnt

**Command Structure**

[CFE\\_ES\\_ResetPRCountCmd\\_t](#)

**Command Verification**

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

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- `$sc_$cpu_ES_ProcResetCnt` - Current number of processor resets will go to zero
- The [CFE\\_ES\\_RESET\\_PR\\_COUNT\\_EID](#) informational event message will be generated.

**Error Conditions**

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

**Criticality**

This command is not critical. The only impact would be that the system would have to have more processor resets before an automatic power-on reset occurred.

**See also**

[CFE\\_ES\\_SET\\_MAX\\_PR\\_COUNT\\_CC](#), [CFE\\_ES\\_RESET\\_COUNTERS\\_CC](#)

Definition at line 831 of file cfe\_es\_fncodes.h.

**12.134.2.13 CFE\_ES\_RESTART\_APP\_CC #define CFE\_ES\_RESTART\_APP\_CC CFE\_ES\_CCVAL(RESTART\_APP)**

**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 306 of file `cfe_es_fcncodes.h`.

#### 12.134.2.14 CFE\_ES\_RESTART\_CC #define CFE\_ES\_RESTART\_CC CFE\_ES\_CCVAL(RESTART)

**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 164 of file `cfe_es_fcncodes.h`.

**12.134.2.15 CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC** #define CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC CFE\_ES\_CCVAL (SEND←\_MEM\_POOL\_STATS)

**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 950 of file cfe\_es\_fcncodes.h.

**12.134.2.16 CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC** #define CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC CFE\_ES\_CCVAL(SET←\_MAX\_PR\_COUNT)

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

### Description

This command allows the user to specify the number of Processor Resets that are allowed before the next Processor Reset is upgraded to a Power-On Reset.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_SetMaxPRCnT

### Command Structure

CFE\_ES\_SetMaxPRCountCmd\_t

### Command Verification

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

- **\$sc\_\$cpu\_ES\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_ES\_MaxProcResets** - Current maximum number of processor resets before an automatic power-on reset will go to the command specified value.
- The **CFE\_ES\_SET\_MAX\_PR\_COUNT\_EID** informational event message will be generated.

### Error Conditions

There are no error conditions for this command. If the Executive Services receives the command, the event is sent (although it may be filtered by EVS) and the counter is incremented unconditionally.

### Criticality

If the operator were to set the Maximum Processor Reset Count to too high a value, the processor would require an inordinate number of consecutive processor resets before an automatic power-on reset would occur. This could potentially leave the spacecraft without any control for a significant amount of time if a processor reset fails to clear a problem.

### See also

CFE\_ES\_RESET\_PR\_COUNT\_CC

Definition at line 869 of file cfe\_es\_fcncodes.h.

**12.134.2.17 CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC** #define CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC CFE\_ES\_CCVAL (SET←\_PERF\_FILTER\_MASK)

**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 717 of file cfe\_es\_fcncodes.h.

**12.134.2.18 CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC** #define CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC CFE\_ES\_CCVAL (SET←\_PERF\_TRIGGER\_MASK)

**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 754 of file cfe\_es\_fcncodes.h.

### 12.134.2.19 CFE\_ES\_START\_APP\_CC #define CFE\_ES\_START\_APP\_CC CFE\_ES\_CCVAL (START\_APP)

**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](#) (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 207 of file cfe\_es\_fcncodes.h.

**12.134.2.20 CFE\_ES\_START\_PERF\_DATA\_CC** #define CFE\_ES\_START\_PERF\_DATA\_CC [CFE\\_ES\\_CCVAL](#) (START\_←  
PERF\_DATA)

**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 630 of file `cfe_es_fcncodes.h`.

## 12.134.2.21 CFE\_ES\_STOP\_APP\_CC #define CFE\_ES\_STOP\_APP\_CC [CFE\\_ES\\_CCVAL\(STOP\\_APP\)](#)

**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 260 of file `cfe_es_fcncodes.h`.

**12.134.2.22 CFE\_ES\_STOP\_PERF\_DATA\_CC** #define CFE\_ES\_STOP\_PERF\_DATA\_CC [CFE\\_ES\\_CCVAL](#)(STOP\_<br>PERF\_DATA)

**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 680 of file cfe\_es\_fcncodes.h.

## 12.134.2.23 CFE\_ES\_WRITE\_ER\_LOG\_CC #define CFE\_ES\_WRITE\_ER\_LOG\_CC CFE\_ES\_CCVAL(WRITE\_ER\_LOG)

**Name** Writes Exception and Reset Log to a File

### Description

This command causes the contents of the Executive Services Exception and Reset Log to be written to the specified file.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_WriteERLog2File

### Command Structure

[CFE\\_ES\\_WriteERLogCmd\\_t](#)

### Command Verification

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

- `$sc_$cpu_ES_CMDPC` - command execution counter will increment
- The [CFE\\_ES\\_ERLOG2\\_EID](#) debug event message will be generated.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_ER\\_LOG\\_FILE](#) configuration parameter) will be updated with the latest information.

### Error Conditions

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

- A previous request to write the ER log has not yet completed
- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- **\$sc\_\$cpu\_ES\_CMDEC** - command error counter will increment
- A command specific error event message is issued for all error cases

### Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

### See also

[CFE\\_ES\\_CLEAR\\_SYS\\_LOG\\_CC](#), [CFE\\_ES\\_WRITE\\_SYS\\_LOG\\_CC](#), [CFE\\_ES\\_CLEAR\\_ER\\_LOG\\_CC](#)

Definition at line 585 of file cfe\_es\_fcncodes.h.

**12.134.2.24 CFE\_ES\_WRITE\_SYS\_LOG\_CC** #define CFE\_ES\_WRITE\_SYS\_LOG\_CC CFE\_ES\_CCVAL(WRITE\_SYS←\_LOG)

**Name** Writes contents of Executive Services System Log to a File

### Description

This command causes the contents of the Executive Services System Log to be written to a log file.

**Command Mnemonic(s)** \$sc\_\$cpu\_ES\_WriteSysLog2File

### Command Structure

[CFE\\_ES\\_WriteSysLogCmd\\_t](#)

### Command Verification

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

- **\$sc\_\$cpu\_ES\_CMDPC** - command execution counter will increment
- The [CFE\\_ES\\_SYSLOG2\\_EID](#) debug event message will be generated.
- The file specified in the command (or the default specified by the [CFE\\_PLATFORM\\_ES\\_DEFAULT\\_SYSLOG\\_FILE](#) configuration parameter) will be updated with the latest information.

### Error Conditions

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

- The specified FileName cannot be parsed
- An Error occurs while trying to write to the file

Evidence of failure may be found in the following telemetry:

- `$sc_$cpu_ES_CMDEC` - command error counter will increment
- A command specific error event message is issued for all error cases

### Criticality

This command is not inherently dangerous. It will create a new file in the file system (or overwrite an existing one) and could, if performed repeatedly without sufficient file management by the operator, fill the file system.

### See also

[CFE\\_ES\\_CLEAR\\_SYS\\_LOG\\_CC](#), [CFE\\_ES\\_CLEAR\\_ER\\_LOG\\_CC](#), [CFE\\_ES\\_WRITE\\_ER\\_LOG\\_CC](#), [CFE\\_ES\\_OVER\\_WRITE\\_SYS](#)

Definition at line 508 of file `cfe_es_fcncodes.h`.

## 12.135 cfe/modules/es/fsw/inc/cfe\_es\_interface\_cfg.h File Reference

```
#include "cfe_es_interface_cfg_values.h"
```

### Macros

- #define CFE\_MISSION\_ES\_MAX\_APPLICATIONS CFE\_MISSION\_ES\_CFGVAL(MAX\_APPLICATIONS)
- #define DEFAULT\_CFE\_MISSION\_ES\_MAX\_APPLICATIONS 16
- #define CFE\_MISSION\_ES\_PERF\_MAX\_IDS CFE\_MISSION\_ES\_CFGVAL(PERF\_MAX\_IDS)
- #define DEFAULT\_CFE\_MISSION\_ES\_PERF\_MAX\_IDS 128
- #define CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS CFE\_MISSION\_ES\_CFGVAL(POOL\_MAX\_BUCKETS)
- #define DEFAULT\_CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS 17
- #define CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH CFE\_MISSION\_ES\_CFGVAL(CDS\_MAX\_NAME\_LENGTH)
- #define DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH 16
- #define CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_MISSION\_ES\_CFGVAL(DEFAULT\_CRC)
- #define DEFAULT\_CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_ES\_CrcType\_16\_ARC
- #define CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN CFE\_MISSION\_ES\_CFGVAL(CDS\_MAX\_FULL\_NAME\_LEN)
- #define DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN 40

### Checksum/CRC algorithm identifiers

- #define CFE\_MISSION\_ES\_CRC\_8 CFE\_MISSION\_ES\_CFGVAL(CRC\_8)
- #define DEFAULT\_CFE\_MISSION\_ES\_CRC\_8 CFE\_ES\_CrcType\_CRC\_8
- #define CFE\_MISSION\_ES\_CRC\_16 CFE\_MISSION\_ES\_CFGVAL(CRC\_16)
- #define DEFAULT\_CFE\_MISSION\_ES\_CRC\_16 CFE\_ES\_CrcType\_CRC\_16
- #define CFE\_MISSION\_ES\_CRC\_32 CFE\_MISSION\_ES\_CFGVAL(CRC\_32)
- #define DEFAULT\_CFE\_MISSION\_ES\_CRC\_32 CFE\_ES\_CrcType\_CRC\_32

### 12.135.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Mission Configuration Header File

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.135.2 Macro Definition Documentation

**12.135.2.1 CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN** #define CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME←  
\_LEN CFE\_MISSION\_ES\_CFGVAL(CDS\_MAX\_FULL\_NAME\_LEN)

**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 mes-  
sages, 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 146 of file cfe\_es\_interface\_cfg.h.

**12.135.2.2 CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH** #define CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH

**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 110 of file cfe\_es\_interface\_cfg.h.

**12.135.2.3 CFE\_MISSION\_ES\_CRC\_16** #define CFE\_MISSION\_ES\_CRC\_16 CFE\_MISSION\_ES\_CFGVAL(CRC\_16)

Definition at line 158 of file cfe\_es\_interface\_cfg.h.

**12.135.2.4 CFE\_MISSION\_ES\_CRC\_32** #define CFE\_MISSION\_ES\_CRC\_32 CFE\_MISSION\_ES\_CFGVAL(CRC\_32)  
Definition at line 160 of file cfe\_es\_interface\_cfg.h.

**12.135.2.5 CFE\_MISSION\_ES\_CRC\_8** #define CFE\_MISSION\_ES\_CRC\_8 CFE\_MISSION\_ES\_CFGVAL(CRC\_8)  
Definition at line 156 of file cfe\_es\_interface\_cfg.h.

**12.135.2.6 CFE\_MISSION\_ES\_DEFAULT\_CRC** #define CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_MISSION\_ES\_CFGVAL(DEFAULT\_CRC)

**Purpose** Mission Default CRC algorithm

**Description:**

Indicates the which CRC algorithm should be used as the default for verifying the contents of Critical Data Stores and when calculating Table Image data integrity values.

**Limits**

Currently only CFE\_ES\_CrcType\_16\_ARC is supported (see brief in CFE\_ES\_CrcType\_Enum definition in [cfe\\_es\\_api\\_typedefs.h](#))

Definition at line 125 of file cfe\_es\_interface\_cfg.h.

**12.135.2.7 CFE\_MISSION\_ES\_MAX\_APPLICATIONS** #define CFE\_MISSION\_ES\_MAX\_APPLICATIONS CFE\_MISSION\_ES\_CFGVAL(MAX\_APPLICATIONS)

**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 47 of file cfe\_es\_interface\_cfg.h.

**12.135.2.8 CFE\_MISSION\_ES\_PERF\_MAX\_IDS** #define CFE\_MISSION\_ES\_PERF\_MAX\_IDS CFE\_MISSION\_ES\_CFGVAL(PERF\_MAX\_IDS)

**Purpose** Define Max Number of Performance IDs for messages

**Description:**

Defines the maximum number of perf ids allowed.

Each performance id is used to identify something that needs to be measured. Performance ids are limited to the range of 0 to CFE\_MISSION\_ES\_PERF\_MAX\_IDS - 1. Any performance ids outside of this range will be ignored and will be flagged as an error.

This affects the layout of telemetry messages but does not affect run time behavior or internal allocation.

**Limits**

All CPUs within the same SB domain (mission) must share the same definition Note this affects the size of messages, so it must not cause any message to exceed the max length.

Definition at line 70 of file cfe\_es\_interface\_cfg.h.

**12.135.2.9 CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS** #define CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS CFE\_MISSION\_ES\_CFGVAL (P  
\_MAX\_BUCKETS)

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

**Description:**

The upper limit for the number of block sizes supported in the generic pool implementation, which in turn implements the memory pools and CDS. This definition is used as the array size with the pool stats structure, and therefore should be consistent across all CPUs in a mission, as well as with the ground station.

There is also a platform-specific limit which may be fewer than this value.

**Limits:**

Must be at least one. No specific upper limit, but the number is anticipated to be reasonably small (i.e. tens, not hundreds). Large values have not been tested.

Definition at line 92 of file cfe\_es\_interface\_cfg.h.

**12.135.2.10 DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN** #define DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN 40

Definition at line 149 of file cfe\_es\_interface\_cfg.h.

**12.135.2.11 DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH** #define DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH 16

Definition at line 111 of file cfe\_es\_interface\_cfg.h.

**12.135.2.12 DEFAULT\_CFE\_MISSION\_ES\_CRC\_16** #define DEFAULT\_CFE\_MISSION\_ES\_CRC\_16 CFE\_ES\_CrcType\_CRC\_16

Definition at line 159 of file cfe\_es\_interface\_cfg.h.

**12.135.2.13 DEFAULT\_CFE\_MISSION\_ES\_CRC\_32** #define DEFAULT\_CFE\_MISSION\_ES\_CRC\_32 CFE\_ES\_CrcType\_CRC\_32

Definition at line 161 of file cfe\_es\_interface\_cfg.h.

**12.135.2.14 DEFAULT\_CFE\_MISSION\_ES\_CRC\_8** #define DEFAULT\_CFE\_MISSION\_ES\_CRC\_8 CFE\_ES\_CrcType\_CRC\_8

Definition at line 157 of file cfe\_es\_interface\_cfg.h.

**12.135.2.15 DEFAULT\_CFE\_MISSION\_ES\_DEFAULT\_CRC** #define DEFAULT\_CFE\_MISSION\_ES\_DEFAULT\_CRC CFE\_ES\_CrcType\_16\_ARC

Definition at line 126 of file cfe\_es\_interface\_cfg.h.

**12.135.2.16 DEFAULT\_CFE\_MISSION\_ES\_MAX\_APPLICATIONS** #define DEFAULT\_CFE\_MISSION\_ES\_MAX\_APPLICATIONS 16  
Definition at line 48 of file cfe\_es\_interface\_cfg.h.

**12.135.2.17 DEFAULT\_CFE\_MISSION\_ES\_PERF\_MAX\_IDS** #define DEFAULT\_CFE\_MISSION\_ES\_PERF\_MAX\_IDS 128  
Definition at line 71 of file cfe\_es\_interface\_cfg.h.

**12.135.2.18 DEFAULT\_CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS** #define DEFAULT\_CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS 17  
Definition at line 93 of file cfe\_es\_interface\_cfg.h.

## 12.136 cfe/modules/es/fsw/inc/cfe\_es\_internal\_cfg.h File Reference

```
#include "cfe_es_mission_cfg.h"
#include "cfe_es_internal_cfg_values.h"
```

### Macros

- #define CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY CFE\_PLATFORM\_ES\_CFGVAL(START\_TASK\_PRIORITY)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY 68
- #define CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(START\_TASK\_STACK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE 8192
- #define CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING CFE\_PLATFORM\_ES\_CFGVAL(NONVOL\_DISK\_MOUNT\_STRING)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING "/cf"
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_MOUNT\_STRING)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING "/ram"
- #define CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS CFE\_PLATFORM\_ES\_CFGVAL(MAX\_APPLICATIONS)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS 32
- #define CFE\_PLATFORM\_ES\_MAX\_LIBRARIES CFE\_PLATFORM\_ES\_CFGVAL(MAX\_LIBRARIES)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_LIBRARIES 10
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES CFE\_PLATFORM\_ES\_CFGVAL(ER\_LOG\_ENTRIES)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES 20
- #define CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(ER\_LOG\_MAX\_CONTEXT\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE 256
- #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(SYSTEM\_LOG\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072
- #define CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(OBJECT\_TABLE\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE 30
- #define CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS CFE\_PLATFORM\_ES\_CFGVAL(MAX\_GEN\_COUNTERS)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS 8
- #define CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE CFE\_PLATFORM\_ES\_CFGVAL(APP\_SCAN\_RATE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE 1000

- #define CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT CFE\_PLATFORM\_ES\_CFGVAL(APP\_KILL\_TIMEOUT)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT 5
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_SECTOR\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE 512
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_NUM\_SECTORS)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS 4096
- #define CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_PERCENT\_RESERVED)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED 30
- #define CFE\_PLATFORM\_ES\_CDS\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(CDS\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_SIZE (128 \* 1024)
- #define CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(USER\_RESERVED\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE (1024 \* 1024)
- #define CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN CFE\_PLATFORM\_ES\_CFGVAL(MEMPOOL\_ALIGN\_SIZE\_MIN)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN 4
- #define CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE CFE\_PLATFORM\_ES\_CFGVAL(NONVOL\_STARTUP\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE "/cf/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE CFE\_PLATFORM\_ES\_CFGVAL(VOLATILE\_STARTUP\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE "/ram/cfe\_es\_startup.scr"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_APP\_LOG\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE "/ram/cfe\_es\_app\_info.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_TASK\_LOG\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE "/ram/cfe\_es\_taskinfo.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_SYSLOG\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE "/ram/cfe\_es\_syslog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_ER\_LOG\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE "/ram/cfe\_erlog.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_PERF\_DUMP\_FILENAME)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME "/ram/cfe\_es\_perf.dat"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_CDS\_REG\_DUMP\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE "/ram/cfe\_cds\_reg.log"
- #define CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_POR\_SYSLOG\_MODE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE 0
- #define CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_PR\_SYSLOG\_MODE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE 1
- #define CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(PERF\_DATA\_BUFFER\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE 10000

- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE CFE\_PLATFORM\_ES\_CFGVAL(PERF\_FILTMASK\_↔\_NONE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE (0)
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL CFE\_PLATFORM\_ES\_CFGVAL(PERF\_FILTMASK\_ALL)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL (~0)
- #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT CFE\_PLATFORM\_ES\_CFGVAL(PERF\_FILTMASK\_↔\_INIT)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT (~0)
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE CFE\_PLATFORM\_ES\_CFGVAL(PERF\_TRIGMASK\_↔\_NONE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE 0
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL CFE\_PLATFORM\_ES\_CFGVAL(PERF\_TRIGMASK\_↔\_ALL)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL ~0
- #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT CFE\_PLATFORM\_ES\_CFGVAL(PERF\_TRIGMASK\_↔\_INIT)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT 0
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY CFE\_PLATFORM\_ES\_CFGVAL(PERF\_CHILD\_↔\_PRIORITY)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY 200
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(PERF\_CHILD\_↔\_STACK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY CFE\_PLATFORM\_ES\_CFGVAL(PERF\_CHILD\_↔\_MS\_DELAY)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY 20
- #define CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS CFE\_PLATFORM\_ES\_CFGVAL(PERF\_↔\_ENTRIES\_BTWN\_DLYS)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS 50
- #define CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_STACK\_↔\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MAX\_↔\_NUM\_ENTRIES)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES 512
- #define CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS CFE\_PLATFORM\_ES\_CFGVAL(MAX\_↔\_PROCESSOR\_RESETS)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS 2
- #define CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS CFE\_PLATFORM\_ES\_CFGVAL(POOL\_MAX\_↔\_BUCKETS)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS 17
- #define CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS CFE\_PLATFORM\_ES\_CFGVAL(MAX\_MEMORY\_↔\_POOLS)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS 10
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_↔\_01)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_↔\_02)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_↔\_03)

- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03 32
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_04)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_05)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_06)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_07)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_08)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_09)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_10)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_11)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_12)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_13)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_14)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_15)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE←\_16)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(MAX\_BLOCK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM←BLOCK\_SIZE\_01)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM←BLOCK\_SIZE\_02)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM←BLOCK\_SIZE\_03)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 32

- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_04)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 48
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_05)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_06)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_07)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_08)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_09)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_10)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_11)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_12)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_13)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_14)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_15)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_16)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MAX\_BLOCK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE 80000
- #define CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC CFE\_PLATFORM\_ES\_CFGVAL(STARTUP\_SYNC\_POLL\_MSEC)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC 50
- #define CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC CFE\_PLATFORM\_ES\_CFGVAL(STARTUP\_SCRIPT\_TIMEOUT\_MSEC)
- #define DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC 1000

### 12.136.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Platform Configuration Header File

#### 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.

### 12.136.2 Macro Definition Documentation

**12.136.2.1 CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT** `#define CFE_PLATFORM_ES_APP_KILL_TIMEOUT CFE_PLATFORM_ES_CFGVAL(_KILL_TIMEOUT)`

**Purpose** Define ES Application Kill Timeout

#### Description:

ES Application Kill Timeout. This parameter controls the number of "scan periods" that ES will wait for an application to Exit after getting the signal Delete, Reload or Restart. The sequence works as follows:

1. ES will set the control request for an App to Delete/Restart/Reload and set this kill timer to the value in this parameter.
2. If the App is responding and Calls it's RunLoop function, it will drop out of it's main loop and call CFE\_ES→\_ExitApp. Once it calls Exit App, then ES can delete, restart, or reload the app the next time it scans the app table.
3. If the App is not responding, the ES App will decrement this Kill Timeout value each time it runs. If the timeout value reaches zero, ES will kill the app.

The Kill timeout value depends on the [CFE\\_PLATFORM\\_ES\\_APP\\_SCAN\\_RATE](#). If the Scan Rate is 1000, or 1 second, and this [CFE\\_PLATFORM\\_ES\\_APP\\_KILL\\_TIMEOUT](#) is set to 5, then it will take 5 seconds to kill a non-responding App. If the Scan Rate is 250, or 1/4 second, and the [CFE\\_PLATFORM\\_ES\\_APP\\_KILL\\_TIMEOUT](#) is set to 2, then it will take 1/2 second to time out.

#### Limits

There is a lower limit of 1 and an upper limit of 100 on this configuration parameter. Units are number of [CFE\\_PLATFORM\\_ES\\_APP\\_SCAN\\_RATE](#) cycles.

Definition at line 243 of file cfe\_es\_internal\_cfg.h.

**12.136.2.2 CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE** `#define CFE_PLATFORM_ES_APP_SCAN_RATE CFE_PLATFORM_ES_CFGVAL(APP←_SCAN_RATE)`

**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 213 of file cfe\_es\_internal\_cfg.h.

**12.136.2.3 CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MAX\_BLOCK\_SIZE)

Definition at line 851 of file cfe\_es\_internal\_cfg.h.

**12.136.2.4 CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES** #define CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MAX\_NUM\_ENTRIES)

**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 704 of file cfe\_es\_internal\_cfg.h.

**12.136.2.5 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_01)

**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 819 of file cfe\_es\_internal\_cfg.h.

**12.136.2.6 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_02)

Definition at line 821 of file cfe\_es\_internal\_cfg.h.

**12.136.2.7 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_03)

Definition at line 823 of file cfe\_es\_internal\_cfg.h.

**12.136.2.8 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_04)  
Definition at line 825 of file cfe\_es\_internal\_cfg.h.

**12.136.2.9 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_05)  
Definition at line 827 of file cfe\_es\_internal\_cfg.h.

**12.136.2.10 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_06)  
Definition at line 829 of file cfe\_es\_internal\_cfg.h.

**12.136.2.11 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_07)  
Definition at line 831 of file cfe\_es\_internal\_cfg.h.

**12.136.2.12 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_08)  
Definition at line 833 of file cfe\_es\_internal\_cfg.h.

**12.136.2.13 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_09)  
Definition at line 835 of file cfe\_es\_internal\_cfg.h.

**12.136.2.14 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_10)  
Definition at line 837 of file cfe\_es\_internal\_cfg.h.

**12.136.2.15 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_11)  
Definition at line 839 of file cfe\_es\_internal\_cfg.h.

**12.136.2.16 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_12)  
Definition at line 841 of file cfe\_es\_internal\_cfg.h.

**12.136.2.17 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_13)  
Definition at line 843 of file cfe\_es\_internal\_cfg.h.

**12.136.2.18 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK←  
\_SIZE\_14 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_14)

Definition at line 845 of file cfe\_es\_internal\_cfg.h.

**12.136.2.19 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK←  
\_SIZE\_15 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_15)

Definition at line 847 of file cfe\_es\_internal\_cfg.h.

**12.136.2.20 CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK←  
\_SIZE\_16 CFE\_PLATFORM\_ES\_CFGVAL(CDS\_MEM\_BLOCK\_SIZE\_16)

Definition at line 849 of file cfe\_es\_internal\_cfg.h.

**12.136.2.21 CFE\_PLATFORM\_ES\_CDS\_SIZE** #define CFE\_PLATFORM\_ES\_CDS\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(CDS←  
\_SIZE)

**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 324 of file cfe\_es\_internal\_cfg.h.

**12.136.2.22 CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG←  
\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_APP\_LOG\_FILE)

**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 411 of file cfe\_es\_internal\_cfg.h.

**12.136.2.23 CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_CDS\_REG\_DUMP\_FILE)

**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 490 of file cfe\_es\_internal\_cfg.h.

**12.136.2.24 CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_ER\_LOG\_FILE)

**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 460 of file cfe\_es\_internal\_cfg.h.

**12.136.2.25 CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME** #define CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_PERF\_DUMP\_FILENAME)

**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 475 of file cfe\_es\_internal\_cfg.h.

**12.136.2.26 CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE** #define CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_POR\_SYSLOG\_MODE)

**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 509 of file cfe\_es\_internal\_cfg.h.

**12.136.2.27 CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE** #define CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_PR\_SYSLOG\_MODE)

**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 528 of file cfe\_es\_internal\_cfg.h.

**12.136.2.28 CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_STACK\_SIZE)

**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 690 of file cfe\_es\_internal\_cfg.h.

**12.136.2.29 CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_←  
FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_SYSLOG\_FILE)

**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 444 of file cfe\_es\_internal\_cfg.h.

**12.136.2.30 CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE** #define CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_←  
LOG\_FILE CFE\_PLATFORM\_ES\_CFGVAL(DEFAULT\_TASK\_LOG\_FILE)

**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 427 of file cfe\_es\_internal\_cfg.h.

**12.136.2.31 CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES** #define CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES CFE\_PLATFORM\_ES\_CFGVAL(E←  
\_LOG\_ENTRIES)

**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 135 of file cfe\_es\_internal\_cfg.h.

**12.136.2.32 CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE** #define CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(ER\_LOG\_MAX\_CONTEXT\_SIZE)

**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 150 of file cfe\_es\_internal\_cfg.h.

**12.136.2.33 CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS** #define CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS CFE\_PLATFORM\_ES\_CFGVAL(APPLICATIONS)

**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 106 of file cfe\_es\_internal\_cfg.h.

**12.136.2.34 CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(M\_BLOCK\_SIZE)

Definition at line 807 of file cfe\_es\_internal\_cfg.h.

**12.136.2.35 CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS** #define CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS CFE\_PLATFORM\_ES\_CFGVAL(M\_GEN\_COUNTERS)

**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 193 of file cfe\_es\_internal\_cfg.h.

**12.136.2.36 CFE\_PLATFORM\_ES\_MAX\_LIBRARIES** #define CFE\_PLATFORM\_ES\_MAX\_LIBRARIES CFE\_PLATFORM\_ES\_CFGVAL(MAX\_LIBRARIES)

**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 121 of file cfe\_es\_internal\_cfg.h.

**12.136.2.37 CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS** #define CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS CFE\_PLATFORM\_ES\_CFGVAL(MAX\_MEMORY\_POOLS)

**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 755 of file cfe\_es\_internal\_cfg.h.

**12.136.2.38 CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS** #define CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS CFE\_PLATFORM\_ES\_CFGVAL(MAX\_PROCESSOR\_RESETS)

**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 720 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.39 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_01 CFE_PLATFORM_ES_CFGVAL(MEM_BLOCK_SIZE_01)
```

**Purpose** Define Default ES Memory Pool Block Sizes

**Description:**

Default Intermediate ES Memory Pool Block Sizes. If an application is using the CFE\_ES Memory Pool APIs ([CFE\\_ES\\_PoolCreate](#), [CFE\\_ES\\_PoolCreateNoSem](#), [CFE\\_ES\\_GetPoolBuf](#) and [CFE\\_ES\\_PutPoolBuf](#)) but finds these sizes inappropriate for their use, they may wish to use the [CFE\\_ES\\_PoolCreateEx](#) API to specify their own intermediate block sizes

**Limits**

These sizes MUST be increasing and MUST be an integral multiple of 4. Also, CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE must be larger than CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE and both CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE and CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE. Note that if Table Services have been removed from the CFE, the table size limits are still enforced although the table size definitions may be reduced.

Definition at line 775 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.40 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_02 CFE_PLATFORM_ES_CFGVAL(MEM_BLOCK_SIZE_02)
```

Definition at line 777 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.41 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_03 CFE_PLATFORM_ES_CFGVAL(MEM_BLOCK_SIZE_03)
```

Definition at line 779 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.42 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_04 CFE_PLATFORM_ES_CFGVAL(MEM_BLOCK_SIZE_04)
```

Definition at line 781 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.43 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_05 CFE_PLATFORM_ES_CFGVAL(MEM_BLOCK_SIZE_05)
```

Definition at line 783 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.44 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_06 CFE_PLATFORM_ES_CFGVAL(MEM_BLOCK_SIZE_06)
```

Definition at line 785 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.45 CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 #define CFE_PLATFORM_ES_MEM_BLOCK_SIZE_07 CFE_PLATFORM_ES_CFGVAL(MEM_BLOCK_SIZE_07)
```

Definition at line 787 of file cfe\_es\_internal\_cfg.h.

**12.136.2.46 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_08)  
Definition at line 789 of file cfe\_es\_internal\_cfg.h.

**12.136.2.47 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_09)  
Definition at line 791 of file cfe\_es\_internal\_cfg.h.

**12.136.2.48 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_10)  
Definition at line 793 of file cfe\_es\_internal\_cfg.h.

**12.136.2.49 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_11)  
Definition at line 795 of file cfe\_es\_internal\_cfg.h.

**12.136.2.50 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_12)  
Definition at line 797 of file cfe\_es\_internal\_cfg.h.

**12.136.2.51 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_13)  
Definition at line 799 of file cfe\_es\_internal\_cfg.h.

**12.136.2.52 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_14)  
Definition at line 801 of file cfe\_es\_internal\_cfg.h.

**12.136.2.53 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_15)  
Definition at line 803 of file cfe\_es\_internal\_cfg.h.

**12.136.2.54 CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16 CFE\_PLATFORM\_ES\_CFGVAL(MEM\_BLOCK\_SIZE\_16)  
Definition at line 805 of file cfe\_es\_internal\_cfg.h.

**12.136.2.55 CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN** #define CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN CFE\_PLATFORM\_ES\_CFGVAL(MEMPOOL\_ALIGN\_SIZE\_MIN)

**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 365 of file cfe\_es\_internal\_cfg.h.

**12.136.2.56 CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING** #define CFE\_PLATFORM\_ES\_NONVOL\_←  
DISK\_MOUNT\_STRING CFE\_PLATFORM\_ES\_CFGVAL(NONVOL\_DISK\_MOUNT\_STRING)

**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 72 of file cfe\_es\_internal\_cfg.h.

**12.136.2.57 CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE** #define CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_←  
FILE CFE\_PLATFORM\_ES\_CFGVAL(NONVOL\_STARTUP\_FILE)

**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 380 of file cfe\_es\_internal\_cfg.h.

**12.136.2.58 CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE** #define CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE CFE\_PLATFORM\_ES\_CF←  
\_TABLE\_SIZE)

**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 181 of file cfe\_es\_internal\_cfg.h.

**12.136.2.59 CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY** #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY CFE\_PLATFORM\_ES\_CFGVAL(PERF\_CHILD\_MS\_DELAY)

**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 662 of file cfe\_es\_internal\_cfg.h.

**12.136.2.60 CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY** #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY CFE\_PLATFORM\_ES\_CHILD\_PRIORITY)

**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 631 of file cfe\_es\_internal\_cfg.h.

**12.136.2.61 CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(PERF\_CHILD\_STACK\_SIZE)

**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 646 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.62 CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE #define CFE_PLATFORM_ES_PERF_DATA_BUFFER_SIZE CFE_PLATFORM_ES_CFGVAL(PERF_DATA_BUFFER_SIZE)
```

**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 545 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.63 CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS #define CFE_PLATFORM_ES_PERF_ENTRIES_BTWN_DLYS CFE_PLATFORM_ES_CFGVAL(PERF_ENTRIES_BTWN_DLYS)
```

**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 673 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.64 CFE_PLATFORM_ES_PERF_FILTMASK_ALL #define CFE_PLATFORM_ES_PERF_FILTMASK_ALL CFE_PLATFORM_ES_CFGVAL(PERF_FILTMASK_ALL)
```

**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 567 of file cfe\_es\_internal\_cfg.h.

```
12.136.2.65 CFE_PLATFORM_ES_PERF_FILTMASK_INIT #define CFE_PLATFORM_ES_PERF_FILTMASK_INIT CFE_PLATFORM_ES_CFGVAL(PERF_FILTMASK_INIT)
```

**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 579 of file cfe\_es\_internal\_cfg.h.

**12.136.2.66 CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE** #define CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_←  
NONE CFE\_PLATFORM\_ES\_CFGVAL(PERF\_FILTMASK\_NONE)

**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 556 of file cfe\_es\_internal\_cfg.h.

**12.136.2.67 CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL** #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_←  
ALL CFE\_PLATFORM\_ES\_CFGVAL(PERF\_TRIGMASK\_ALL)

**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 603 of file cfe\_es\_internal\_cfg.h.

**12.136.2.68 CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT** #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_←  
INIT CFE\_PLATFORM\_ES\_CFGVAL(PERF\_TRIGMASK\_INIT)

**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 615 of file cfe\_es\_internal\_cfg.h.

**12.136.2.69 CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE** #define CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_←  
NONE CFE\_PLATFORM\_ES\_CFGVAL(PERF\_TRIGMASK\_NONE)

**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 591 of file cfe\_es\_internal\_cfg.h.

**12.136.2.70 CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS** #define CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS CFE\_PLATFORM\_ES\_CFC  
\_MAX\_BUCKETS)

**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 736 of file cfe\_es\_internal\_cfg.h.

**12.136.2.71 CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_MOUNT\_STRING)

**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 89 of file cfe\_es\_internal\_cfg.h.

**12.136.2.72 CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_NUM\_SECTORS)

**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 281 of file cfe\_es\_internal\_cfg.h.

**12.136.2.73 CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_PERCENT\_RESERVED)

**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 306 of file cfe\_es\_internal\_cfg.h.

**12.136.2.74 CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE** #define CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(RAM\_DISK\_SECTOR\_SIZE)

**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 262 of file cfe\_es\_internal\_cfg.h.

**12.136.2.75 CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY CFE\_PLATFORM\_ES\_TASK\_PRIORITY

**Purpose** Define ES Task Priority

**Description:**

Defines the cFE\_ES Task priority.

**Limits**

Not Applicable

Definition at line 43 of file cfe\_es\_internal\_cfg.h.

**12.136.2.76 CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_ES\_START\_TASK\_←  
STACK\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(START\_TASK\_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 cfe\_es\_internal\_cfg.h.

**12.136.2.77 CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_←  
\_SCRIPT\_TIMEOUT\_MSEC CFE\_PLATFORM\_ES\_CFGVAL(STARTUP\_SCRIPT\_TIMEOUT\_MSEC)

**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 893 of file cfe\_es\_internal\_cfg.h.

**12.136.2.78 CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC** #define CFE\_PLATFORM\_ES\_STARTUP\_←  
SYNC\_POLL\_MSEC CFE\_PLATFORM\_ES\_CFGVAL(STARTUP\_SYNC\_POLL\_MSEC)

**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 874 of file cfe\_es\_internal\_cfg.h.

**12.136.2.79 CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE** #define CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(\_LOG\_SIZE)

**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 166 of file `cfe_es_internal_cfg.h`.

**12.136.2.80 CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE** #define CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE CFE\_PLATFORM\_ES\_CFGVAL(USER\_RESERVED\_SIZE)

**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 345 of file `cfe_es_internal_cfg.h`.

**12.136.2.81 CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE** #define CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE CFE\_PLATFORM\_ES\_CFGVAL(VOLATILE\_STARTUP\_FILE)

**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 395 of file `cfe_es_internal_cfg.h`.

**12.136.2.82 DEFAULT\_CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT** #define DEFAULT\_CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT 5

Definition at line 244 of file cfe\_es\_internal\_cfg.h.

**12.136.2.83 DEFAULT\_CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE** #define DEFAULT\_CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE 1000

Definition at line 214 of file cfe\_es\_internal\_cfg.h.

**12.136.2.84 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE 80000

Definition at line 852 of file cfe\_es\_internal\_cfg.h.

**12.136.2.85 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES 512

Definition at line 705 of file cfe\_es\_internal\_cfg.h.

**12.136.2.86 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01 8

Definition at line 820 of file cfe\_es\_internal\_cfg.h.

**12.136.2.87 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02 16

Definition at line 822 of file cfe\_es\_internal\_cfg.h.

**12.136.2.88 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03 32

Definition at line 824 of file cfe\_es\_internal\_cfg.h.

**12.136.2.89 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04 48

Definition at line 826 of file cfe\_es\_internal\_cfg.h.

**12.136.2.90 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05 64

Definition at line 828 of file cfe\_es\_internal\_cfg.h.

**12.136.2.91 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06 96

Definition at line 830 of file cfe\_es\_internal\_cfg.h.

**12.136.2.92 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07 128

Definition at line 832 of file cfe\_es\_internal\_cfg.h.

**12.136.2.93 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08 160

Definition at line 834 of file cfe\_es\_internal\_cfg.h.

**12.136.2.94 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09 256

Definition at line 836 of file cfe\_es\_internal\_cfg.h.

**12.136.2.95 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10 512

Definition at line 838 of file cfe\_es\_internal\_cfg.h.

**12.136.2.96 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11 1024

Definition at line 840 of file cfe\_es\_internal\_cfg.h.

**12.136.2.97 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12 2048

Definition at line 842 of file cfe\_es\_internal\_cfg.h.

**12.136.2.98 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13 4096

Definition at line 844 of file cfe\_es\_internal\_cfg.h.

**12.136.2.99 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14 8192

Definition at line 846 of file cfe\_es\_internal\_cfg.h.

**12.136.2.100 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15 16384

Definition at line 848 of file cfe\_es\_internal\_cfg.h.

**12.136.2.101 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16 32768

Definition at line 850 of file cfe\_es\_internal\_cfg.h.

**12.136.2.102 DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_SIZE (128 \* 1024)

Definition at line 325 of file cfe\_es\_internal\_cfg.h.

**12.136.2.103 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE "/ram/cfe\_es\_app\_info.log"

Definition at line 412 of file cfe\_es\_internal\_cfg.h.

**12.136.2.104 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE "/ram/cfe\_cds\_reg.log"

Definition at line 491 of file cfe\_es\_internal\_cfg.h.

**12.136.2.105 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE "/ram/cfe\_erlog.log"

Definition at line 461 of file cfe\_es\_internal\_cfg.h.

**12.136.2.106 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME "/ram/cfe\_es\_perf.dat"

Definition at line 476 of file cfe\_es\_internal\_cfg.h.

**12.136.2.107 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE 0

Definition at line 510 of file cfe\_es\_internal\_cfg.h.

**12.136.2.108 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE 1

Definition at line 529 of file cfe\_es\_internal\_cfg.h.

**12.136.2.109 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE 8192

Definition at line 691 of file cfe\_es\_internal\_cfg.h.

**12.136.2.110 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE "/ram/cfe\_es\_syslog.log"

Definition at line 445 of file cfe\_es\_internal\_cfg.h.

**12.136.2.111 DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE** #define DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE "/ram/cfe\_es\_taskinfo.log"

Definition at line 428 of file cfe\_es\_internal\_cfg.h.

**12.136.2.112 DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES** #define DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES 20  
Definition at line 136 of file cfe\_es\_internal\_cfg.h.

**12.136.2.113 DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE 256  
Definition at line 151 of file cfe\_es\_internal\_cfg.h.

**12.136.2.114 DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS** #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS 32  
Definition at line 107 of file cfe\_es\_internal\_cfg.h.

**12.136.2.115 DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE 80000  
Definition at line 808 of file cfe\_es\_internal\_cfg.h.

**12.136.2.116 DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS** #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS 8  
Definition at line 194 of file cfe\_es\_internal\_cfg.h.

**12.136.2.117 DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_LIBRARIES** #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_LIBRARIES 10  
Definition at line 122 of file cfe\_es\_internal\_cfg.h.

**12.136.2.118 DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS** #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS 10  
Definition at line 756 of file cfe\_es\_internal\_cfg.h.

**12.136.2.119 DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS** #define DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS 2  
Definition at line 721 of file cfe\_es\_internal\_cfg.h.

**12.136.2.120 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01** #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01 8  
Definition at line 776 of file cfe\_es\_internal\_cfg.h.

**12.136.2.121 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02** #define DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02 16  
Definition at line 778 of file cfe\_es\_internal\_cfg.h.

**12.136.2.122 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_03 32

Definition at line 780 of file cfe\_es\_internal\_cfg.h.

**12.136.2.123 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_04 48

Definition at line 782 of file cfe\_es\_internal\_cfg.h.

**12.136.2.124 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_05 64

Definition at line 784 of file cfe\_es\_internal\_cfg.h.

**12.136.2.125 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_06 96

Definition at line 786 of file cfe\_es\_internal\_cfg.h.

**12.136.2.126 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_07 128

Definition at line 788 of file cfe\_es\_internal\_cfg.h.

**12.136.2.127 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_08 160

Definition at line 790 of file cfe\_es\_internal\_cfg.h.

**12.136.2.128 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_09 256

Definition at line 792 of file cfe\_es\_internal\_cfg.h.

**12.136.2.129 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_10 512

Definition at line 794 of file cfe\_es\_internal\_cfg.h.

**12.136.2.130 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_11 1024

Definition at line 796 of file cfe\_es\_internal\_cfg.h.

**12.136.2.131 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_12 2048

Definition at line 798 of file cfe\_es\_internal\_cfg.h.

**12.136.2.132 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_13 4096  
Definition at line 800 of file cfe\_es\_internal\_cfg.h.

**12.136.2.133 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_14 8192  
Definition at line 802 of file cfe\_es\_internal\_cfg.h.

**12.136.2.134 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_15 16384  
Definition at line 804 of file cfe\_es\_internal\_cfg.h.

**12.136.2.135 DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
MEM\_BLOCK\_SIZE\_16 32768  
Definition at line 806 of file cfe\_es\_internal\_cfg.h.

**12.136.2.136 DEFAULT\_CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN** #define DEFAULT\_CFE\_PLATFORM\_↪  
\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN 4  
Definition at line 366 of file cfe\_es\_internal\_cfg.h.

**12.136.2.137 DEFAULT\_CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING** #define DEFAULT\_CFE\_↪  
PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING "/cf"  
Definition at line 73 of file cfe\_es\_internal\_cfg.h.

**12.136.2.138 DEFAULT\_CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE** #define DEFAULT\_CFE\_PLATFORM\_↪  
ES\_NONVOL\_STARTUP\_FILE "/cf/cfe\_es\_startup.scr"  
Definition at line 381 of file cfe\_es\_internal\_cfg.h.

**12.136.2.139 DEFAULT\_CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
OBJECT\_TABLE\_SIZE 30  
Definition at line 182 of file cfe\_es\_internal\_cfg.h.

**12.136.2.140 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY** #define DEFAULT\_CFE\_PLATFORM\_↪  
ES\_PERF\_CHILD\_MS\_DELAY 20  
Definition at line 663 of file cfe\_es\_internal\_cfg.h.

**12.136.2.141 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_ES\_↪  
\_PERF\_CHILD\_PRIORITY 200  
Definition at line 632 of file cfe\_es\_internal\_cfg.h.

**12.136.2.142 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE 4096

Definition at line 647 of file cfe\_es\_internal\_cfg.h.

**12.136.2.143 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE 10000

Definition at line 546 of file cfe\_es\_internal\_cfg.h.

**12.136.2.144 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS 50

Definition at line 674 of file cfe\_es\_internal\_cfg.h.

**12.136.2.145 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL (~0)

Definition at line 568 of file cfe\_es\_internal\_cfg.h.

**12.136.2.146 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT (~0)

Definition at line 580 of file cfe\_es\_internal\_cfg.h.

**12.136.2.147 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE (0)

Definition at line 557 of file cfe\_es\_internal\_cfg.h.

**12.136.2.148 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL ~0

Definition at line 604 of file cfe\_es\_internal\_cfg.h.

**12.136.2.149 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT 0

Definition at line 616 of file cfe\_es\_internal\_cfg.h.

**12.136.2.150 DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE** #define DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE 0

Definition at line 592 of file cfe\_es\_internal\_cfg.h.

**12.136.2.151 DEFAULT\_CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS** #define DEFAULT\_CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS 17

Definition at line 737 of file cfe\_es\_internal\_cfg.h.

**12.136.2.152 DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING** #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING "/ram"  
Definition at line 90 of file cfe\_es\_internal\_cfg.h.

**12.136.2.153 DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS** #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS 4096  
Definition at line 282 of file cfe\_es\_internal\_cfg.h.

**12.136.2.154 DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED** #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED 30  
Definition at line 307 of file cfe\_es\_internal\_cfg.h.

**12.136.2.155 DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE 512  
Definition at line 263 of file cfe\_es\_internal\_cfg.h.

**12.136.2.156 DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY 68  
Definition at line 44 of file cfe\_es\_internal\_cfg.h.

**12.136.2.157 DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE 8192  
Definition at line 60 of file cfe\_es\_internal\_cfg.h.

**12.136.2.158 DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC** #define DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC 1000  
Definition at line 894 of file cfe\_es\_internal\_cfg.h.

**12.136.2.159 DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC** #define DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC 50  
Definition at line 875 of file cfe\_es\_internal\_cfg.h.

**12.136.2.160 DEFAULT\_CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE 3072  
Definition at line 167 of file cfe\_es\_internal\_cfg.h.

**12.136.2.161 DEFAULT\_CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE (1024 \* 1024)  
Definition at line 346 of file cfe\_es\_internal\_cfg.h.

**12.136.2.162 DEFAULT\_CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE** #define DEFAULT\_CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE "/ram/cfe\_es\_startup.scr"  
Definition at line 396 of file cfe\_es\_internal\_cfg.h.

## 12.137 cfe/modules/es/fsw/inc/cfe\_es\_topicids.h File Reference

```
#include "cfe_es_topicid_values.h"
```

### Macros

- #define CFE\_MISSION\_ES\_CMD\_TOPICID CFE\_MISSION\_ES\_TIDVAL(CMD)
- #define DEFAULT\_CFE\_MISSION\_ES\_CMD\_TOPICID 6
- #define CFE\_MISSION\_ES\_SEND\_HK\_TOPICID CFE\_MISSION\_ES\_TIDVAL(SEND\_HK)
- #define DEFAULT\_CFE\_MISSION\_ES\_SEND\_HK\_TOPICID 8
- #define CFE\_MISSION\_ES\_HK\_TLM\_TOPICID CFE\_MISSION\_ES\_TIDVAL(HK\_TLM)
- #define DEFAULT\_CFE\_MISSION\_ES\_HK\_TLM\_TOPICID 0
- #define CFE\_MISSION\_ES\_APP\_TLM\_TOPICID CFE\_MISSION\_ES\_TIDVAL(APP\_TLM)
- #define DEFAULT\_CFE\_MISSION\_ES\_APP\_TLM\_TOPICID 11
- #define CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID CFE\_MISSION\_ES\_TIDVAL(MEMSTATS\_TLM)
- #define DEFAULT\_CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID 16

### 12.137.1 Detailed Description

CFE Executive Services (CFE\_ES) Application Topic IDs

### 12.137.2 Macro Definition Documentation

**12.137.2.1 CFE\_MISSION\_ES\_APP\_TLM\_TOPICID** #define CFE\_MISSION\_ES\_APP\_TLM\_TOPICID CFE\_MISSION\_ES\_TIDVAL(APP\_TLM)

Definition at line 53 of file cfe\_es\_topicids.h.

**12.137.2.2 CFE\_MISSION\_ES\_CMD\_TOPICID** #define CFE\_MISSION\_ES\_CMD\_TOPICID CFE\_MISSION\_ES\_TIDVAL(CMD)

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE ES command messages

**Limits**

Not Applicable

Definition at line 37 of file cfe\_es\_topicids.h.

**12.137.2.3 CFE\_MISSION\_ES\_HK\_TLM\_TOPICID** #define CFE\_MISSION\_ES\_HK\_TLM\_TOPICID CFE\_MISSION\_ES\_TIDVAL (HK↔\_TLM)

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE ES telemetry messages

**Limits**

Not Applicable

Definition at line 51 of file cfe\_es\_topicids.h.

**12.137.2.4 CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID** #define CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID CFE\_MISSION\_ES\_TIDVAL (MEMSTATS↔\_TLM)

Definition at line 55 of file cfe\_es\_topicids.h.

**12.137.2.5 CFE\_MISSION\_ES\_SEND\_HK\_TOPICID** #define CFE\_MISSION\_ES\_SEND\_HK\_TOPICID CFE\_MISSION\_ES\_TIDVAL (SEND↔\_HK)

Definition at line 39 of file cfe\_es\_topicids.h.

**12.137.2.6 DEFAULT\_CFE\_MISSION\_ES\_APP\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_ES\_APP\_TLM\_TOPICID 11

Definition at line 54 of file cfe\_es\_topicids.h.

**12.137.2.7 DEFAULT\_CFE\_MISSION\_ES\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_ES\_CMD\_TOPICID 6

Definition at line 38 of file cfe\_es\_topicids.h.

**12.137.2.8 DEFAULT\_CFE\_MISSION\_ES\_HK\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_ES\_HK\_TLM\_TOPICID 0

Definition at line 52 of file cfe\_es\_topicids.h.

**12.137.2.9 DEFAULT\_CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID 16

Definition at line 56 of file cfe\_es\_topicids.h.

**12.137.2.10 DEFAULT\_CFE\_MISSION\_ES\_SEND\_HK\_TOPICID** #define DEFAULT\_CFE\_MISSION\_ES\_SEND\_HK\_TOPICID 8

Definition at line 40 of file cfe\_es\_topicids.h.

## 12.138 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.*

### 12.138.1 Detailed Description

Declarations and prototypes for cfe\_evs\_extern\_typedefs module

### 12.138.2 Typedef Documentation

#### 12.138.2.1 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.

#### 12.138.2.2 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.

**12.138.2.3 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.

**12.138.2.4 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.

**12.138.2.5 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.

### 12.138.3 Enumeration Type Documentation

**12.138.3.1 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.

**12.138.3.2 CFE\_EVS\_EventOutput** `enum CFE_EVS_EventOutput`  
Label definitions associated with CFE\_EVS\_EventOutput\_Enum\_t.

Enumerator

<code>CFE_EVS_EventOutput_PORT1</code>	Output Port 1.
<code>CFE_EVS_EventOutput_PORT2</code>	Output Port 2.
<code>CFE_EVS_EventOutput_PORT3</code>	Output Port 3.
<code>CFE_EVS_EventOutput_PORT4</code>	Output Port 4.

Definition at line 130 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.138.3.3 CFE\_EVS\_EventType enum [CFE\\_EVS\\_EventType](#)**

Label definitions associated with CFE\_EVS\_EventType\_Enum\_t.

## Enumerator

<code>CFE_EVS_EventType_DEBUG</code>	Events that are intended only for debugging, not nominal operations.
<code>CFE_EVS_EventType_INFORMATION</code>	Events that identify a state change or action that is not an error.
<code>CFE_EVS_EventType_ERROR</code>	Events that identify an error but are not catastrophic (e.g. - bad command).
<code>CFE_EVS_EventType_CRITICAL</code>	Events that identify errors that are unrecoverable autonomously.

Definition at line 79 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.138.3.4 CFE\_EVS\_LogMode enum [CFE\\_EVS\\_LogMode](#)**

Label definitions associated with CFE\_EVS\_LogMode\_Enum\_t.

## Enumerator

<code>CFE_EVS_LogMode_OVERWRITE</code>	Overwrite Log Mode.
<code>CFE_EVS_LogMode_DISCARD</code>	Discard Log Mode.

Definition at line 56 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.138.3.5 CFE\_EVS\_MsgFormat enum [CFE\\_EVS\\_MsgFormat](#)**

Label definitions associated with CFE\_EVS\_MsgFormat\_Enum\_t.

## Enumerator

<code>CFE_EVS_MsgFormat_SHORT</code>	Short Format Messages.
<code>CFE_EVS_MsgFormat_LONG</code>	Long Format Messages.

Definition at line 33 of file default\_cfe\_evs\_extern\_typedefs.h.

**12.139 cfe/modules/evs/config/default\_cfe\_evs\_fcncode\_values.h File Reference****Macros**

- `#define CFE_EVS_CCVAL(x) CFE_EVS_FunctionCode_##x`

**Enumerations**

- enum `CFE_EVS_FunctionCode_` {
   
`CFE_EVS_FunctionCode_NOOP = 0, CFE_EVS_FunctionCode_RESET_COUNTERS = 1, CFE_EVS_FunctionCode_ENABLE_EVENTS = 2, CFE_EVS_FunctionCode_DISABLE_EVENT_TYPE = 3,`
  
`CFE_EVS_FunctionCode_SET_EVENT_FORMAT_MODE = 4, CFE_EVS_FunctionCode_ENABLE_APP_EVENT_TYPE = 5, CFE_EVS_FunctionCode_DISABLE_APP_EVENT_TYPE = 6, CFE_EVS_FunctionCode_ENABLE_APP_EVENTS = 7,`
  
`CFE_EVS_FunctionCode_DISABLE_APP_EVENTS = 8, CFE_EVS_FunctionCode_RESET_APP_COUNTER = 9, CFE_EVS_FunctionCode_SET_FILTER = 10, CFE_EVS_FunctionCode_ENABLE_PORTS = 11,`
  
`CFE_EVS_FunctionCode_DISABLE_PORTS = 12, CFE_EVS_FunctionCode_RESET_FILTER = 13, CFE_EVS_FunctionCode_RESET_ALL_FILTERS = 14, CFE_EVS_FunctionCode_ADD_EVENT_FILTER = 15,`

```
15 ,
CFE_EVS_FunctionCode_DELETE_EVENT_FILTER = 16 , CFE_EVS_FunctionCode_WRITE_APP_DATA_FILE
= 17 , CFE_EVS_FunctionCode_WRITE_LOG_DATA_FILE = 18 , CFE_EVS_FunctionCode_SET_LOG_MODE
= 19 ,
CFE_EVS_FunctionCode_CLEAR_LOG = 20 }
```

### 12.139.1 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.

### 12.139.2 Macro Definition Documentation

**12.139.2.1 CFE\_EVS\_CCVAL** #define CFE\_EVS\_CCVAL(  
x ) CFE\_EVS\_FunctionCode\_##x

Definition at line 35 of file default\_cfe\_evs\_fcncode\_values.h.

### 12.139.3 Enumeration Type Documentation

#### 12.139.3.1 CFE\_EVS\_FunctionCode\_ enum CFE\_EVS\_FunctionCode\_

##### Enumerator

CFE_EVS_FunctionCode_NOOP	
CFE_EVS_FunctionCode_RESET_COUNTERS	
CFE_EVS_FunctionCode_ENABLE_EVENT_TYPE	
CFE_EVS_FunctionCode_DISABLE_EVENT_TYPE	
CFE_EVS_FunctionCode_SET_EVENT_FORMAT_MODE	
CFE_EVS_FunctionCode_ENABLE_APP_EVENT_TYPE	
CFE_EVS_FunctionCode_DISABLE_APP_EVENT_TYPE	
CFE_EVS_FunctionCode_ENABLE_APP_EVENTS	
CFE_EVS_FunctionCode_DISABLE_APP_EVENTS	
CFE_EVS_FunctionCode_RESET_APP_COUNTER	
CFE_EVS_FunctionCode_SET_FILTER	
CFE_EVS_FunctionCode_ENABLE_PORTS	
CFE_EVS_FunctionCode_DISABLE_PORTS	
CFE_EVS_FunctionCode_RESET_FILTER	
CFE_EVS_FunctionCode_RESET_ALL_FILTERS	
CFE_EVS_FunctionCode_ADD_EVENT_FILTER	
CFE_EVS_FunctionCode_DELETE_EVENT_FILTER	
CFE_EVS_FunctionCode_WRITE_APP_DATA_FILE	
CFE_EVS_FunctionCode_WRITE_LOG_DATA_FILE	
CFE_EVS_FunctionCode_SET_LOG_MODE	
CFE_EVS_FunctionCode_CLEAR_LOG	

Definition at line 37 of file default\_cfe\_evs\_fcncode\_values.h.

## 12.140 cfe/modules/evs/config/default\_cfe\_evs\_interface\_cfg\_values.h File Reference

### Macros

- #define CFE\_MISSION\_EVS\_CFGVAL(x) DEFAULT\_CFE\_MISSION\_EVS\_##x

#### 12.140.1 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.

#### 12.140.2 Macro Definition Documentation

##### 12.140.2.1 CFE\_MISSION\_EVS\_CFGVAL #define CFE\_MISSION\_EVS\_CFGVAL ( x ) DEFAULT\_CFE\_MISSION\_EVS\_##x

Definition at line 36 of file default\_cfe\_evs\_interface\_cfg\_values.h.

## 12.141 cfe/modules/evs/config/default\_cfe\_evs\_internal\_cfg\_values.h File Reference

### Macros

- #define CFE\_PLATFORM\_EVS\_CFGVAL(x) DEFAULT\_CFE\_PLATFORM\_EVS\_##x

#### 12.141.1 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.

#### 12.141.2 Macro Definition Documentation

##### 12.141.2.1 CFE\_PLATFORM\_EVS\_CFGVAL #define CFE\_PLATFORM\_EVS\_CFGVAL ( x ) DEFAULT\_CFE\_PLATFORM\_EVS\_##x

Definition at line 36 of file default\_cfe\_evs\_internal\_cfg\_values.h.

## 12.142 cfe/modules/evs/config/default\_cfe\_evs\_mission\_cfg.h File Reference

```
#include "cfe_evs_interface_cfg.h"
```

### 12.142.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.143 cfe/modules/evs/config/default\_cfe\_evs\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_evs_fcncodes.h"
#include "cfe_evs_msgdefs.h"
#include "cfe_evs_msgstruct.h"
```

### 12.143.1 Detailed Description

Specification for the CFE Event Services (CFE\_EVS) command and telemetry message data types.

This is a compatibility header for the "cfe\_evs\_msg.h" file that has traditionally provided the message definitions for cFS apps.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.144 cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_evs_extern_typedefs.h"
#include "cfe_evs_fcncodes.h"
```

## Data Structures

- struct [CFE\\_EVS\\_LogFileCmd\\_Payload](#)  
*Write Event Log to File Command Payload.*
- struct [CFE\\_EVS\\_AppDataCmd\\_Payload](#)  
*Write Event Services Application Information to File Command Payload.*
- struct [CFE\\_EVS\\_SetLogMode\\_Payload](#)  
*Set Log Mode Command Payload.*
- struct [CFE\\_EVS\\_SetEventFormatCode\\_Payload](#)  
*Set Event Format Mode Command Payload.*
- struct [CFE\\_EVS\\_BitMaskCmd\\_Payload](#)  
*Generic Bitmask Command Payload.*
- struct [CFE\\_EVS\\_AppNameCmd\\_Payload](#)  
*Generic App Name Command Payload.*
- struct [CFE\\_EVS\\_AppNameEventIDCmd\\_Payload](#)  
*Generic App Name and Event ID Command Payload.*

- struct [CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload](#)  
*Generic App Name and Bitmask Command Payload.*
- struct [CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload](#)  
*Generic App Name, Event ID, Mask Command Payload.*
- struct [CFE\\_EVS\\_AppTlmData](#)
- struct [CFE\\_EVS\\_HousekeepingTlm\\_Payload](#)
- struct [CFE\\_EVS\\_PacketID](#)
- struct [CFE\\_EVS\\_LongEventTlm\\_Payload](#)
- struct [CFE\\_EVS\\_ShortEventTlm\\_Payload](#)

## Macros

- #define [CFE\\_EVS\\_DEBUG\\_BIT](#) (1 << (CFE\_EVS\_EventType\_DEBUG - 1))
- #define [CFE\\_EVS\\_INFORMATION\\_BIT](#) (1 << (CFE\_EVS\_EventType\_INFORMATION - 1))
- #define [CFE\\_EVS\\_ERROR\\_BIT](#) (1 << (CFE\_EVS\_EventType\_ERROR - 1))
- #define [CFE\\_EVS\\_CRITICAL\\_BIT](#) (1 << (CFE\_EVS\_EventType\_CRITICAL - 1))
- #define [CFE\\_EVS\\_PORT1\\_BIT](#) (1 << (CFE\_EVS\_EventOutput\_PORT1 - 1))
- #define [CFE\\_EVS\\_PORT2\\_BIT](#) (1 << (CFE\_EVS\_EventOutput\_PORT2 - 1))
- #define [CFE\\_EVS\\_PORT3\\_BIT](#) (1 << (CFE\_EVS\_EventOutput\_PORT3 - 1))
- #define [CFE\\_EVS\\_PORT4\\_BIT](#) (1 << (CFE\_EVS\_EventOutput\_PORT4 - 1))

## TypeDefs

- typedef struct [CFE\\_EVS\\_LogFileCmd\\_Payload](#) CFE\_EVS\_LogFileCmd\_Payload\_t  
*Write Event Log to File Command Payload.*
- typedef struct [CFE\\_EVS\\_AppDataCmd\\_Payload](#) CFE\_EVS\_AppDataCmd\_Payload\_t  
*Write Event Services Application Information to File Command Payload.*
- typedef struct [CFE\\_EVS\\_SetLogMode\\_Payload](#) CFE\_EVS\_SetLogMode\_Payload\_t  
*Set Log Mode Command Payload.*
- typedef struct [CFE\\_EVS\\_SetEventFormatCode\\_Payload](#) CFE\_EVS\_SetEventFormatMode\_Payload\_t  
*Set Event Format Mode Command Payload.*
- typedef struct [CFE\\_EVS\\_BitMaskCmd\\_Payload](#) CFE\_EVS\_BitMaskCmd\_Payload\_t  
*Generic Bitmask Command Payload.*
- typedef struct [CFE\\_EVS\\_AppNameCmd\\_Payload](#) CFE\_EVS\_AppNameCmd\_Payload\_t  
*Generic App Name Command Payload.*
- typedef struct [CFE\\_EVS\\_AppNameEventIDCmd\\_Payload](#) CFE\_EVS\_AppNameEventIDCmd\_Payload\_t  
*Generic App Name and Event ID Command Payload.*
- typedef struct [CFE\\_EVS\\_AppNameBitMaskCmd\\_Payload](#) CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t  
*Generic App Name and Bitmask Command Payload.*
- typedef struct [CFE\\_EVS\\_AppNameEventIDMaskCmd\\_Payload](#) CFE\_EVS\_AppNameEventIDMaskCmd\_Payload\_t  
*Generic App Name, Event ID, Mask Command Payload.*
- typedef struct [CFE\\_EVS\\_AppTlmData](#) CFE\_EVS\_AppTlmData\_t
- typedef struct [CFE\\_EVS\\_HousekeepingTlm\\_Payload](#) CFE\_EVS\_HousekeepingTlm\_Payload\_t
- typedef struct [CFE\\_EVS\\_PacketID](#) CFE\_EVS\_PacketID\_t
- typedef struct [CFE\\_EVS\\_LongEventTlm\\_Payload](#) CFE\_EVS\_LongEventTlm\_Payload\_t
- typedef struct [CFE\\_EVS\\_ShortEventTlm\\_Payload](#) CFE\_EVS\_ShortEventTlm\_Payload\_t

### 12.144.1 Detailed Description

Specification for the CFE Event Services (CFE\_EVS) command and telemetry message payload structures and constant definitions.

## 12.144.2 Macro Definition Documentation

**12.144.2.1 CFE\_EVS\_CRITICAL\_BIT** #define CFE\_EVS\_CRITICAL\_BIT (1 << (CFE\_EVS\_EventType\_CRITICAL - 1))

Definition at line 37 of file default\_cfe\_evs\_msgdefs.h.

**12.144.2.2 CFE\_EVS\_DEBUG\_BIT** #define CFE\_EVS\_DEBUG\_BIT (1 << (CFE\_EVS\_EventType\_DEBUG - 1))

Definition at line 34 of file default\_cfe\_evs\_msgdefs.h.

**12.144.2.3 CFE\_EVS\_ERROR\_BIT** #define CFE\_EVS\_ERROR\_BIT (1 << (CFE\_EVS\_EventType\_ERROR - 1))

Definition at line 36 of file default\_cfe\_evs\_msgdefs.h.

**12.144.2.4 CFE\_EVS\_INFORMATION\_BIT** #define CFE\_EVS\_INFORMATION\_BIT (1 << (CFE\_EVS\_EventType\_INFORMATION - 1))

Definition at line 35 of file default\_cfe\_evs\_msgdefs.h.

**12.144.2.5 CFE\_EVS\_PORT1\_BIT** #define CFE\_EVS\_PORT1\_BIT (1 << (CFE\_EVS\_EventOutput\_PORT1 - 1))

Definition at line 40 of file default\_cfe\_evs\_msgdefs.h.

**12.144.2.6 CFE\_EVS\_PORT2\_BIT** #define CFE\_EVS\_PORT2\_BIT (1 << (CFE\_EVS\_EventOutput\_PORT2 - 1))

Definition at line 41 of file default\_cfe\_evs\_msgdefs.h.

**12.144.2.7 CFE\_EVS\_PORT3\_BIT** #define CFE\_EVS\_PORT3\_BIT (1 << (CFE\_EVS\_EventOutput\_PORT3 - 1))

Definition at line 42 of file default\_cfe\_evs\_msgdefs.h.

**12.144.2.8 CFE\_EVS\_PORT4\_BIT** #define CFE\_EVS\_PORT4\_BIT (1 << (CFE\_EVS\_EventOutput\_PORT4 - 1))

Definition at line 43 of file default\_cfe\_evs\_msgdefs.h.

## 12.144.3 Typedef Documentation

**12.144.3.1 CFE\_EVS\_AppDataCmd\_Payload\_t** typedef struct CFE\_EVS\_AppDataCmd\_Payload CFE\_EVS\_AppDataCmd\_Payload\_t

Write Event Services Application Information to File Command Payload.

For command details, see [CFE\\_EVS\\_WRITE\\_APP\\_DATA\\_FILE\\_CC](#)

**12.144.3.2 CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t** typedef struct CFE\_EVS\_AppNameBitMaskCmd\_Payload CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t

Generic App Name and Bitmask Command Payload.

For command details, see [CFE\\_EVS\\_ENABLE\\_APP\\_EVENT\\_TYPE\\_CC](#) and/or [CFE\\_EVS\\_DISABLE\\_APP\\_EVENT\\_TYPE\\_CC](#)

**12.144.3.3 CFE\_EVS\_AppNameCmd\_Payload\_t** `typedef struct CFE_EVS_AppNameCmd_Payload CFE_EVS_AppNameCmd_Payload_t`  
Generic App Name Command Payload.

For command details, see [CFE\\_EVS\\_ENABLE\\_APP\\_EVENTS\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENTS\\_CC](#),  
[CFE\\_EVS\\_RESET\\_APP\\_COUNTER\\_CC](#) and/or [CFE\\_EVS\\_RESET\\_ALL\\_FILTERS\\_CC](#)

**12.144.3.4 CFE\_EVS\_AppNameEventIDCmd\_Payload\_t** `typedef struct CFE_EVS_AppNameEventIDCmd_Payload CFE_EVS_AppNameEventIDCmd_Payload_t`

Generic App Name and Event ID Command Payload.

For command details, see [CFE\\_EVS\\_RESET\\_FILTER\\_CC](#) and [CFE\\_EVS\\_DELETE\\_EVENT\\_FILTER\\_CC](#)

**12.144.3.5 CFE\_EVS\_AppNameEventIDMaskCmd\_Payload\_t** `typedef struct CFE_EVS_AppNameEventIDMaskCmd_Payload CFE_EVS_AppNameEventIDMaskCmd_Payload_t`

Generic App Name, Event ID, Mask Command Payload.

For command details, see [CFE\\_EVS\\_SET\\_FILTER\\_CC](#), [CFE\\_EVS\\_ADD\\_EVENT\\_FILTER\\_CC](#) and/or [CFE\\_EVS\\_DELETE\\_EVENT\\_FILTER\\_CC](#)

**12.144.3.6 CFE\_EVS\_AppTlmData\_t** `typedef struct CFE_EVS_AppTlmData CFE_EVS_AppTlmData_t`

**12.144.3.7 CFE\_EVS\_BitMaskCmd\_Payload\_t** `typedef struct CFE_EVS_BitMaskCmd_Payload CFE_EVS_BitMaskCmd_Payload_t`  
Generic Bitmask Command Payload.

For command details, see [CFE\\_EVS\\_ENABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_EVENT\\_TYPE\\_CC](#),  
[CFE\\_EVS\\_ENABLE\\_PORTS\\_CC](#) and/or [CFE\\_EVS\\_DISABLE\\_PORTS\\_CC](#)

**12.144.3.8 CFE\_EVS\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_EVS_HousekeepingTlm_Payload CFE_EVS_HousekeepingTlm_Payload_t`

**Name** Event Services Housekeeping Telemetry Packet

**12.144.3.9 CFE\_EVS\_LogFileCmd\_Payload\_t** `typedef struct CFE_EVS_LogFileCmd_Payload CFE_EVS_LogFileCmd_Payload_t`  
Write Event Log to File Command Payload.

For command details, see [CFE\\_EVS\\_WRITE\\_LOG\\_DATA\\_FILE\\_CC](#)

**12.144.3.10 CFE\_EVS\_LongEventTlm\_Payload\_t** `typedef struct CFE_EVS_LongEventTlm_Payload CFE_EVS_LongEventTlm_Payload_t`

**Name** Event Message Telemetry Packet (Long format)

**12.144.3.11 CFE\_EVS\_PacketID\_t** `typedef struct CFE_EVS_PacketID CFE_EVS_PacketID_t`  
Telemetry packet structures

**12.144.3.12 CFE\_EVS\_SetEventFormatMode\_Payload\_t** `typedef struct CFE_EVS_SetEventFormatCode_Payload CFE_EVS_SetEventFormatMode_Payload_t`

Set Event Format Mode Command Payload.

For command details, see [CFE\\_EVS\\_SET\\_EVENT\\_FORMAT\\_MODE\\_CC](#)

**12.144.3.13 CFE\_EVS\_SetLogMode\_Payload\_t** `typedef struct CFE_EVS_SetLogMode_Payload CFE_EVS_SetLogMode_Payload_t`  
Set Log Mode Command Payload.

For command details, see [CFE\\_EVS\\_SET\\_LOG\\_MODE\\_CC](#)

**12.144.3.14 CFE\_EVS\_ShortEventTlm\_Payload\_t** `typedef struct CFE_EVS_ShortEventTlm_Payload CFE_EVS_ShortEventTlm_`

**Name** Event Message Telemetry Packet (Short format)

**12.145 cfe/modules/evs/config/default\_cfe\_evs\_msgid\_values.h File Reference**

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_evs_topicids.h"
```

**Macros**

- `#define CFE_PLATFORM_EVS_CMD_MIDVAL(x) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_`  
`MISSION_EVS_##x##_TOPICID)`
- `#define CFE_PLATFORM_EVS_TLM_MIDVAL(x) CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_`  
`MISSION_EVS_##x##_TOPICID)`

**12.145.1 Detailed Description**

CFE Event Services (CFE\_EVS) Application Message IDs

**12.145.2 Macro Definition Documentation****12.145.2.1 CFE\_PLATFORM\_EVS\_CMD\_MIDVAL** `#define CFE_PLATFORM_EVS_CMD_MIDVAL(`  
`x ) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_EVS_##x##_TOPICID)`

Definition at line 29 of file default\_cfe\_evs\_msgid\_values.h.

**12.145.2.2 CFE\_PLATFORM\_EVS\_TLM\_MIDVAL** `#define CFE_PLATFORM_EVS_TLM_MIDVAL(`  
`x ) CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_EVS_##x##_TOPICID)`

Definition at line 30 of file default\_cfe\_evs\_msgid\_values.h.

**12.146 cfe/modules/evs/config/default\_cfe\_evs\_msgids.h File Reference**

```
#include "cfe_evs_msgid_values.h"
```

**Macros**

- `#define CFE_EVS_CMD_MID CFE_PLATFORM_EVS_CMD_MIDVAL(CMD)`
- `#define CFE_EVS_SEND_HK_MID CFE_PLATFORM_EVS_CMD_MIDVAL(SEND_HK)`
- `#define CFE_EVS_HK_TLM_MID CFE_PLATFORM_EVS_TLM_MIDVAL(HK_TLM)`
- `#define CFE_EVS_LONG_EVENT_MSG_MID CFE_PLATFORM_EVS_TLM_MIDVAL(LONG_EVENT_MSG)`
- `#define CFE_EVS_SHORT_EVENT_MSG_MID CFE_PLATFORM_EVS_TLM_MIDVAL(SHORT_EVENT_MSG)`

**12.146.1 Detailed Description**

CFE Event Services (CFE\_EVS) Application Message IDs

**12.146.2 Macro Definition Documentation**

**12.146.2.1 CFE\_EVS\_CMD\_MID** #define CFE\_EVS\_CMD\_MID CFE\_PLATFORM\_EVS\_CMD\_MIDVAL (CMD)  
Definition at line 32 of file default\_cfe\_evs\_msgids.h.

**12.146.2.2 CFE\_EVS\_HK\_TLM\_MID** #define CFE\_EVS\_HK\_TLM\_MID CFE\_PLATFORM\_EVS\_TLM\_MIDVAL (HK\_TLM)  
Definition at line 38 of file default\_cfe\_evs\_msgids.h.

**12.146.2.3 CFE\_EVS\_LONG\_EVENT\_MSG\_MID** #define CFE\_EVS\_LONG\_EVENT\_MSG\_MID CFE\_PLATFORM\_EVS\_TLM\_MIDVAL (LONG←  
\_EVENT\_MSG)

Definition at line 39 of file default\_cfe\_evs\_msgids.h.

**12.146.2.4 CFE\_EVS\_SEND\_HK\_MID** #define CFE\_EVS\_SEND\_HK\_MID CFE\_PLATFORM\_EVS\_CMD\_MIDVAL (SEND←  
HK)

Definition at line 33 of file default\_cfe\_evs\_msgids.h.

**12.146.2.5 CFE\_EVS\_SHORT\_EVENT\_MSG\_MID** #define CFE\_EVS\_SHORT\_EVENT\_MSG\_MID CFE\_PLATFORM\_EVS\_TLM\_MIDVAL (SHOP←  
\_EVENT\_MSG)

Definition at line 40 of file default\_cfe\_evs\_msgids.h.

## 12.147 cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h File Reference

```
#include "common_types.h"
#include "cfe_evs_msgdefs.h"
#include "cfe_evs_extern_typedefs.h"
#include "cfe_msg_hdr.h"
```

### Data Structures

- struct **CFE\_EVS\_NoopCmd**
- struct **CFE\_EVS\_ResetCountersCmd**
- struct **CFE\_EVS\_ClearLogCmd**
- struct **CFE\_EVS\_SendHkCmd**
- struct **CFE\_EVS\_WriteLogFileCmd**  
*Write Event Log to File Command.*
- struct **CFE\_EVS\_WriteAppDataFileCmd**  
*Write Event Services Application Information to File Command.*
- struct **CFE\_EVS\_SetLogModeCmd**  
*Set Log Mode Command.*
- struct **CFE\_EVS\_SetEventFormatModeCmd**  
*Set Event Format Mode Command.*
- struct **CFE\_EVS\_EnablePortsCmd**
- struct **CFE\_EVS\_DisablePortsCmd**
- struct **CFE\_EVS\_EnableEventTypeCmd**
- struct **CFE\_EVS\_DisableEventTypeCmd**
- struct **CFE\_EVS\_EnableAppEventsCmd**
- struct **CFE\_EVS\_DisableAppEventsCmd**
- struct **CFE\_EVS\_ResetAppCounterCmd**

- struct [CFE\\_EVS\\_ResetAllFiltersCmd](#)
- struct [CFE\\_EVS\\_ResetFilterCmd](#)
- struct [CFE\\_EVS\\_DeleteEventFilterCmd](#)
- struct [CFE\\_EVS\\_EnableAppEventTypeCmd](#)
- struct [CFE\\_EVS\\_DisableAppEventTypeCmd](#)
- struct [CFE\\_EVS\\_AddEventFilterCmd](#)
- struct [CFE\\_EVS\\_SetFilterCmd](#)
- struct [CFE\\_EVS\\_HousekeepingTlm](#)
- struct [CFE\\_EVS\\_LongEventTlm](#)
- struct [CFE\\_EVS\\_ShortEventTlm](#)

## Typedefs

- typedef struct [CFE\\_EVS\\_NoopCmd](#) CFE\_EVS\_NoopCmd\_t
- typedef struct [CFE\\_EVS\\_ResetCountersCmd](#) CFE\_EVS\_ResetCountersCmd\_t
- typedef struct [CFE\\_EVS\\_ClearLogCmd](#) CFE\_EVS\_ClearLogCmd\_t
- typedef struct [CFE\\_EVS\\_SendHkCmd](#) CFE\_EVS\_SendHkCmd\_t
- typedef struct [CFE\\_EVS\\_WriteLogFileCmd](#) CFE\_EVS\_WriteLogFileCmd\_t
  - Write Event Log to File Command.*
- typedef struct [CFE\\_EVS\\_WriteAppDataFileCmd](#) CFE\_EVS\_WriteAppDataFileCmd\_t
  - Write Event Services Application Information to File Command.*
- typedef struct [CFE\\_EVS\\_SetLogModeCmd](#) CFE\_EVS\_SetLogModeCmd\_t
  - Set Log Mode Command.*
- typedef struct [CFE\\_EVS\\_SetEventFormatModeCmd](#) CFE\_EVS\_SetEventFormatModeCmd\_t
  - Set Event Format Mode Command.*
- typedef struct [CFE\\_EVS\\_EnablePortsCmd](#) CFE\_EVS\_EnablePortsCmd\_t
- typedef struct [CFE\\_EVS\\_DisablePortsCmd](#) CFE\_EVS\_DisablePortsCmd\_t
- typedef struct [CFE\\_EVS\\_EnableEventTypeCmd](#) CFE\_EVS\_EnableEventTypeCmd\_t
- typedef struct [CFE\\_EVS\\_DisableEventTypeCmd](#) CFE\_EVS\_DisableEventTypeCmd\_t
- typedef struct [CFE\\_EVS\\_EnableAppEventsCmd](#) CFE\_EVS\_EnableAppEventsCmd\_t
- typedef struct [CFE\\_EVS\\_DisableAppEventsCmd](#) CFE\_EVS\_DisableAppEventsCmd\_t
- typedef struct [CFE\\_EVS\\_ResetAppCounterCmd](#) CFE\_EVS\_ResetAppCounterCmd\_t
- typedef struct [CFE\\_EVS\\_ResetAllFiltersCmd](#) CFE\_EVS\_ResetAllFiltersCmd\_t
- typedef struct [CFE\\_EVS\\_ResetFilterCmd](#) CFE\_EVS\_ResetFilterCmd\_t
- typedef struct [CFE\\_EVS\\_DeleteEventFilterCmd](#) CFE\_EVS\_DeleteEventFilterCmd\_t
- typedef struct [CFE\\_EVS\\_EnableAppEventTypeCmd](#) CFE\_EVS\_EnableAppEventTypeCmd\_t
- typedef struct [CFE\\_EVS\\_DisableAppEventTypeCmd](#) CFE\_EVS\_DisableAppEventTypeCmd\_t
- typedef struct [CFE\\_EVS\\_AddEventFilterCmd](#) CFE\_EVS\_AddEventFilterCmd\_t
- typedef struct [CFE\\_EVS\\_SetFilterCmd](#) CFE\_EVS\_SetFilterCmd\_t
- typedef struct [CFE\\_EVS\\_HousekeepingTlm](#) CFE\_EVS\_HousekeepingTlm\_t
- typedef struct [CFE\\_EVS\\_LongEventTlm](#) CFE\_EVS\_LongEventTlm\_t
- typedef struct [CFE\\_EVS\\_ShortEventTlm](#) CFE\_EVS\_ShortEventTlm\_t

### 12.147.1 Detailed Description

Specification for the CFE Event Services (CFE\_EVS) command and telemetry message frame structures.

### 12.147.2 Typedef Documentation

**12.147.2.1 CFE\_EVS\_AddEventFilterCmd\_t** `typedef struct CFE_EVS_AddEventFilterCmd CFE_EVS_AddEventFilterCmd_t`

**12.147.2.2 CFE\_EVS\_ClearLogCmd\_t** `typedef struct CFE_EVS_ClearLogCmd CFE_EVS_ClearLogCmd_t`

**12.147.2.3 CFE\_EVS\_DeleteEventFilterCmd\_t** `typedef struct CFE_EVS_DeleteEventFilterCmd CFE_EVS_DeleteEventFilterCmd_t`

**12.147.2.4 CFE\_EVS\_DisableAppEventsCmd\_t** `typedef struct CFE_EVS_DisableAppEventsCmd CFE_EVS_DisableAppEventsCmd_t`

**12.147.2.5 CFE\_EVS\_DisableAppEventTypeCmd\_t** `typedef struct CFE_EVS_DisableAppEventTypeCmd CFE_EVS_DisableAppEventTypeCmd_t`

**12.147.2.6 CFE\_EVS\_DisableEventTypeCmd\_t** `typedef struct CFE_EVS_DisableEventTypeCmd CFE_EVS_DisableEventTypeCmd_t`

**12.147.2.7 CFE\_EVS\_DisablePortsCmd\_t** `typedef struct CFE_EVS_DisablePortsCmd CFE_EVS_DisablePortsCmd_t`

**12.147.2.8 CFE\_EVS\_EnableAppEventsCmd\_t** `typedef struct CFE_EVS_EnableAppEventsCmd CFE_EVS_EnableAppEventsCmd_t`

**12.147.2.9 CFE\_EVS\_EnableAppEventTypeCmd\_t** `typedef struct CFE_EVS_EnableAppEventTypeCmd CFE_EVS_EnableAppEventTypeCmd_t`

**12.147.2.10 CFE\_EVS\_EnableEventTypeCmd\_t** `typedef struct CFE_EVS_EnableEventTypeCmd CFE_EVS_EnableEventTypeCmd_t`

**12.147.2.11 CFE\_EVS\_EnablePortsCmd\_t** `typedef struct CFE_EVS_EnablePortsCmd CFE_EVS_EnablePortsCmd_t`

**12.147.2.12 CFE\_EVS\_HousekeepingTlm\_t** `typedef struct CFE_EVS_HousekeepingTlm CFE_EVS_HousekeepingTlm_t`

**12.147.2.13 CFE\_EVS\_LongEventTlm\_t** `typedef struct CFE_EVS_LongEventTlm CFE_EVS_LongEventTlm_t`

**12.147.2.14 CFE\_EVS\_NoopCmd\_t** `typedef struct CFE_EVS_NoopCmd CFE_EVS_NoopCmd_t`

**12.147.2.15 CFE\_EVS\_ResetAllFiltersCmd\_t** `typedef struct CFE_EVS_ResetAllFiltersCmd CFE_EVS_ResetAllFiltersCmd_t`

**12.147.2.16 CFE\_EVS\_ResetAppCounterCmd\_t** `typedef struct CFE_EVS_ResetAppCounterCmd CFE_EVS_ResetAppCounterCmd_t`

**12.147.2.17 CFE\_EVS\_ResetCountersCmd\_t** `typedef struct CFE_EVS_ResetCountersCmd CFE_EVS_ResetCountersCmd_t`

**12.147.2.18 CFE\_EVS\_ResetFilterCmd\_t** `typedef struct CFE_EVS_ResetFilterCmd CFE_EVS_ResetFilterCmd_t`

**12.147.2.19 CFE\_EVS\_SendHkCmd\_t** `typedef struct CFE_EVS_SendHkCmd CFE_EVS_SendHkCmd_t`

**12.147.2.20 CFE\_EVS\_SetEventFormatModeCmd\_t** `typedef struct CFE_EVS_SetEventFormatModeCmd CFE_EVS_SetEventFormatModeCmd_t`  
Set Event Format Mode Command.

**12.147.2.21 CFE\_EVS\_SetFilterCmd\_t** `typedef struct CFE_EVS_SetFilterCmd CFE_EVS_SetFilterCmd_t`

**12.147.2.22 CFE\_EVS\_SetLogModeCmd\_t** `typedef struct CFE_EVS_SetLogModeCmd CFE_EVS_SetLogModeCmd_t`  
Set Log Mode Command.

**12.147.2.23 CFE\_EVS\_ShortEventTlm\_t** `typedef struct CFE_EVS_ShortEventTlm CFE_EVS_ShortEventTlm_t`

**12.147.2.24 CFE\_EVS\_WriteAppDataFileCmd\_t** `typedef struct CFE_EVS_WriteAppDataFileCmd CFE_EVS_WriteAppDataFileCmd_t`  
Write Event Services Application Information to File Command.

**12.147.2.25 CFE\_EVS\_WriteLogDataFileCmd\_t** `typedef struct CFE_EVS_WriteLogDataFileCmd CFE_EVS_WriteLogDataFileCmd_t`  
Write Event Log to File Command.

## 12.148 cfe/modules/evs/config/default\_cfe\_evs\_platform\_cfg.h File Reference

```
#include "cfe_evs_mission_cfg.h"
#include "cfe_evs_internal_cfg.h"
```

### 12.148.1 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.

## 12.149 cfe/modules/evs/config/default\_cfe\_evs\_topicid\_values.h File Reference

### Macros

- `#define CFE_MISSION_EVS_TIDVAL(x) DEFAULT_CFE_MISSION_EVS_##x##_TOPICID`

### 12.149.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Topic IDs

### 12.149.2 Macro Definition Documentation

**12.149.2.1 CFE\_MISSION\_EVS\_TIDVAL** #define CFE\_MISSION\_EVS\_TIDVAL (  
  x ) DEFAULT\_CFE\_MISSION\_EVS\_##x##\_TOPICID

Definition at line 26 of file default\_cfe\_evs\_topicid\_values.h.

## 12.150 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

- #define `CFE_EVS_ENAEVTTYPE_EID` 20  
*EVS Disable Ports Command Success Event ID.*
- #define `CFE_EVS_DISEVTTYPE_EID` 21  
*EVS Enable Event Type Command Success Event ID.*
- #define `CFE_EVS_SETEVTFMTMOD_EID` 22  
*EVS Disable Event Type Command Success Event ID.*
- #define `CFE_EVS_ENAAPPEVTTYPE_EID` 23  
*EVS Set Event Format Mode Command Success Event ID.*
- #define `CFE_EVS_DISAPPENTTYPE_EID` 24  
*EVS Enable App Event Type Command Success Event ID.*
- #define `CFE_EVS_ENAAPPEVT_EID` 25  
*EVS Disable App Event Type Command Success Event ID.*
- #define `CFE_EVS_DISAPPEVT_EID` 26  
*EVS Enable 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.*

### 12.150.1 Detailed Description

cFE Event Services Event IDs

### 12.150.2 Macro Definition Documentation

**12.150.2.1 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.

**12.150.2.2 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.

**12.150.2.3 CFE\_EVS\_DISAPPENTTYPE\_EID** #define CFE\_EVS\_DISAPPENTTYPE\_EID 24  
EVS Disable App Event Type Command Success Event ID.

Type: DEBUG

Cause:

[EVS Disable App Event Type Command](#) success.  
Definition at line 301 of file cfe\_evs\_eventids.h.

**12.150.2.4 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.

**12.150.2.5 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.

**12.150.2.6 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.

**12.150.2.7 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.

**12.150.2.8 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.

**12.150.2.9 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.

**12.150.2.10 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.

**12.150.2.11 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.

**12.150.2.12 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.

**12.150.2.13 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.

**12.150.2.14 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.

**12.150.2.15 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.

**12.150.2.16 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.

**12.150.2.17 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.

**12.150.2.18 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.

**12.150.2.19 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.

**12.150.2.20 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.

**12.150.2.21 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.

**12.150.2.22 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.

**12.150.2.23 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.

**12.150.2.24 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.

**12.150.2.25 CFE\_EVS\_ERR\_WRLOGFILE\_EID** #define CFE\_EVS\_ERR\_WRLOGFILE\_EID 2  
EVS Write Event Log Command File Write Entry Failed Event ID.

Type: ERROR

Cause:

[EVS Write Event Log Command](#) failure writing data to the file.

Definition at line 65 of file cfe\_evs\_eventids.h.

**12.150.2.26 CFE\_EVS\_EVT\_FILTERED\_EID** #define CFE\_EVS\_EVT\_FILTERED\_EID 37  
EVS Add Filter Command Duplicate Registration Event ID.

Type: ERROR

Cause:

[EVS Add Filter Command](#) failure due to event already being registered for filtering.

Definition at line 412 of file cfe\_evs\_eventids.h.

**12.150.2.27 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.

**12.150.2.28 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.

**12.150.2.29 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.

**12.150.2.30 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.

**12.150.2.31 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.

**12.150.2.32 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.

**12.150.2.33 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.

**12.150.2.34 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.

**12.150.2.35 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.

**12.150.2.36 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.

**12.150.2.37 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.

**12.150.2.38 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.

**12.150.2.39 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.

**12.150.2.40 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.

**12.150.2.41 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.

## 12.151 cfe/modules/evs/fsw/inc/cfe\_evs\_fcncodes.h File Reference

```
#include "cfe_evs_fcncode_values.h"
```

### Macros

#### Event Services Command Codes

- #define CFE\_EVS\_NOOP\_CC CFE\_EVS\_CCVAL(NOOP)
- #define CFE\_EVS\_RESET\_COUNTERS\_CC CFE\_EVS\_CCVAL(RESET\_COUNTERS)
- #define CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC CFE\_EVS\_CCVAL(ENABLE\_EVENT\_TYPE)
- #define CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC CFE\_EVS\_CCVAL(DISABLE\_EVENT\_TYPE)
- #define CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC CFE\_EVS\_CCVAL(SET\_EVENT\_FORMAT\_MODE)
- #define CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC CFE\_EVS\_CCVAL(ENABLE\_APP\_EVENT\_TYPE)
- #define CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC CFE\_EVS\_CCVAL(DISABLE\_APP\_EVENT\_TYPE)
- #define CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC CFE\_EVS\_CCVAL(ENABLE\_APP\_EVENTS)
- #define CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC CFE\_EVS\_CCVAL(DISABLE\_APP\_EVENTS)
- #define CFE\_EVS\_RESET\_APP\_COUNTER\_CC CFE\_EVS\_CCVAL(RESET\_APP\_COUNTER)
- #define CFE\_EVS\_SET\_FILTER\_CC CFE\_EVS\_CCVAL(SET\_FILTER)
- #define CFE\_EVS\_ENABLE\_PORTS\_CC CFE\_EVS\_CCVAL(ENABLE\_PORTS)
- #define CFE\_EVS\_DISABLE\_PORTS\_CC CFE\_EVS\_CCVAL(DISABLE\_PORTS)
- #define CFE\_EVS\_RESET\_FILTER\_CC CFE\_EVS\_CCVAL(RESET\_FILTER)
- #define CFE\_EVS\_RESET\_ALL\_FILTERS\_CC CFE\_EVS\_CCVAL(RESET\_ALL\_FILTERS)
- #define CFE\_EVS\_ADD\_EVENT\_FILTER\_CC CFE\_EVS\_CCVAL(ADD\_EVENT\_FILTER)
- #define CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC CFE\_EVS\_CCVAL(DELETE\_EVENT\_FILTER)
- #define CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC CFE\_EVS\_CCVAL(WRITE\_APP\_DATA\_FILE)
- #define CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC CFE\_EVS\_CCVAL(WRITE\_LOG\_DATA\_FILE)
- #define CFE\_EVS\_SET\_LOG\_MODE\_CC CFE\_EVS\_CCVAL(SET\_LOG\_MODE)
- #define CFE\_EVS\_CLEAR\_LOG\_CC CFE\_EVS\_CCVAL(CLEAR\_LOG)

### 12.151.1 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.

### 12.151.2 Macro Definition Documentation

**12.151.2.1 CFE\_EVS\_ADD\_EVENT\_FILTER\_CC** #define CFE\_EVS\_ADD\_EVENT\_FILTER\_CC CFE\_EVS\_CCVAL (ADD←\_EVENT\_FILTER)

**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 695 of file cfe\_evs\_fcncodes.h.

**12.151.2.2 CFE\_EVS\_CLEAR\_LOG\_CC** #define CFE\_EVS\_CLEAR\_LOG\_CC CFE\_EVS\_CCVAL (CLEAR\_LOG)

**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 875 of file cfe\_evs\_fcncodes.h.

## 12.151.2.3 CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC #define CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC CFE\_EVS\_CCVAL(DELETE←\_EVENT\_FILTER)

### 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 730 of file cfe\_evs\_fcncodes.h.

**12.151.2.4 CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC** #define CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_←  
CC CFE\_EVS\_CCVAL (DISABLE\_APP\_EVENT\_TYPE)

**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 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 355 of file cfe\_evs\_fcncodes.h.

## 12.151.2.5 CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC #define CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC CFE\_EVS\_CCVAL (DISABLE↔\_APP\_EVENTS)

**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 433 of file cfe\_evs\_fcncodes.h.

**12.151.2.6 CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC** #define CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC CFE\_EVS\_CCVAL (DISABLE←\_EVENT\_TYPE)

**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 203 of file cfe\_evs\_fcncodes.h.

**12.151.2.7 CFE\_EVS\_DISABLE\_PORTS\_CC** #define CFE\_EVS\_DISABLE\_PORTS\_CC CFE\_EVS\_CCVAL(DISABLE\_←  
PORTS)

**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 589 of file cfe\_evs\_fcncodes.h.

**12.151.2.8 CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC** #define CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC CFE\_EVS\_CCVAL(ENABLE\_APP\_EVENT\_TYPE)

**Name** Enable Application Event Type

**Description**

This command enables the command specified event type for the command specified application, allowing the application to send event messages of the command specified event type through Event Service. An Event Type is defined to be a classification of an Event Message such as debug, informational, critical, and error. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_EnaAppEvtType, \$sc\_\$cpu\_EVS\_EnaAppEvtTypeMask

**Command Structure**

[CFE\\_EVS\\_EnableAppEventTypeCmd\\_t](#) The following bit positions apply to structure member named 'BitMask'. Bit 0 - Debug Bit 1 - Informational Bit 2 - Error Bit 3 - Critical A one in a bit position means the event type will be enabled (or unfiltered) for the specified application. A zero in a bit position means the filtering state is unchanged for the specified application.

**Command Verification**

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

- [\\$sc\\_\\$cpu\\_EVS\\_CMDPC](#) - command execution counter will increment
- The generation of [CFE\\_EVS\\_ENAAPPEVTTYPE\\_EID](#) debug event message

**Error Conditions**

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

- BitMask field invalid - mask cannot be zero, and only bits 0-3 may be set
- Application name is not valid or not registered with event services

Evidence of failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_EVS\\_CMDEC](#) - command error counter will increment
- An Error specific event message

**Criticality**

Enabling an application event type is not particularly hazardous, as the result may be turning on necessary event messages and communication to the ground system. However, inappropriately enabling an application's event type could result in flooding of the ground system.

**See also**

[CFE\\_EVS\\_ENABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_EVENT\\_TYPE\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENT\\_TYPE\\_CC](#),  
[CFE\\_EVS\\_ENABLE\\_APP\\_EVENTS\\_CC](#), [CFE\\_EVS\\_DISABLE\\_APP\\_EVENTS\\_CC](#)

Definition at line 302 of file cfe\_evs\_fcncodes.h.

**12.151.2.9 CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC** #define CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC CFE\_EVS\_CCVAL (ENABLE↔\_APP\_EVENTS)

**Name** Enable Event Services for an Application

#### Description

This command enables the command specified application to send events through the Event Service. Note: In order for this command to take effect, applications must be registered for Event Service.

**Command Mnemonic(s)** \$sc\_\$cpu\_EVS\_EnaAppEvGen

#### Command Structure

CFE\_EVS\_EnableAppEventsCmd\_t

#### Command Verification

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

- \$sc\_\$cpu\_EVS\_CMDPC - command execution counter will increment
- The generation of CFE\_EVS\_ENAAPPEVT\_EID debug event message
- The setting of the Active Flag in 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 394 of file cfe\_evs\_fcncodes.h.

**12.151.2.10 CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC** #define CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC CFE\_EVS\_CCVAL (ENABLE↔\_EVENT\_TYPE)

**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 154 of file cfe\_evs\_fcncodes.h.

**12.151.2.11 CFE\_EVS\_ENABLE\_PORTS\_CC** #define CFE\_EVS\_ENABLE\_PORTS\_CC [CFE\\_EVS\\_CCVAL](#)(ENABLE\_<PORTS>)

**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 550 of file cfe\_evs\_fcncodes.h.

## 12.151.2.12 CFE\_EVS\_NOOP\_CC #define CFE\_EVS\_NOOP\_CC CFE\_EVS\_CCVAL(NOOP)

**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 67 of file cfe\_evs\_fcncodes.h.

**12.151.2.13 CFE\_EVS\_RESET\_ALL\_FILTERS\_CC** #define CFE\_EVS\_RESET\_ALL\_FILTERS\_CC CFE\_EVS\_CCVAL(RESET↔\_ALL\_FILTERS)

**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 659 of file cfe\_evs\_fcncodes.h.

**12.151.2.14 CFE\_EVS\_RESET\_APP\_COUNTER\_CC** #define CFE\_EVS\_RESET\_APP\_COUNTER\_CC CFE\_EVS\_CCVAL (RESET←\_APP\_COUNTER)

**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 469 of file cfe\_evs\_fcncodes.h.

**12.151.2.15 CFE\_EVS\_RESET\_COUNTERS\_CC** #define CFE\_EVS\_RESET\_COUNTERS\_CC CFE\_EVS\_CCVAL (RESET←\_COUNTERS)

**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 106 of file cfe\_evs\_fcncodes.h.

**12.151.2.16 CFE\_EVS\_RESET\_FILTER\_CC** #define CFE\_EVS\_RESET\_FILTER\_CC CFE\_EVS\_CCVAL (RESET\_←  
FILTER)

**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 625 of file cfe\_evs\_fcncodes.h.

**12.151.2.17 CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC** #define CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_←  
CC CFE\_EVS\_CCVAL(SET\_EVENT\_FORMAT\_MODE)

**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 250 of file `cfe_evs_fcncodes.h`.

## 12.151.2.18 CFE\_EVS\_SET\_FILTER\_CC #define CFE\_EVS\_SET\_FILTER\_CC CFE\_EVS\_CCVAL(SET\_FILTER)

**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 511 of file cfe\_evs\_fcncodes.h.

**12.151.2.19 CFE\_EVS\_SET\_LOG\_MODE\_CC** #define CFE\_EVS\_SET\_LOG\_MODE\_CC CFE\_EVS\_CCVAL(SET\_LOG\_←  
MODE)

**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 840 of file cfe\_evs\_fcncodes.h.

**12.151.2.20 CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC** #define CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC CFE\_EVS\_CCVAL(WRITE←\_APP\_DATA\_FILE)

**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 769 of file cfe\_evs\_fcncodes.h.

**12.151.2.21 CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC** #define CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC CFE\_EVS\_CCVAL(WRITE↔\_LOG\_DATA\_FILE)

**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 804 of file `cfe_evs_fcncodes.h`.

## 12.152 cfe/modules/evs/fsw/inc/cfe\_evs\_interface\_cfg.h File Reference

```
#include "cfe_evs_interface_cfg_values.h"
```

#### Macros

- #define CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH CFE\_MISSION\_EVS\_CFGVAL(MAX\_MESSAGE↔LENGTH)
- #define DEFAULT\_CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH 122

### 12.152.1 Detailed Description

CFE Event Services (CFE\_EVS) Interface Configuration Header File

### 12.152.2 Macro Definition Documentation

**12.152.2.1 CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH** `#define CFE_MISSION_EVS_MAX_MESSAGE_LENGTH CFE_MISSION_EVS_MESSAGE_LENGTH)`

**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 41 of file cfe\_evs\_interface\_cfg.h.

**12.152.2.2 DEFAULT\_CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH** `#define DEFAULT_CFE_MISSION_EVS_MAX_MESSAGE_LENGTH 122`

Definition at line 42 of file cfe\_evs\_interface\_cfg.h.

## 12.153 cfe/modules/evs/fsw/inc/cfe\_evs\_internal\_cfg.h File Reference

```
#include "cfe_evs_mission_cfg.h"
#include "cfe_evs_internal_cfg_values.h"
```

### Macros

- `#define CFE_PLATFORM_EVS_START_TASK_PRIORITY CFE_PLATFORM_EVS_CFGVAL(START_TASK_PRIORITY)`
- `#define DEFAULT_CFE_PLATFORM_EVS_START_TASK_PRIORITY 61`
- `#define CFE_PLATFORM_EVS_START_TASK_STACK_SIZE CFE_PLATFORM_EVS_CFGVAL(START_TASK_STACK_SIZE)`
- `#define DEFAULT_CFE_PLATFORM_EVS_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE`
- `#define CFE_PLATFORM_EVS_MAX_EVENT_FILTERS CFE_PLATFORM_EVS_CFGVAL(MAX_EVENT_FILTERS)`
- `#define DEFAULT_CFE_PLATFORM_EVS_MAX_EVENT_FILTERS 8`
- `#define CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST CFE_PLATFORM_EVS_CFGVAL(MAX_APP_EVENT_BURST)`
- `#define DEFAULT_CFE_PLATFORM_EVS_MAX_APP_EVENT_BURST 32`
- `#define CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC CFE_PLATFORM_EVS_CFGVAL(APP_EVENTS_PER_SEC)`
- `#define DEFAULT_CFE_PLATFORM_EVS_APP_EVENTS_PER_SEC 15`
- `#define CFE_PLATFORM_EVS_DEFAULT_LOG_FILE CFE_PLATFORM_EVS_CFGVAL(DEFAULT_LOG_FILE)`
- `#define DEFAULT_CFE_PLATFORM_EVS_DEFAULT_LOG_FILE "/ram/cfe_evs.log"`

- #define CFE\_PLATFORM\_EVS\_LOG\_MAX CFE\_PLATFORM\_EVS\_CFGVAL(LOG\_MAX)
- #define DEFAULT\_CFE\_PLATFORM\_EVS\_LOG\_MAX 20
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE CFE\_PLATFORM\_EVS\_CFGVAL(DEFAULT\_APP\_DATA\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE "/ram/cfe\_evs\_app.dat"
- #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT CFE\_PLATFORM\_EVS\_CFGVAL(PORT\_DEFAULT)
- #define DEFAULT\_CFE\_PLATFORM\_EVS\_PORT\_DEFAULT 0x0001
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG CFE\_PLATFORM\_EVS\_CFGVAL(DEFAULT\_TYPE\_FLAG)
- #define DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG 0xE
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE CFE\_PLATFORM\_EVS\_CFGVAL(DEFAULT\_LOG\_MODE)
- #define DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE 1
- #define CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE CFE\_PLATFORM\_EVS\_CFGVAL(DEFAULT\_MSG\_FORMAT\_MODE)
- #define DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE CFE\_EVS\_MsgFormat\_LONG

### 12.153.1 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).

### 12.153.2 Macro Definition Documentation

#### 12.153.2.1 CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC #define CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC CFE\_PLATFORM\_EVS\_CFGVAL(APP\_EVENTS\_PER\_SEC)

**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 99 of file cfe\_evs\_internal\_cfg.h.

#### 12.153.2.2 CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE #define CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE CFE\_PLATFORM\_EVS\_CFGVAL(DEFAULT\_APP\_DATA\_FILE)

**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 143 of file `cfe_evs_internal_cfg.h`.

**12.153.2.3 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE** `#define CFE_PLATFORM_EVS_DEFAULT_LOG_FILE CFE_PLATFORM_EVS_CFO_`  
`_LOG_FILE)`

**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 114 of file `cfe_evs_internal_cfg.h`.

**12.153.2.4 CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE** `#define CFE_PLATFORM_EVS_DEFAULT_LOG_`  
`MODE CFE_PLATFORM_EVS_CFGVAL(DEFAULT_LOG_MODE)`

**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 193 of file `cfe_evs_internal_cfg.h`.

**12.153.2.5 CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE** `#define CFE_PLATFORM_EVS_DEFAULT_`  
`MSG_FORMAT_MODE CFE_PLATFORM_EVS_CFGVAL(DEFAULT_MSG_FORMAT_MODE)`

**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 207 of file `cfe_evs_internal_cfg.h`.

**12.153.2.6 CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG** #define CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG CFE\_PLATFORM\_EVS\_CFGVAL(DEFAULT\_TYPE\_FLAG)

**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 176 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.7 CFE\_PLATFORM\_EVS\_LOG\_MAX** #define CFE\_PLATFORM\_EVS\_LOG\_MAX CFE\_PLATFORM\_EVS\_CFGVAL(LOG\_MAX)

**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 127 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.8 CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST** #define CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST CFE\_PLATFORM\_EVS\_CFGVAL(MAX\_APP\_EVENT\_BURST)

**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 86 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.9 CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS** #define CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS

**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 73 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.10 CFE\_PLATFORM\_EVS\_PORT\_DEFAULT** #define CFE\_PLATFORM\_EVS\_PORT\_DEFAULT CFE\_PLATFORM\_EVS\_CFGVAL(PORT\_DEFAULT)

**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 158 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.11 CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY CFE\_PLATFORM\_EVS\_CFGVAL(START\_TASK\_PRIORITY)

**Purpose** Define EVS Task Priority

**Description:**

Defines the cFE\_EVS Task priority.

**Limits**

Not Applicable

Definition at line 43 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.12 CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_EVS\_START\_TASK\_←  
STACK\_SIZE CFE\_PLATFORM\_EVS\_CFGVAL(START\_TASK\_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 cfe\_evs\_internal\_cfg.h.

**12.153.2.13 DEFAULT\_CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC** #define DEFAULT\_CFE\_PLATFORM\_←  
EVS\_APP\_EVENTS\_PER\_SEC 15

Definition at line 100 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.14 DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE** #define DEFAULT\_CFE\_PLATFORM\_←  
\_EVS\_DEFAULT\_APP\_DATA\_FILE "/ram/cfe\_evs\_app.dat"

Definition at line 144 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.15 DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE** #define DEFAULT\_CFE\_PLATFORM\_EVS\_←  
DEFAULT\_LOG\_FILE "/ram/cfe\_evs.log"

Definition at line 115 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.16 DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE** #define DEFAULT\_CFE\_PLATFORM\_EVS\_←  
\_DEFAULT\_LOG\_MODE 1

Definition at line 194 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.17 DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE** #define DEFAULT\_CFE\_←  
PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE CFE\_EVS\_MsgFormat\_LONG

Definition at line 208 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.18 DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG** #define DEFAULT\_CFE\_PLATFORM\_EVS\_←  
\_DEFAULT\_TYPE\_FLAG 0xE

Definition at line 177 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.19 DEFAULT\_CFE\_PLATFORM\_EVS\_LOG\_MAX** #define DEFAULT\_CFE\_PLATFORM\_EVS\_LOG\_MAX 20

Definition at line 128 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.20 DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST** #define DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST 32  
Definition at line 87 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.21 DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS** #define DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS 8  
Definition at line 74 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.22 DEFAULT\_CFE\_PLATFORM\_EVS\_PORT\_DEFAULT** #define DEFAULT\_CFE\_PLATFORM\_EVS\_PORT\_DEFAULT DEFAULT 0x0001  
Definition at line 159 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.23 DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY 61  
Definition at line 44 of file cfe\_evs\_internal\_cfg.h.

**12.153.2.24 DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE  
Definition at line 60 of file cfe\_evs\_internal\_cfg.h.

## 12.154 cfe/modules/evs/fsw/inc/cfe\_evs\_topicids.h File Reference

```
#include "cfe_evs_topicid_values.h"
```

### Macros

- #define CFE\_MISSION\_EVS\_CMD\_TOPICID CFE\_MISSION\_EVS\_TIDVAL(CMD)
- #define DEFAULT\_CFE\_MISSION\_EVS\_CMD\_TOPICID 1
- #define CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID CFE\_MISSION\_EVS\_TIDVAL(SEND\_HK)
- #define DEFAULT\_CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID 9
- #define CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID CFE\_MISSION\_EVS\_TIDVAL(HK\_TLM)
- #define DEFAULT\_CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID 1
- #define CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID CFE\_MISSION\_EVS\_TIDVAL(LONG\_EVENT\_MSG)
- #define DEFAULT\_CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID 8
- #define CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID CFE\_MISSION\_EVS\_TIDVAL(SHORT\_EVENT\_MSG)
- #define DEFAULT\_CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID 9

### 12.154.1 Detailed Description

CFE Event Services (CFE\_EVS) Application Topic IDs

### 12.154.2 Macro Definition Documentation

**12.154.2.1 CFE\_MISSION\_EVS\_CMD\_TOPICID** #define CFE\_MISSION\_EVS\_CMD\_TOPICID CFE\_MISSION\_EVS\_TIDVAL (CMD)

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE EVS command messages

**Limits**

Not Applicable

Definition at line 37 of file cfe\_evs\_topicids.h.

**12.154.2.2 CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID** #define CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID CFE\_MISSION\_EVS\_TIDVAL (HK↔\_TLM)

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE EVS telemetry messages

**Limits**

Not Applicable

Definition at line 51 of file cfe\_evs\_topicids.h.

**12.154.2.3 CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID** #define CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG↔\_TOPICID CFE\_MISSION\_EVS\_TIDVAL (LONG\_EVENT\_MSG)

Definition at line 53 of file cfe\_evs\_topicids.h.

**12.154.2.4 CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID** #define CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID CFE\_MISSION\_EVS\_TIDVAL (SEND\_HK)

Definition at line 39 of file cfe\_evs\_topicids.h.

**12.154.2.5 CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID** #define CFE\_MISSION\_EVS\_SHORT\_EVENT↔\_MSG\_TOPICID CFE\_MISSION\_EVS\_TIDVAL (SHORT\_EVENT\_MSG)

Definition at line 55 of file cfe\_evs\_topicids.h.

**12.154.2.6 DEFAULT\_CFE\_MISSION\_EVS\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_EVS\_CMD\_TOPICID 1

Definition at line 38 of file cfe\_evs\_topicids.h.

**12.154.2.7 DEFAULT\_CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_EVS\_HK\_TLM↔\_TOPICID 1

Definition at line 52 of file cfe\_evs\_topicids.h.

**12.154.2.8 DEFAULT\_CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID** #define DEFAULT\_CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID 8  
Definition at line 54 of file cfe\_evs\_topicids.h.

**12.154.2.9 DEFAULT\_CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID** #define DEFAULT\_CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID 9  
Definition at line 40 of file cfe\_evs\_topicids.h.

**12.154.2.10 DEFAULT\_CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID** #define DEFAULT\_CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID 9  
Definition at line 56 of file cfe\_evs\_topicids.h.

## 12.155 cfe/modules/fs/config/default\_cfe\_fs\_extern\_typedefs.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_fs_filedef.h"
```

### 12.155.1 Detailed Description

Public type definitions for the CFE FS module

## 12.156 cfe/modules/fs/config/default\_cfe\_fs\_filedef.h File Reference

```
#include "common_types.h"
#include "cfe_mission_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\\_ES\\_APPDATA](#) = 15 ,  
    [CFE\\_FS\\_SubType\\_ES\\_EVENTLOG](#) = 16 , [CFE\\_FS\\_SubType\\_SB\\_PIPEDATA](#) = 20 , [CFE\\_FS\\_SubType\\_SB\\_ROUTEDATA](#) = 21 , [CFE\\_FS\\_SubType\\_SB\\_MAPDATA](#) = 22 ,  
    [CFE\\_FS\\_SubType\\_ES\\_QUERYALLTASKS](#) = 23 }

*File subtypes used within cFE.*

### 12.156.1 Detailed Description

Public type definitions for the CFE FS module

### 12.156.2 Typedef Documentation

**12.156.2.1 CFE\_FS\_Header\_t** `typedef struct CFE_FS_Header CFE_FS_Header_t`  
 Standard cFE File header structure definition.

**12.156.2.2 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.

### 12.156.3 Enumeration Type Documentation

**12.156.3.1 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 <a href="#">\$sc_\$cpu_ES_WriteERLog2File</a> 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 <a href="#">\$sc_\$cpu_ES_WriteSysLog2File</a> 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 <a href="#">\$sc_\$cpu_ES_WriteApplInfo2File</a> command.
<code>CFE_FS_SubType_ES_PERFDATA</code>	Executive Services Performance Data File. Executive Services Performance Analyzer Data File which is generated in response to a <a href="#">\$sc_\$cpu_ES_StopLAData</a> command.
<code>CFE_FS_SubType_ES_CDS_REG</code>	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.
<code>CFE_FS_SubType_TBL_REG</code>	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.
<code>CFE_FS_SubType_TBL_IMG</code>	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.

**Enumerator**

CFE_FS_SubType_EVS_APPDATA	Event Services Application Data Dump File. Event Services Application Data Dump File which is generated in response to a <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.

## 12.157 cfe/modules/fs/config/default\_cfe\_fs\_interface\_cfg\_values.h File Reference

### Macros

- #define CFE\_MISSION\_FS\_CFGVAL(x) DEFAULT\_CFE\_FS\_##x

#### 12.157.1 Detailed Description

CFE File Services (CFE\_FS) Module Public Definitions

This provides default values for configurable items that affect the interface(s) of this module.

### 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.

#### 12.157.2 Macro Definition Documentation

##### 12.157.2.1 CFE\_MISSION\_FS\_CFGVAL #define CFE\_MISSION\_FS\_CFGVAL ( x ) DEFAULT\_CFE\_FS\_##x

Definition at line 34 of file default\_cfe\_fs\_interface\_cfg\_values.h.

## 12.158 cfe/modules/fs/config/default\_cfe\_fs\_mission\_cfg.h File Reference

```
#include "cfe_fs_interface_cfg.h"
```

### 12.158.1 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.

## 12.159 cfe/modules/fs/fsw/inc/cfe\_fs\_interface\_cfg.h File Reference

```
#include "cfe_fs_interface_cfg_values.h"
```

### Macros

- #define CFE\_FS\_HDR\_DESC\_MAX\_LEN CFE\_MISSION\_FS\_CFGVAL(HDR\_DESC\_MAX\_LEN)  
*Max length of description field in a standard cFE File Header.*
- #define DEFAULT\_CFE\_FS\_HDR\_DESC\_MAX\_LEN 32
- #define CFE\_FS\_FILE\_CONTENT\_ID CFE\_MISSION\_FS\_CFGVAL(FILE\_CONTENT\_ID)  
*Magic Number for cFE compliant files (= 'cFE1')*
- #define DEFAULT\_CFE\_FS\_FILE\_CONTENT\_ID 0x63464531

### 12.159.1 Detailed Description

CFE File Services (CFE\_FS) Module Public Definitions

This provides default values for configurable items that affect the interface(s) of this module.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.159.2 Macro Definition Documentation

**12.159.2.1 CFE\_FS\_FILE\_CONTENT\_ID** #define CFE\_FS\_FILE\_CONTENT\_ID CFE\_MISSION\_FS\_CFGVAL(FILE\_CONTENT\_ID)

Magic Number for cFE compliant files (= 'cFE1')

Definition at line 52 of file cfe\_fs\_interface\_cfg.h.

**12.159.2.2 CFE\_FS\_HDR\_DESC\_MAX\_LEN** #define CFE\_FS\_HDR\_DESC\_MAX\_LEN CFE\_MISSION\_FS\_CFGVAL(HDR\_DESC\_MAX\_LEN)

Max length of description field in a standard cFE File Header.

#### Note

the value of CFE\_FS\_HDR\_DESC\_MAX\_LEN, if modified, should be constrained to multiples of 4, as it is used within a structure that also contains uint32 types. This ensures that the entire structure remains 32-bit aligned without the need for implicit padding bytes.

Definition at line 46 of file cfe\_fs\_interface\_cfg.h.

**12.159.2.3 DEFAULT\_CFE\_FS\_FILE\_CONTENT\_ID** #define DEFAULT\_CFE\_FS\_FILE\_CONTENT\_ID 0x63464531  
Definition at line 53 of file cfe\_fs\_interface\_cfg.h.

**12.159.2.4 DEFAULT\_CFE\_FS\_HDR\_DESC\_MAX\_LEN** #define DEFAULT\_CFE\_FS\_HDR\_DESC\_MAX\_LEN 32  
Definition at line 47 of file cfe\_fs\_interface\_cfg.h.

## 12.160 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.*

### 12.160.1 Detailed Description

Define CCSDS packet header types

- Avoid direct access for portability, use APIs
- Used to construct message structures

### 12.160.2 Typedef Documentation

**12.160.2.1 CCSDS\_ExtendedHeader\_t** [typedef struct CCSDS\\_ExtendedHeader](#) CCSDS\_ExtendedHeader\_t  
CCSDS packet extended header.

**12.160.2.2 CCSDS\_PrimaryHeader\_t** [typedef struct CCSDS\\_PrimaryHeader](#) CCSDS\_PrimaryHeader\_t  
CCSDS packet primary header.

## 12.161 cfe/modules/resourceid/fsw/inc/cfe\_core\_resourceid\_basevalues.h File Reference

```
#include "cfe_resourceid_basevalue.h"
```

## Enumerations

- enum {
   
`CFE_RESOURCEID_ES_TASKID_BASE_OFFSET = OS_OBJECT_TYPE_OS_TASK, CFE_RESOURCEID_ES_APPID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 1, CFE_RESOURCEID_ES_LIBID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 2, CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 3, CFE_RESOURCEID_ES_POOLID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 4, CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 5, CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET = OS_OBJECT_TYPE_USER + 6, CFE_RESOURCEID_CONFIGID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 7,`
  
`CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 8, CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 9, CFE_RESOURCEID_TBL_LOADBUFFID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 10, CFE_RESOURCEID_TBL_ACCESSID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 11,`
  
`CFE_RESOURCEID_TBL_REGID_BASE_OFFSET = OS_OBJECT_TYPE_USER + 12 }`
- enum {
   
`CFE_ES_TASKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_TASKID_BASE_OFFSET), CFE_ES_APPID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_APPID_BASE_OFFSET), CFE_ES_LIBID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_LIBID_BASE_OFFSET), CFE_ES_COUNTID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_COUNTID_BASE_OFFSET),`
  
`CFE_ES_POOLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_POOLID_BASE_OFFSET), CFE_ES_CDSBLOCKID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_ES_CDSBLOCKID_BASE_OFFSET), CFE_SB_PIPEID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_SB_PIPEID_RESOURCE_BASE_OFFSET), CFE_CONFIGID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_CONFIGID_BASE_OFFSET),`
  
`CFE_TBL_VALRESULTID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_VALRESULTID_BASE_OFFSET), CFE_TBL_DUMPCTRLID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_DUMPCTRLID_BASE_OFFSET), CFE_TBL_LOADBUFFID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_LOADBUFFID_BASE_OFFSET), CFE_TBL_HANDLE_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_ACCESSID_BASE_OFFSET),`
  
`CFE_TBL_REGID_BASE = CFE_RESOURCEID_MAKE_BASE(CFE_RESOURCEID_TBL_REGID_BASE_OFFSET) }`

### 12.161.1 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.

## 12.162 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.*

### 12.162.1 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.

### 12.162.2 Macro Definition Documentation

**12.162.2.1 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.

**12.162.2.2 CFE\_RESOURCEID\_MAX** `#define CFE_RESOURCEID_MAX OS_OBJECT_INDEX_MASK`

Definition at line 65 of file cfe\_resourceid\_basevalue.h.

**12.162.2.3 CFE\_RESOURCEID\_SHIFT** `#define CFE_RESOURCEID_SHIFT OS_OBJECT_TYPE_SHIFT`

Definition at line 64 of file cfe\_resourceid\_basevalue.h.

## 12.163 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.*

### 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\\_RoutId\\_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.*

### 12.163.1 Detailed Description

Declarations and prototypes for cfe\_sb\_extern\_typedefs module

### 12.163.2 Typedef Documentation

#### 12.163.2.1 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 89 of file default\_cfe\_sb\_extern\_typedefs.h.

#### 12.163.2.2 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 112 of file default\_cfe\_sb\_extern\_typedefs.h.

#### 12.163.2.3 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 53 of file default\_cfe\_sb\_extern\_typedefs.h.

#### 12.163.2.4 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 76 of file default\_cfe\_sb\_extern\_typedefs.h.

#### 12.163.2.5 CFE\_SB\_Routeld\_Atom\_t `typedef uint16 CFE_SB_RouteId_Atom_t`

An integer type that should be used for indexing into the Routing Table.

Definition at line 81 of file default\_cfe\_sb\_extern\_typedefs.h.

### 12.163.3 Enumeration Type Documentation

#### 12.163.3.1 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 35 of file default\_cfe\_sb\_extern\_typedefs.h.

#### 12.163.3.2 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 58 of file default\_cfe\_sb\_extern\_typedefs.h.

## 12.164 cfe/modules/sb/config/default\_cfe\_sb\_fcncode\_values.h File Reference

### Macros

- #define CFE\_SB\_CCVAL(x) CFE\_SB\_FunctionCode\_##x

### Enumerations

- enum CFE\_SB\_FunctionCode\_ {  
    CFE\_SB\_FunctionCode\_NOOP = 0, CFE\_SB\_FunctionCode\_RESET\_COUNTERS = 1, CFE\_SB\_FunctionCode\_SEND\_SB\_STA  
    = 2, CFE\_SB\_FunctionCode\_WRITE\_ROUTING\_INFO = 3,  
    CFE\_SB\_FunctionCode\_ENABLE\_ROUTE = 4, CFE\_SB\_FunctionCode\_DISABLE\_ROUTE = 5, CFE\_SB\_FunctionCode\_WRITE  
    = 7, CFE\_SB\_FunctionCode\_WRITE\_MAP\_INFO = 8,  
    CFE\_SB\_FunctionCode\_ENABLE\_SUB\_REPORTING = 9, CFE\_SB\_FunctionCode\_DISABLE\_SUB\_REPORTING  
    = 10, CFE\_SB\_FunctionCode\_SEND\_PREV\_SUBS = 11 }

### 12.164.1 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.

### 12.164.2 Macro Definition Documentation

```
12.164.2.1 CFE_SB_CCVAL #define CFE_SB_CCVAL (
    x ) CFE_SB_FunctionCode_##x
```

Definition at line 35 of file default\_cfe\_sb\_fnccode\_values.h.

### 12.164.3 Enumeration Type Documentation

#### 12.164.3.1 CFE\_SB\_FunctionCode\_ enum CFE\_SB\_FunctionCode\_

Enumerator

CFE_SB_FunctionCode_NOOP
CFE_SB_FunctionCode_RESET_COUNTERS
CFE_SB_FunctionCode_SEND_SB_STATS
CFE_SB_FunctionCode_WRITE_ROUTING_INFO
CFE_SB_FunctionCode_ENABLE_ROUTE
CFE_SB_FunctionCode_DISABLE_ROUTE
CFE_SB_FunctionCode_WRITE_PIPE_INFO
CFE_SB_FunctionCode_WRITE_MAP_INFO
CFE_SB_FunctionCode_ENABLE_SUB_REPORTING
CFE_SB_FunctionCode_DISABLE_SUB_REPORTING
CFE_SB_FunctionCode_SEND_PREV_SUBS

Definition at line 37 of file default\_cfe\_sb\_fnccode\_values.h.

## 12.165 cfe/modules/sb/config/default\_cfe\_sb\_interface\_cfg\_values.h File Reference

### Macros

- #define CFE\_MISSION\_SB\_CFGVAL(x) DEFAULT\_CFE\_MISSION\_SB\_##x

#### 12.165.1 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.

#### 12.165.2 Macro Definition Documentation

##### 12.165.2.1 CFE\_MISSION\_SB\_CFGVAL #define CFE\_MISSION\_SB\_CFGVAL (

```
x ) DEFAULT_CFE_MISSION_SB_##x
```

Definition at line 36 of file default\_cfe\_sb\_interface\_cfg\_values.h.

## 12.166 cfe/modules/sb/config/default\_cfe\_sb\_internal\_cfg\_values.h File Reference

### Macros

- #define CFE\_PLATFORM\_SB\_CFGVAL(x) DEFAULT\_CFE\_PLATFORM\_SB\_##x

#### 12.166.1 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.

#### 12.166.2 Macro Definition Documentation

##### 12.166.2.1 CFE\_PLATFORM\_SB\_CFGVAL #define CFE\_PLATFORM\_SB\_CFGVAL ( x ) DEFAULT\_CFE\_PLATFORM\_SB\_##x

Definition at line 36 of file default\_cfe\_sb\_internal\_cfg\_values.h.

## 12.167 cfe/modules/sb/config/default\_cfe\_sb\_mission\_cfg.h File Reference

```
#include "cfe_sb_interface_cfg.h"
```

#### 12.167.1 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.

## 12.168 cfe/modules/sb/config/default\_cfe\_sb\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_sb_fcncodes.h"
#include "cfe_sb_msgdefs.h"
#include "cfe_sb_msgstruct.h"
```

#### 12.168.1 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.

## 12.169 cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_sb_extern_typedefs.h"
#include "cfe_sb_fcncodes.h"
```

### Data Structures

- struct [CFE\\_SB\\_WriteFileInfoCmd\\_Payload](#)  
*Write File Info Command Payload.*
- struct [CFE\\_SB\\_RouteCmd\\_Payload](#)  
*Enable/Disable Route Command Payload.*
- struct [CFE\\_SB\\_HousekeepingTlm\\_Payload](#)
- struct [CFE\\_SB\\_PipeDepthStats](#)  
*SB Pipe Depth Statistics.*
- struct [CFE\\_SB\\_PipeInfoEntry](#)  
*SB Pipe Information File Entry.*
- struct [CFE\\_SB\\_StatsTlm\\_Payload](#)
- struct [CFE\\_SB\\_RoutingFileEntry](#)  
*SB Routing File Entry.*
- struct [CFE\\_SB\\_MsgMapFileEntry](#)  
*SB Map File Entry.*
- struct [CFE\\_SB\\_SingleSubscriptionTlm\\_Payload](#)
- struct [CFE\\_SB\\_SubEntries](#)  
*SB Previous Subscriptions Entry.*
- struct [CFE\\_SB\\_AllSubscriptionsTlm\\_Payload](#)

### Typedefs

- typedef struct [CFE\\_SB\\_WriteFileInfoCmd\\_Payload](#) [CFE\\_SB\\_WriteFileInfoCmd\\_Payload\\_t](#)  
*Write File Info Command Payload.*
- typedef struct [CFE\\_SB\\_RouteCmd\\_Payload](#) [CFE\\_SB\\_RouteCmd\\_Payload\\_t](#)  
*Enable/Disable Route Command Payload.*
- typedef struct [CFE\\_SB\\_HousekeepingTlm\\_Payload](#) [CFE\\_SB\\_HousekeepingTlm\\_Payload\\_t](#)
- typedef struct [CFE\\_SB\\_PipeDepthStats](#) [CFE\\_SB\\_PipeDepthStats\\_t](#)  
*SB Pipe Depth Statistics.*
- typedef struct [CFE\\_SB\\_PipeInfoEntry](#) [CFE\\_SB\\_PipeInfoEntry\\_t](#)  
*SB Pipe Information File Entry.*
- typedef struct [CFE\\_SB\\_StatsTlm\\_Payload](#) [CFE\\_SB\\_StatsTlm\\_Payload\\_t](#)
- typedef struct [CFE\\_SB\\_RoutingFileEntry](#) [CFE\\_SB\\_RoutingFileEntry\\_t](#)  
*SB Routing File Entry.*
- typedef struct [CFE\\_SB\\_MsgMapFileEntry](#) [CFE\\_SB\\_MsgMapFileEntry\\_t](#)  
*SB Map File Entry.*

- typedef struct [CFE\\_SB\\_SingleSubscriptionTlm\\_Payload](#) CFE\_SB\_SingleSubscriptionTlm\_Payload\_t
- typedef struct [CFE\\_SB\\_SubEntries](#) CFE\_SB\_SubEntries\_t
  - SB Previous Subscriptions Entry.*
- typedef struct [CFE\\_SB\\_AllSubscriptionsTlm\\_Payload](#) CFE\_SB\_AllSubscriptionsTlm\_Payload\_t

### 12.169.1 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

### 12.169.2 Typedef Documentation

#### 12.169.2.1 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.

#### 12.169.2.2 CFE\_SB\_HousekeepingTlm\_Payload\_t [typedef struct CFE\\_SB\\_HousekeepingTlm\\_Payload](#) [CFE\\_SB\\_HousekeepingTlm\\_Payload\\_t](#)

**Name** Software Bus task housekeeping Packet

#### 12.169.2.3 CFE\_SB\_MsgMapFileEntry\_t [typedef struct CFE\\_SB\\_MsgMapFileEntry](#) [CFE\\_SB\\_MsgMapFileEntry\\_t](#)

SB Map File Entry.  
Structure of one element of the map information in response to [CFE\\_SB\\_WRITE\\_MAP\\_INFO\\_CC](#)

#### 12.169.2.4 CFE\_SB\_PipeDepthStats\_t [typedef struct CFE\\_SB\\_PipeDepthStats](#) [CFE\\_SB\\_PipeDepthStats\\_t](#)

SB Pipe Depth Statistics.  
Used in SB Statistics Telemetry Packet [CFE\\_SB\\_StatsTlm\\_t](#)

#### 12.169.2.5 CFE\_SB\_PipeInfoEntry\_t [typedef struct CFE\\_SB\\_PipeInfoEntry](#) [CFE\\_SB\\_PipeInfoEntry\\_t](#)

SB Pipe Information File Entry.  
This statistics structure is output as part of the CFE SB "Send Pipe Info" command (CFE\_SB\_SEND\_PIPE\_INFO\_CC). Previous versions of CFE simply wrote the internal CFE\_SB\_PipeD\_t object to the file, but this also contains information such as pointers which are not relevant outside the running CFE process.

By defining the pipe info structure separately, it also provides some independence, such that the internal CFE\_SB\_PipeD\_t definition can evolve without changing the binary format of the information file.

#### 12.169.2.6 CFE\_SB\_RouteCmd\_Payload\_t [typedef struct CFE\\_SB\\_RouteCmd\\_Payload](#) [CFE\\_SB\\_RouteCmd\\_Payload\\_t](#)

Enable/Disable Route Command Payload.  
This structure contains a definition used by two SB commands, 'Enable Route' [CFE\\_SB\\_ENABLE\\_ROUTE\\_CC](#) and 'Disable Route' [CFE\\_SB\\_DISABLE\\_ROUTE\\_CC](#). A route is the destination pipe for a particular message and is therefore defined as a MsgId and Pipelid combination.

**12.169.2.7 CFE\_SB\_RoutingFileEntry\_t** `typedef struct CFE_SB_RoutingFileEntry CFE_SB_RoutingFileEntry_t`  
SB Routing File Entry.

Structure of one element of the routing information in response to [CFE\\_SB\\_WRITE\\_ROUTING\\_INFO\\_CC](#)

**12.169.2.8 CFE\_SB\_SingleSubscriptionTlm\_Payload\_t** `typedef struct CFE_SB_SingleSubscriptionTlm_Payload CFE_SB_SingleSubscriptionTlm_Payload_t`

**Name** SB Subscription Report Packet

This structure defines the pkt sent by SB when a subscription or a request to unsubscribe is received while subscription reporting is enabled. By default subscription reporting is disabled. This feature is intended to be used primarily by Software Bus Networking Application (SBN)

See also

[CFE\\_SB\\_ENABLE\\_SUB\\_REPORTING\\_CC](#), [CFE\\_SB\\_DISABLE\\_SUB\\_REPORTING\\_CC](#)

**12.169.2.9 CFE\_SB\_StatsTlm\_Payload\_t** `typedef struct CFE_SB_StatsTlm_Payload CFE_SB_StatsTlm_Payload_t`

**Name** SB Statistics Telemetry Packet

SB Statistics packet sent in response to [CFE\\_SB\\_SEND\\_SB\\_STATS\\_CC](#)

**12.169.2.10 CFE\_SB\_SubEntries\_t** `typedef struct CFE_SB_SubEntries CFE_SB_SubEntries_t`  
SB Previous Subscriptions Entry.

This structure defines an entry used in the CFE\_SB\_PrevSubsPkt\_t Intended to be used primarily by Software Bus Networking Application (SBN)

Used in structure definition [CFE\\_SB\\_AllSubscriptionsTlm\\_t](#)

**12.169.2.11 CFE\_SB\_WriteFileInfoCmd\_Payload\_t** `typedef struct CFE_SB_WriteFileInfoCmd_Payload CFE_SB_WriteFileInfoCmd_Payload_t`

Write File Info Command Payload.

This structure contains a generic definition used by SB commands that write to a file

## 12.170 cfe/modules/sb/config/default\_cfe\_sb\_msgid\_values.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_sb_topicids.h"
```

### Macros

- `#define CFE_PLATFORM_SB_CMD_MIDVAL(x) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_PLATFORM_CMD_TOPICID_##x##_TOPICID)`
- `#define CFE_PLATFORM_SB_TLM_MIDVAL(x) CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_PLATFORM_TLM_TOPICID_##x##_TOPICID)`

### 12.170.1 Detailed Description

CFE Event Services (CFE\_SB) Application Message IDs

### 12.170.2 Macro Definition Documentation

**12.170.2.1 CFE\_PLATFORM\_SB\_CMD\_MIDVAL** #define CFE\_PLATFORM\_SB\_CMD\_MIDVAL( x ) CFE\_PLATFORM\_CMD\_TOPICID\_TO\_MIDV(CFE\_MISSION\_SB\_##x##\_TOPICID)  
Definition at line 29 of file default\_cfe\_sb\_msgid\_values.h.

**12.170.2.2 CFE\_PLATFORM\_SB\_TLM\_MIDVAL** #define CFE\_PLATFORM\_SB\_TLM\_MIDVAL( x ) CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV(CFE\_MISSION\_SB\_##x##\_TOPICID)  
Definition at line 30 of file default\_cfe\_sb\_msgid\_values.h.

## 12.171 cfe/modules/sb/config/default\_cfe\_sb\_msgids.h File Reference

```
#include "cfe_sb_msgid_values.h"
```

### Macros

- #define CFE\_SB\_CMD\_MID CFE\_PLATFORM\_SB\_CMD\_MIDVAL(CMD) /\* Default=0x1803 \*/
- #define CFE\_SB\_SEND\_HK\_MID CFE\_PLATFORM\_SB\_CMD\_MIDVAL(SEND\_HK) /\* Default=0x180B \*/
- #define CFE\_SB\_SUB\_RPT\_CTRL\_MID CFE\_PLATFORM\_SB\_CMD\_MIDVAL(SUB\_RPT\_CTRL) /\* Default=0x180E \*/
- #define CFE\_SB\_HK\_TLM\_MID CFE\_PLATFORM\_SB\_TLM\_MIDVAL(HK\_TLM) /\* Default=0x0803 \*/
- #define CFE\_SB\_STATS\_TLM\_MID CFE\_PLATFORM\_SB\_TLM\_MIDVAL(STATS\_TLM) /\* Default=0x080A \*/
- #define CFE\_SB\_ALLSUBS\_TLM\_MID CFE\_PLATFORM\_SB\_TLM\_MIDVAL(ALLSUBS\_TLM) /\* Default=0x080D \*/
- #define CFE\_SB\_ONESUB\_TLM\_MID CFE\_PLATFORM\_SB\_TLM\_MIDVAL(ONESUB\_TLM) /\* Default=0x080E \*/

### 12.171.1 Detailed Description

CFE Event Services (CFE\_SB) Application Message IDs

### 12.171.2 Macro Definition Documentation

**12.171.2.1 CFE\_SB\_ALLSUBS\_TLM\_MID** #define CFE\_SB\_ALLSUBS\_TLM\_MID CFE\_PLATFORM\_SB\_TLM\_MIDVAL(ALLSUBS\_TLM) /\* Default=0x080D \*/  
Definition at line 40 of file default\_cfe\_sb\_msgids.h.

**12.171.2.2 CFE\_SB\_CMD\_MID** #define CFE\_SB\_CMD\_MID CFE\_PLATFORM\_SB\_CMD\_MIDVAL(CMD) /\* Default=0x1803 \*/  
Definition at line 31 of file default\_cfe\_sb\_msgids.h.

**12.171.2.3 CFE\_SB\_HK\_TLM\_MID** #define CFE\_SB\_HK\_TLM\_MID CFE\_PLATFORM\_SB\_TLM\_MIDVAL(HK\_TLM) /\* Default=0x0803 \*/  
Definition at line 38 of file default\_cfe\_sb\_msgids.h.

**12.171.2.4 CFE\_SB\_ONESUB\_TLM\_MID** #define CFE\_SB\_ONESUB\_TLM\_MID CFE\_PLATFORM\_SB\_TLM\_MIDVAL(ONESUB\_TLM) /\* Default=0x080E \*/  
Definition at line 41 of file default\_cfe\_sb\_msgids.h.

```
12.171.2.5 CFE_SB_SEND_HK_MID #define CFE_SB_SEND_HK_MID CFE_PLATFORM_SB_CMD_MIDVAL(SEND_HK)
/* Default=0x180B */
Definition at line 32 of file default_cfe_sb_msgids.h.
```

```
12.171.2.6 CFE_SB_STATS_TLM_MID #define CFE_SB_STATS_TLM_MID CFE_PLATFORM_SB_TLM_MIDVAL(STATS↔_
_TLM) /* Default=0x080A */
Definition at line 39 of file default_cfe_sb_msgids.h.
```

```
12.171.2.7 CFE_SB_SUB_RPT_CTRL_MID #define CFE_SB_SUB_RPT_CTRL_MID CFE_PLATFORM_SB_CMD_MIDVAL(SUB↔_
_RPT_CTRL) /* Default=0x180E */
Definition at line 33 of file default_cfe_sb_msgids.h.
```

## 12.172 cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h File Reference

```
#include "cfe_sb_msgdefs.h"
#include "cfe_msg_hdr.h"
```

### Data Structures

- struct [CFE\\_SB\\_NoopCmd](#)
- struct [CFE\\_SB\\_ResetCountersCmd](#)
- struct [CFE\\_SB\\_EnableSubReportingCmd](#)
- struct [CFE\\_SB\\_DisableSubReportingCmd](#)
- struct [CFE\\_SB\\_SendSbStatsCmd](#)
- struct [CFE\\_SB\\_SendPrevSubsCmd](#)
- struct [CFE\\_SB\\_SendHkCmd](#)
- struct [CFE\\_SB\\_WriteRoutingInfoCmd](#)
- struct [CFE\\_SB\\_WritePipeInfoCmd](#)
- struct [CFE\\_SB\\_WriteMapInfoCmd](#)
- struct [CFE\\_SB\\_EnableRouteCmd](#)
- struct [CFE\\_SB\\_DisableRouteCmd](#)
- struct [CFE\\_SB\\_HousekeepingTlm](#)
- struct [CFE\\_SB\\_StatsTlm](#)
- struct [CFE\\_SB\\_SingleSubscriptionTlm](#)
- struct [CFE\\_SB\\_AllSubscriptionsTlm](#)

### Typedefs

- typedef struct [CFE\\_SB\\_NoopCmd](#) [CFE\\_SB\\_NoopCmd\\_t](#)
- typedef struct [CFE\\_SB\\_ResetCountersCmd](#) [CFE\\_SB\\_ResetCountersCmd\\_t](#)
- typedef struct [CFE\\_SB\\_EnableSubReportingCmd](#) [CFE\\_SB\\_EnableSubReportingCmd\\_t](#)
- typedef struct [CFE\\_SB\\_DisableSubReportingCmd](#) [CFE\\_SB\\_DisableSubReportingCmd\\_t](#)
- typedef struct [CFE\\_SB\\_SendSbStatsCmd](#) [CFE\\_SB\\_SendSbStatsCmd\\_t](#)
- typedef struct [CFE\\_SB\\_SendPrevSubsCmd](#) [CFE\\_SB\\_SendPrevSubsCmd\\_t](#)
- typedef struct [CFE\\_SB\\_SendHkCmd](#) [CFE\\_SB\\_SendHkCmd\\_t](#)
- typedef struct [CFE\\_SB\\_WriteRoutingInfoCmd](#) [CFE\\_SB\\_WriteRoutingInfoCmd\\_t](#)
- typedef struct [CFE\\_SB\\_WritePipeInfoCmd](#) [CFE\\_SB\\_WritePipeInfoCmd\\_t](#)
- typedef struct [CFE\\_SB\\_WriteMapInfoCmd](#) [CFE\\_SB\\_WriteMapInfoCmd\\_t](#)
- typedef struct [CFE\\_SB\\_EnableRouteCmd](#) [CFE\\_SB\\_EnableRouteCmd\\_t](#)

- `typedef struct CFE_SB_DisableRouteCmd CFE_SB_DisableRouteCmd_t`
- `typedef struct CFE_SB_HousekeepingTlm CFE_SB_HousekeepingTlm_t`
- `typedef struct CFE_SB_StatsTlm CFE_SB_StatsTlm_t`
- `typedef struct CFE_SB_SingleSubscriptionTlm CFE_SB_SingleSubscriptionTlm_t`
- `typedef struct CFE_SB_AllSubscriptionsTlm CFE_SB_AllSubscriptionsTlm_t`

### 12.172.1 Detailed Description

Purpose: cFE Executive Services (SB) Command and Telemetry packet definition file.

### 12.172.2 Typedef Documentation

**12.172.2.1 CFE\_SB\_AllSubscriptionsTlm\_t** `typedef struct CFE_SB_AllSubscriptionsTlm CFE_SB_AllSubscriptionsTlm_t`

**12.172.2.2 CFE\_SB\_DisableRouteCmd\_t** `typedef struct CFE_SB_DisableRouteCmd CFE_SB_DisableRouteCmd_t`

**12.172.2.3 CFE\_SB\_DisableSubReportingCmd\_t** `typedef struct CFE_SB_DisableSubReportingCmd CFE_SB_DisableSubReportin`

**12.172.2.4 CFE\_SB\_EnableRouteCmd\_t** `typedef struct CFE_SB_EnableRouteCmd CFE_SB_EnableRouteCmd_t`

**12.172.2.5 CFE\_SB\_EnableSubReportingCmd\_t** `typedef struct CFE_SB_EnableSubReportingCmd CFE_SB_EnableSubReportin`

**12.172.2.6 CFE\_SB\_HousekeepingTlm\_t** `typedef struct CFE_SB_HousekeepingTlm CFE_SB_HousekeepingTlm_t`

**12.172.2.7 CFE\_SB\_NoopCmd\_t** `typedef struct CFE_SB_NoopCmd CFE_SB_NoopCmd_t`

**12.172.2.8 CFE\_SB\_ResetCountersCmd\_t** `typedef struct CFE_SB_ResetCountersCmd CFE_SB_ResetCountersCmd_t`

**12.172.2.9 CFE\_SB\_SendHkCmd\_t** `typedef struct CFE_SB_SendHkCmd CFE_SB_SendHkCmd_t`

**12.172.2.10 CFE\_SB\_SendPrevSubsCmd\_t** `typedef struct CFE_SB_SendPrevSubsCmd CFE_SB_SendPrevSubsCmd_t`

**12.172.2.11 CFE\_SB\_SendSbStatsCmd\_t** `typedef struct CFE_SB_SendSbStatsCmd CFE_SB_SendSbStatsCmd_t`

**12.172.2.12 CFE\_SB\_SingleSubscriptionTlm\_t** `typedef struct CFE_SB_SingleSubscriptionTlm CFE_SB_SingleSubscription`

12.172.2.13 **CFE\_SB\_StatsTlm\_t** `typedef struct CFE_SB_StatsTlm CFE_SB_StatsTlm_t`

12.172.2.14 **CFE\_SB\_WriteMapInfoCmd\_t** `typedef struct CFE_SB_WriteMapInfoCmd CFE_SB_WriteMapInfoCmd_t`

12.172.2.15 **CFE\_SB\_WritePipeInfoCmd\_t** `typedef struct CFE_SB_WritePipeInfoCmd CFE_SB_WritePipeInfoCmd_t`

12.172.2.16 **CFE\_SB\_WriteRoutingInfoCmd\_t** `typedef struct CFE_SB_WriteRoutingInfoCmd CFE_SB_WriteRoutingInfoCmd_t`

## 12.173 cfe/modules/sb/config/default\_cfe\_sb\_platform\_cfg.h File Reference

```
#include "cfe_sb_mission_cfg.h"
#include "cfe_sb_internal_cfg.h"
```

### 12.173.1 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.

## 12.174 cfe/modules/sb/config/default\_cfe\_sb\_topicid\_values.h File Reference

### Macros

- `#define CFE_MISSION_SB_TIDVAL(x) DEFAULT_CFE_MISSION_SB_##x##_TOPICID`

### 12.174.1 Detailed Description

CFE Software Bus (CFE\_SB) Application Topic IDs

### 12.174.2 Macro Definition Documentation

12.174.2.1 **CFE\_MISSION\_SB\_TIDVAL** `#define CFE_MISSION_SB_TIDVAL(`  
`x ) DEFAULT_CFE_MISSION_SB_##x##_TOPICID`

Definition at line 26 of file default\_cfe\_sb\_topicid\_values.h.

## 12.175 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

- #define `CFE_SB_ENBL RTE1_EID` 33  
*SB Send Statistics Command Success Event ID.*
- #define `CFE_SB_ENBL RTE2_EID` 34  
*SB Enable Route Command Invalid MsgId/PipeID Pair Event ID.*
- #define `CFE_SB_ENBL RTE3_EID` 35  
*SB Enable Route Command Success Event ID.*
- #define `CFE_SB_DSBL RTE1_EID` 36  
*SB Disable Route Command Invalid MsgId/PipeID Pair Event ID.*
- #define `CFE_SB_DSBL RTE2_EID` 37  
*SB Disable Route Command Success Event ID.*
- #define `CFE_SB_DSBL RTE3_EID` 38  
*SB Disable Route Command Invalid MsgId or Pipe Event ID.*
- #define `CFE_SB SND RTG EID` 39  
*SB File Write Success Event ID.*
- #define `CFE_SB SND RTG ERR1_EID` 40  
*SB File Write Create File Failure Event ID.*
- #define `CFE_SB_BAD_CMD_CODE_EID` 42  
*SB Invalid Command Code Received Event ID.*
- #define `CFE_SB_BAD_MSGID_EID` 43  
*SB Invalid Message ID Received Event ID.*
- #define `CFE_SB_FULL_SUB_PKT_EID` 44  
*SB Send Previous Subscriptions Command Full Packet Sent Event ID.*
- #define `CFE_SB_PART_SUB_PKT_EID` 45  
*SB Send Previous Subscriptions Command Partial Packet Sent Event ID.*
- #define `CFE_SB_DEL_PIPE_ERR1_EID` 46  
*SB Pipe Delete API Bad Argument Event ID.*
- #define `CFE_SB_PIPE_DELETED_EID` 47  
*SB Pipe Delete API Success Event ID.*
- #define `CFE_SB_SUBSCRIPTION_REMOVED_EID` 48  
*SB Unsubscribe API Success Event ID.*
- #define `CFE_SB_FILEWRITE_ERR_EID` 49  
*SB File Write Failed Event ID.*
- #define `CFE_SB_SUB_INV_PIPE_EID` 50  
*SB Subscribe API Invalid Pipe Event ID.*
- #define `CFE_SB_SUB_INV_CALLER_EID` 51  
*SB Subscribe API Not Owner Event ID.*
- #define `CFE_SB_UNSUB_INV_PIPE_EID` 52  
*SB Unsubscribe API Invalid Pipe Event ID.*
- #define `CFE_SB_UNSUB_INV_CALLER_EID` 53  
*SB Unsubscribe API Not Owner Event ID.*
- #define `CFE_SB_DEL_PIPE_ERR2_EID` 54  
*SB Delete Pipe API Not Owner Event ID.*
- #define `CFE_SB_SETPIPEOPTS_ID_ERR_EID` 55  
*SB Set Pipe Opt API Invalid Pipe Event ID.*
- #define `CFE_SB_SETPIPEOPTS_OWNER_ERR_EID` 56  
*SB Set Pipe Opt API Not Owner Event ID.*
- #define `CFE_SB_SETPIPEOPTS_EID` 57  
*SB Set Pipe Opt API Success Event ID.*
- #define `CFE_SB_GETPIPEOPTS_ID_ERR_EID` 58  
*SB Get Pipe Opt API Invalid Pipe Event ID.*
- #define `CFE_SB_GETPIPEOPTS_PTR_ERR_EID` 59  
*SB Get Pipe Opt API Invalid Pointer Event ID.*
- #define `CFE_SB_GETPIPEOPTS_EID` 60  
*SB Get Pipe Opt 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.*
- #define [CFE\\_SB\\_SEND\\_MESSAGE\\_INTEGRITY\\_FAIL\\_EID](#) 71  
*SB integrity actions on transmit message failure event.*
- #define [CFE\\_SB\\_RCV\\_MESSAGE\\_INTEGRITY\\_FAIL\\_EID](#) 72  
*SB validation of received message failure event.*

#### 12.175.1 Detailed Description

cFE Software Bus Services Event IDs

#### 12.175.2 Macro Definition Documentation

**12.175.2.1 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.

**12.175.2.2 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.

**12.175.2.3 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.

**12.175.2.4 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.

**12.175.2.5 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.

**12.175.2.6 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.

**12.175.2.7 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.

**12.175.2.8 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.

**12.175.2.9 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.

**12.175.2.10 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.

**12.175.2.11 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.

**12.175.2.12 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.

**12.175.2.13 CFE\_SB\_DSBL RTE1\_EID** #define CFE\_SB\_DSBL RTE1\_EID 36  
SB Disable Route Command Invalid MsgId/Pipeld 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.

**12.175.2.14 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.

**12.175.2.15 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.

**12.175.2.16 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.

**12.175.2.17 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.

**12.175.2.18 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.

**12.175.2.19 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.

**12.175.2.20 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.

**12.175.2.21 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.

**12.175.2.22 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.

**12.175.2.23 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.

**12.175.2.24 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.

**12.175.2.25 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.

**12.175.2.26 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.

**12.175.2.27 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.

**12.175.2.28 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.

**12.175.2.29 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.

**12.175.2.30 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.

**12.175.2.31 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.

**12.175.2.32 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.

**12.175.2.33 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.

**12.175.2.34 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.

**12.175.2.35 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.

**12.175.2.36 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.

**12.175.2.37 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.

**12.175.2.38 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.

**12.175.2.39 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.

**12.175.2.40 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.

**12.175.2.41 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.

**12.175.2.42 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.

**12.175.2.43 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.

**12.175.2.44 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.

**12.175.2.45 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.

**12.175.2.46 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.

**12.175.2.47 CFE\_SB\_RCV\_MESSAGE\_INTEGRITY\_FAIL\_EID** #define CFE\_SB\_RCV\_MESSAGE\_INTEGRITY\_FAIL\_EID 72  
SB validation of received message failure event.

Type: ERROR

Cause:

A CFE SB receive transaction has rejected a message due to failure of the associated message integrity action(s).  
Definition at line 785 of file cfe\_sb\_eventids.h.

**12.175.2.48 CFE\_SB\_SEND\_BAD\_ARG\_EID** #define CFE\_SB\_SEND\_BAD\_ARG\_EID 13  
SB Transmit API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Transmit API failed due to an invalid input argument.  
Definition at line 177 of file cfe\_sb\_eventids.h.

**12.175.2.49 CFE\_SB\_SEND\_INV\_MSGID\_EID** #define CFE\_SB\_SEND\_INV\_MSGID\_EID 21  
SB Transmit API Invalid MsgId Event ID.

Type: ERROR

Cause:

An SB Transmit API was called with an invalid message ID.  
Definition at line 266 of file cfe\_sb\_eventids.h.

**12.175.2.50 CFE\_SB\_SEND\_MESSAGE\_INTEGRITY\_FAIL\_EID** #define CFE\_SB\_SEND\_MESSAGE\_INTEGRITY\_FAIL\_EID 71  
SB integrity actions on transmit message failure event.

Type: ERROR

Cause:

A CFE SB transmit transaction has rejected a message due to failure of the associated message integrity action(s).  
Definition at line 773 of file cfe\_sb\_eventids.h.

**12.175.2.51 CFE\_SB\_SEND\_NO\_SUBS\_EID** #define CFE\_SB\_SEND\_NO\_SUBS\_EID 14  
SB Transmit API No MsgId Subscribers Event ID.

Type: INFORMATION

Cause:

An SB Transmit API was called with a Message ID with no subscriptions.  
Definition at line 188 of file cfe\_sb\_eventids.h.

**12.175.2.52 CFE\_SB\_SETPPIPEOPTS\_EID** #define CFE\_SB\_SETPPIPEOPTS\_EID 57  
SB Set Pipe Opt API Success Event ID.

Type: DEBUG

Cause:

[CFE\\_SB\\_SetPipeOpt](#) success.  
Definition at line 628 of file cfe\_sb\_eventids.h.

**12.175.2.53 CFE\_SB\_SETPPIPEOPTS\_ID\_ERR\_EID** #define CFE\_SB\_SETPPIPEOPTS\_ID\_ERR\_EID 55  
SB Set Pipe Opt API Invalid Pipe Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_SetPipeOpt](#) API failure due to an invalid pipe ID  
Definition at line 606 of file cfe\_sb\_eventids.h.

**12.175.2.54 CFE\_SB\_SETPPIPEOPTS\_OWNER\_ERR\_EID** #define CFE\_SB\_SETPPIPEOPTS\_OWNER\_ERR\_EID 56  
SB Set Pipe Opt API Not Owner Event ID.

Type: ERROR

Cause:

[CFE\\_SB\\_SetPipeOpt](#) API failure due to not being the pipe owner.  
Definition at line 617 of file cfe\_sb\_eventids.h.

**12.175.2.55 CFE\_SB SND RTG EID** #define CFE\_SB\_SND\_RTG\_EID 39  
SB File Write Success Event ID.

Type: DEBUG

Cause:

An SB file write successfully completed. OVERLOADED  
Definition at line 438 of file cfe\_sb\_eventids.h.

**12.175.2.56 CFE\_SB SND RTG ERR1 EID** #define CFE\_SB\_SND\_RTG\_ERR1\_EID 40  
SB File Write Create File Failure Event ID.

Type: ERROR

Cause:

An SB file write failure due to file creation error. OVERLOADED  
Definition at line 449 of file cfe\_sb\_eventids.h.

**12.175.2.57 CFE\_SB SND STATS EID** #define CFE\_SB\_SND\_STATS\_EID 32  
SB Send Statistics Command Success Event ID.

Type: DEBUG

Cause:

[SB Send Statistics Command](#) success.  
Definition at line 357 of file cfe\_sb\_eventids.h.

**12.175.2.58 CFE\_SB SUB ARG ERR EID** #define CFE\_SB\_SUB\_ARG\_ERR\_EID 6  
SB Subscribe API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Subscribe API failed due to an invalid input argument.  
Definition at line 97 of file cfe\_sb\_eventids.h.

**12.175.2.59 CFE\_SB\_SUB\_INV\_CALLER\_EID** #define CFE\_SB\_SUB\_INV\_CALLER\_EID 51  
SB Subscribe API Not Owner Event ID.

Type: ERROR

Cause:

An SB Subscribe API failed due to not being the pipe owner.  
Definition at line 562 of file cfe\_sb\_eventids.h.

**12.175.2.60 CFE\_SB\_SUB\_INV\_PIPE\_EID** #define CFE\_SB\_SUB\_INV\_PIPE\_EID 50  
SB Subscribe API Invalid Pipe Event ID.

Type: ERROR

Cause:

An SB Subscribe API failed due to an invalid pipe ID.  
Definition at line 551 of file cfe\_sb\_eventids.h.

**12.175.2.61 CFE\_SB\_SUBSCRIPTION\_RCVD\_EID** #define CFE\_SB\_SUBSCRIPTION\_RCVD\_EID 10  
SB Subscribe API Success Event ID.

Type: DEBUG

Cause:

An SB Subscribe API completed successfully.  
Definition at line 144 of file cfe\_sb\_eventids.h.

**12.175.2.62 CFE\_SB\_SUBSCRIPTION\_REMOVED\_EID** #define CFE\_SB\_SUBSCRIPTION\_REMOVED\_EID 48  
SB Unsubscribe API Success Event ID.

Type: DEBUG

Cause:

An SB Unsubscribe API successfully completed.  
Definition at line 529 of file cfe\_sb\_eventids.h.

**12.175.2.63 CFE\_SB\_SUBSCRIPTION\_RPT\_EID** #define CFE\_SB\_SUBSCRIPTION\_RPT\_EID 22  
SB Subscription Report Sent Event ID.

Type: DEBUG

Cause:

SB Subscription Report sent in response to a successful subscription.  
Definition at line 277 of file cfe\_sb\_eventids.h.

**12.175.2.64 CFE\_SB\_UNSUB\_ARG\_ERR\_EID** #define CFE\_SB\_UNSUB\_ARG\_ERR\_EID 11  
SB Unsubscribe API Bad Argument Event ID.

Type: ERROR

Cause:

An SB Unsubscribe API failed due to an invalid input argument.  
Definition at line 155 of file cfe\_sb\_eventids.h.

**12.175.2.65 CFE\_SB\_UNSUB\_INV\_CALLER\_EID** #define CFE\_SB\_UNSUB\_INV\_CALLER\_EID 53  
SB Unsubscribe API Not Owner Event ID.

Type: ERROR

Cause:

An SB Unsubscribe API failed due to not being the pipe owner.  
Definition at line 584 of file cfe\_sb\_eventids.h.

**12.175.2.66 CFE\_SB\_UNSUB\_INV\_PIPE\_EID** #define CFE\_SB\_UNSUB\_INV\_PIPE\_EID 52  
SB Unsubscribe API Invalid Pipe Event ID.

Type: ERROR

Cause:

An SB Unsubscribe API failed due to an invalid pipe ID.  
Definition at line 573 of file cfe\_sb\_eventids.h.

**12.175.2.67 CFE\_SB\_UNSUB\_NO\_SUBS\_EID** #define CFE\_SB\_UNSUB\_NO\_SUBS\_EID 12  
SB Unsubscribe API No MsgId Subscription Event ID.

Type: INFORMATION

Cause:

An SB Unsubscribe API was called with a Message ID that wasn't subscribed on the pipe Definition at line 166 of file cfe\_sb\_eventids.h.

## 12.176 cfe/modules/sb/fsw/inc/cfe\_sb\_fcncodes.h File Reference

```
#include "cfe_sb_fcncode_values.h"
```

### Macros

- #define CFE\_SB\_NOOP\_CC CFE\_SB\_CCVAL(NOP)
- #define CFE\_SB\_RESET\_COUNTERS\_CC CFE\_SB\_CCVAL(RESET\_COUNTERS)
- #define CFE\_SB\_SEND\_SB\_STATS\_CC CFE\_SB\_CCVAL(SEND\_SB\_STATS)
- #define CFE\_SB\_WRITE\_ROUTING\_INFO\_CC CFE\_SB\_CCVAL(WRITE\_ROUTING\_INFO)
- #define CFE\_SB\_ENABLE\_ROUTE\_CC CFE\_SB\_CCVAL(ENABLE\_ROUTE)
- #define CFE\_SB\_DISABLE\_ROUTE\_CC CFE\_SB\_CCVAL(DISABLE\_ROUTE)
- #define CFE\_SB\_WRITE\_PIPE\_INFO\_CC CFE\_SB\_CCVAL(WRITE\_PIPE\_INFO)
- #define CFE\_SB\_WRITE\_MAP\_INFO\_CC CFE\_SB\_CCVAL(WRITE\_MAP\_INFO)
- #define CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC CFE\_SB\_CCVAL(ENABLE\_SUB\_REPORTING)
- #define CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC CFE\_SB\_CCVAL(DISABLE\_SUB\_REPORTING)
- #define CFE\_SB\_SEND\_PREV\_SUBS\_CC CFE\_SB\_CCVAL(SENDSUBS)

### 12.176.1 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.

### 12.176.2 Macro Definition Documentation

**12.176.2.1 CFE\_SB\_DISABLE\_ROUTE\_CC** #define CFE\_SB\_DISABLE\_ROUTE\_CC CFE\_SB\_CCVAL(DISABLE\_ROUTE)

**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 273 of file cfe\_sb\_fcncodes.h.

**12.176.2.2 CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC** #define CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC [CFE\\_SB\\_CCVAL](#) (DISABLE←\_SUB\_REPORTING)

**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 430 of file cfe\_sb\_fcncodes.h.

### 12.176.2.3 CFE\_SB\_ENABLE\_ROUTE\_CC #define CFE\_SB\_ENABLE\_ROUTE\_CC CFE\_SB\_CCVAL (ENABLE\_ROUTE)

**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 232 of file cfe\_sb\_fcncodes.h.

**12.176.2.4 CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC** #define CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC CFE\_SB\_CCVVAL (ENABLE↔\_SUB\_REPORTING)

**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 397 of file cfe\_sb\_fcncodes.h.

**12.176.2.5 CFE\_SB\_NOOP\_CC** #define CFE\_SB\_NOOP\_CC CFE\_SB\_CCVVAL (NOOP)

**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 68 of file cfe\_sb\_fcncodes.h.

**12.176.2.6 CFE\_SB\_RESET\_COUNTERS\_CC** #define CFE\_SB\_RESET\_COUNTERS\_CC CFE\_SB\_CCVAL(RESET\_←  
COUNTERS)

**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 115 of file cfe\_sb\_fcncodes.h.

**12.176.2.7 CFE\_SB\_SEND\_PREV\_SUBS\_CC** #define CFE\_SB\_SEND\_PREV\_SUBS\_CC CFE\_SB\_CCVAL (SEND\_<br>PREV\_SUBS)

### 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 462 of file cfe\_sb\_fcncodes.h.

**12.176.2.8 CFE\_SB\_SEND\_SB\_STATS\_CC** #define CFE\_SB\_SEND\_SB\_STATS\_CC CFE\_SB\_CCVAL (SEND\_SB←  
STATS)

**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 149 of file [cfe\\_sb\\_fcncodes.h](#).

**12.176.2.9 CFE\_SB\_WRITE\_MAP\_INFO\_CC** #define CFE\_SB\_WRITE\_MAP\_INFO\_CC CFE\_SB\_CCVAL (WRITE\_MAP←  
\_INFO)

**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 364 of file cfe\_sb\_fcncodes.h.

**12.176.2.10 CFE\_SB\_WRITE\_PIPE\_INFO\_CC** #define CFE\_SB\_WRITE\_PIPE\_INFO\_CC [CFE\\_SB\\_CCVAL](#)(WRITE\_<  
PIPE\_INFO>)

**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 318 of file cfe\_sb\_fcncodes.h.

**12.176.2.11 CFE\_SB\_WRITE\_ROUTING\_INFO\_CC** #define CFE\_SB\_WRITE\_ROUTING\_INFO\_CC `CFE_SB_CCVAL`(WRITE←\_ROUTING\_INFO)

**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 194 of file cfe\_sb\_fcncodes.h.

## 12.177 cfe/modules/sb/fsw/inc/cfe\_sb\_interface\_cfg.h File Reference

```
#include "cfe_sb_interface_cfg_values.h"
```

### Macros

- #define CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE CFE\_MISSION\_SB\_CFGVAL(MAX\_SB\_MSG\_SIZE)
- #define DEFAULT\_CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE 32768
- #define CFE\_MISSION\_SB\_MAX\_PIPES CFE\_MISSION\_SB\_CFGVAL(MAX\_PIPES)
- #define DEFAULT\_CFE\_MISSION\_SB\_MAX\_PIPES 32
- #define CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT CFE\_MISSION\_SB\_CFGVAL(SUB\_ENTRIES\_PER\_PKT)
- #define DEFAULT\_CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT 20

### 12.177.1 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.

## 12.177.2 Macro Definition Documentation

**12.177.2.1 CFE\_MISSION\_SB\_MAX\_PIPES** #define CFE\_MISSION\_SB\_MAX\_PIPES CFE\_MISSION\_SB\_CFGVAL(MAX←\_PIPES)

**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 70 of file cfe\_sb\_interface\_cfg.h.

**12.177.2.2 CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE** #define CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE CFE\_MISSION\_SB\_CFGVAL(MAX←\_SB\_MSG\_SIZE)

**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 52 of file cfe\_sb\_interface\_cfg.h.

**12.177.2.3 CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT** #define CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT CFE\_MISSION\_SB\_CFGVAL(SUB\_ENTRIES\_PER\_PKT)

**Purpose** Maximum Number of subscription entries per subscription report packet

**Description:**

The subscription report will group up to this number of entries into each TLM packet

**Limits**

Must not cause the size of the subscription report telemetry packet to exceed mission limits

Definition at line 82 of file cfe\_sb\_interface\_cfg.h.

**12.177.2.4 DEFAULT\_CFE\_MISSION\_SB\_MAX\_PIPES** #define DEFAULT\_CFE\_MISSION\_SB\_MAX\_PIPES 32  
Definition at line 71 of file cfe\_sb\_interface\_cfg.h.

**12.177.2.5 DEFAULT\_CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE** #define DEFAULT\_CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE 32768  
MSG\_SIZE 32768  
Definition at line 53 of file cfe\_sb\_interface\_cfg.h.

**12.177.2.6 DEFAULT\_CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT** #define DEFAULT\_CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT 20  
Definition at line 83 of file cfe\_sb\_interface\_cfg.h.

## 12.178 cfe/modules/sb/fsw/inc/cfe\_sb\_internal\_cfg.h File Reference

```
#include "cfe_sb_internal_cfg_values.h"
```

### Macros

- #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS CFE\_PLATFORM\_SB\_CFGVAL(MAX\_MSG\_IDS)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256
- #define CFE\_PLATFORM\_SB\_MAX\_PIPES CFE\_PLATFORM\_SB\_CFGVAL(MAX\_PIPES)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_PIPES 64
- #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT CFE\_PLATFORM\_SB\_CFGVAL(MAX\_DEST\_PER\_PKT)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT CFE\_PLATFORM\_SB\_CFGVAL(DEFAULT\_MSG\_LIMIT)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4
- #define CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES CFE\_PLATFORM\_SB\_CFGVAL(BUF\_MEMORY\_BYTES)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES 524288
- #define CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID CFE\_PLATFORM\_SB\_CFGVAL(HIGHEST\_VALID\_MSGID)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID 0x1FFF
- #define CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME CFE\_PLATFORM\_SB\_CFGVAL(DEFAULT\_ROUTING\_FILENAME)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME "/ram/cfe\_sb\_route.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME CFE\_PLATFORM\_SB\_CFGVAL(DEFAULT\_PIPE\_FILENAME)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME "/ram/cfe\_sb\_pipe.dat"
- #define CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME CFE\_PLATFORM\_SB\_CFGVAL(DEFAULT\_MAP\_FILENAME)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME "/ram/cfe\_sb\_msgmap.dat"
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT1)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_SB\_SEND\_NO\_SUBS\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK1)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_EVS\_FIRST\_4\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT2)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_SB\_DUP\_SUBSCRIPTION\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK2)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_EVS\_FIRST\_4\_STOP

- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT3 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT3)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT3 CFE\_SB\_MSGID\_LIM\_ERR\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK3)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_EVS\_FIRST\_16\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT4 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT4)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT4 CFE\_SB\_Q\_FULL\_ERR\_EID
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK4)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_EVS\_FIRST\_16\_STOP
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT5 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT5)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT5 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK5)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT6 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT6)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT6 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK6)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT7 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT7)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT7 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK7)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT8 CFE\_PLATFORM\_SB\_CFGVAL(FILTERED\_EVENT8)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT8 0
- #define CFE\_PLATFORM\_SB\_FILTER\_MASK8 CFE\_PLATFORM\_SB\_CFGVAL(FILTER\_MASK8)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK8 CFE\_EVS\_NO\_FILTER
- #define CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS CFE\_PLATFORM\_SB\_CFGVAL(POOL\_MAX\_BUCKETS)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS 17
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_01)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 8
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_02)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 16
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_03)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 20
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_04)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 36
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_05)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 64
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_06)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 96
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_07)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 128
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_08)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 160

- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_09)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 256
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_10)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 512
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_11)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 1024
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_12)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 2048
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_13)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 4096
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_14)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 8192
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_15)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 16384
- #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_16)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 32768
- #define CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE CFE\_PLATFORM\_SB\_CFGVAL(MAX\_BLOCK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE (CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE + 128)
- #define CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY CFE\_PLATFORM\_SB\_CFGVAL(START\_TASK\_PRIORITY)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY 64
- #define CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_SB\_CFGVAL(START\_TASK\_STACK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

### 12.178.1 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.

### 12.178.2 Macro Definition Documentation

**12.178.2.1 CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES** #define CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES CFE\_PLATFORM\_SB\_CFGV<sub>1</sub>\_MEMORY\_BYTES)

**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 129 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.2 CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME CFE\_PLATFORM\_SB\_CFGVAL(DEFAULT\_MAP\_FILENAME)

**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 203 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.3 CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT** #define CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT CFE\_PLATFORM\_SB\_CFGV<sub>1</sub>\_MSG\_LIMIT)

**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 106 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.4 CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_←  
FILENAME CFE\_PLATFORM\_SB\_CFGVAL(DEFAULT\_PIPE\_FILENAME)

**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 185 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.5 CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME** #define CFE\_PLATFORM\_SB\_DEFAULT\_←  
ROUTING\_FILENAME CFE\_PLATFORM\_SB\_CFGVAL(DEFAULT\_ROUTING\_FILENAME)

**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 170 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.6 CFE\_PLATFORM\_SB\_FILTER\_MASK1** #define CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_PLATFORM\_SB\_CFGVAL(FILTER←  
\_MASK1)

Definition at line 224 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.7 CFE\_PLATFORM\_SB\_FILTER\_MASK2** #define CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_PLATFORM\_SB\_CFGVAL(FILTER←  
\_MASK2)

Definition at line 230 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.8 CFE\_PLATFORM\_SB\_FILTER\_MASK3** #define CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_PLATFORM\_SB\_CFGVAL(FILTER←  
\_MASK3)

Definition at line 236 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.9 CFE\_PLATFORM\_SB\_FILTER\_MASK4** #define CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_PLATFORM\_SB\_CFGVAL(FILTER←  
\_MASK4)

Definition at line 242 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.10 CFE\_PLATFORM\_SB\_FILTER\_MASK5** #define CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_PLATFORM\_SB\_CFGVAL(FILTER  
\_MASK5)

Definition at line 248 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.11 CFE\_PLATFORM\_SB\_FILTER\_MASK6** #define CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_PLATFORM\_SB\_CFGVAL(FILTER  
\_MASK6)

Definition at line 254 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.12 CFE\_PLATFORM\_SB\_FILTER\_MASK7** #define CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_PLATFORM\_SB\_CFGVAL(FILTER  
\_MASK7)

Definition at line 260 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.13 CFE\_PLATFORM\_SB\_FILTER\_MASK8** #define CFE\_PLATFORM\_SB\_FILTER\_MASK8 CFE\_PLATFORM\_SB\_CFGVAL(FILTER  
\_MASK8)

Definition at line 266 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.14 CFE\_PLATFORM\_SB\_FILTERED\_EVENT1** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT1 CFE\_PLATFORM\_SB\_CFGVAL(  
\_EVENT1)

#### 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 221 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.15 CFE\_PLATFORM\_SB\_FILTERED\_EVENT2** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT2 CFE\_PLATFORM\_SB\_CFGVAL(  
\_EVENT2)

Definition at line 227 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.16 CFE\_PLATFORM\_SB\_FILTERED\_EVENT3** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT3 CFE\_PLATFORM\_SB\_CFGVAL(  
\_EVENT3)

Definition at line 233 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.17 CFE\_PLATFORM\_SB\_FILTERED\_EVENT4** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT4 CFE\_PLATFORM\_SB\_CFGVAL(  
\_EVENT4)

Definition at line 239 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.18 CFE\_PLATFORM\_SB\_FILTERED\_EVENT5** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENTS5 CFE\_PLATFORM\_SB\_CFGVAL(EVENT5)  
Definition at line 245 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.19 CFE\_PLATFORM\_SB\_FILTERED\_EVENT6** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT6 CFE\_PLATFORM\_SB\_CFGVAL(EVENT6)  
Definition at line 251 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.20 CFE\_PLATFORM\_SB\_FILTERED\_EVENT7** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT7 CFE\_PLATFORM\_SB\_CFGVAL(EVENT7)  
Definition at line 257 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.21 CFE\_PLATFORM\_SB\_FILTERED\_EVENT8** #define CFE\_PLATFORM\_SB\_FILTERED\_EVENT8 CFE\_PLATFORM\_SB\_CFGVAL(EVENT8)  
Definition at line 263 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.22 CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID** #define CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID CFE\_PLATFORM\_SB\_CFGVAL(HIGHEST\_VALID\_MSGID)

**Purpose** Highest Valid Message Id

**Description:**

The value of this constant dictates the range of valid message ID's, from 0 to CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID (inclusive).

Although this can be defined differently across platforms, each platform can only publish/subscribe to message ids within their allowable range. Typically this value is set the same across all mission platforms to avoid this complexity.

**Limits**

This parameter has a lower limit is 1, and an upper limit of 0xFFFFFFFF.

When using the direct message map implementation for software bus routing, this value is used to size the map where a value of 0xFFFF results in a 16 KBytes map and 0xFFFF is 128 KBytes.

When using the hash implementation for software bus routing, a multiple of the CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS is used to size the message map. In that case the range selected here does not impact message map memory use, so it's reasonable to use up to the full range supported by the message ID implementation.

Definition at line 155 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.23 CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE** #define CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE CFE\_PLATFORM\_SB\_CFGVAL(M\_BLOCK\_SIZE)

Definition at line 327 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.24 CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT** #define CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT CFE\_PLATFORM\_SB\_CFGVAL(M\_DEST\_PER\_PKT)

**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 90 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.25 CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS** #define CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS CFE\_PLATFORM\_SB\_CFGVAL(MAX←\_MSG\_IDS)

**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 55 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.26 CFE\_PLATFORM\_SB\_MAX\_PIPES** #define CFE\_PLATFORM\_SB\_MAX\_PIPES CFE\_PLATFORM\_SB\_CFGVAL(MAX←\_PIPES)

**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 73 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.27 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_01)

**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 should match CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS

Definition at line 295 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.28 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_02)

Definition at line 297 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.29 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_03)

Definition at line 299 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.30 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_04)

Definition at line 301 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.31 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_05)

Definition at line 303 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.32 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_06)

Definition at line 305 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.33 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_07)

Definition at line 307 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.34 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_08)

Definition at line 309 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.35 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_09)  
Definition at line 311 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.36 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_10)  
Definition at line 313 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.37 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_11)  
Definition at line 315 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.38 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_12)  
Definition at line 317 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.39 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_13)  
Definition at line 319 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.40 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_14)  
Definition at line 321 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.41 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_15)  
Definition at line 323 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.42 CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16** #define CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 CFE\_PLATFORM\_SB\_CFGVAL(MEM\_BLOCK\_SIZE\_16)  
Definition at line 325 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.43 CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS** #define CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS CFE\_PLATFORM\_SB\_CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS

**Purpose** Number of block sizes in SB memory pool structure

**Description:**

The number of block sizes for the software bus memory pool

**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 SB block size list must correlate with this value

Definition at line 282 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.44 CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY CFE\_PLATFORM\_S  
\_TASK\_PRIORITY)

**Purpose** Define SB Task Priority

**Description:**

Defines the cFE\_SB Task priority.

**Limits**

Not Applicable

Definition at line 339 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.45 CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_SB\_START\_TASK\_←  
STACK\_SIZE CFE\_PLATFORM\_SB\_CFGVAL(START\_TASK\_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 355 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.46 DEFAULT\_CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
BUF\_MEMORY\_BYTES 524288

Definition at line 130 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.47 DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME** #define DEFAULT\_CFE\_PLATFORM\_←  
SB\_DEFAULT\_MAP\_FILENAME "/ram/cfe\_sb\_msgmap.dat"

Definition at line 204 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.48 DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT** #define DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT 4  
Definition at line 107 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.49 DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME** #define DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME "/ram/cfe\_sb\_pipe.dat"  
Definition at line 186 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.50 DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME** #define DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME "/ram/cfe\_sb\_route.dat"  
Definition at line 171 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.51 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK1** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK1 CFE\_EVS\_FIRST\_4\_STOP  
Definition at line 225 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.52 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK2** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK2 CFE\_EVS\_FIRST\_4\_STOP  
Definition at line 231 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.53 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK3** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK3 CFE\_EVS\_FIRST\_16\_STOP  
Definition at line 237 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.54 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK4** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK4 CFE\_EVS\_FIRST\_16\_STOP  
Definition at line 243 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.55 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK5** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK5 CFE\_EVS\_NO\_FILTER  
Definition at line 249 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.56 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK6** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK6 CFE\_EVS\_NO\_FILTER  
Definition at line 255 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.57 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK7** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK7 CFE\_EVS\_NO\_FILTER  
Definition at line 261 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.58 DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK8** #define DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_←  
MASK8 CFE\_EVS\_NO\_FILTER

Definition at line 267 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.59 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT1** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT1 CFE\_SB\_SEND\_NO\_SUBS\_EID

Definition at line 222 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.60 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT2** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT2 CFE\_SB\_DUP\_SUBSCRIP\_EID

Definition at line 228 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.61 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT3** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT3 CFE\_SB\_MSGID\_LIM\_ERR\_EID

Definition at line 234 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.62 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT4** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT4 CFE\_SB\_Q\_FULL\_ERR\_EID

Definition at line 240 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.63 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT5** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT5 0

Definition at line 246 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.64 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT6** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT6 0

Definition at line 252 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.65 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT7** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT7 0

Definition at line 258 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.66 DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT8** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
FILTERED\_EVENT8 0

Definition at line 264 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.67 DEFAULT\_CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID** #define DEFAULT\_CFE\_PLATFORM\_SB\_←  
\_HIGHEST\_VALID\_MSGID 0x1FFF

Definition at line 156 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.68 DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE (CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE + 128)  
Definition at line 328 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.69 DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT** #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT 16  
Definition at line 91 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.70 DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS** #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS 256  
Definition at line 56 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.71 DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_PIPES** #define DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_PIPES 64  
Definition at line 74 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.72 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01** #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01 8  
Definition at line 296 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.73 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02** #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 16  
Definition at line 298 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.74 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03** #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 20  
Definition at line 300 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.75 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04** #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 36  
Definition at line 302 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.76 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05** #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 64  
Definition at line 304 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.77 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06** #define DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 96  
Definition at line 306 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.78 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_07 128  
Definition at line 308 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.79 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_08 160  
Definition at line 310 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.80 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_09 256  
Definition at line 312 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.81 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_10 512  
Definition at line 314 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.82 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_11 1024  
Definition at line 316 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.83 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_12 2048  
Definition at line 318 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.84 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_13 4096  
Definition at line 320 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.85 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_14 8192  
Definition at line 322 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.86 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_15 16384  
Definition at line 324 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.87 DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16** #define DEFAULT\_CFE\_PLATFORM\_SB\_↪  
MEM\_BLOCK\_SIZE\_16 32768  
Definition at line 326 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.88 DEFAULT\_CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS** #define DEFAULT\_CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS 17  
Definition at line 283 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.89 DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY 64  
Definition at line 340 of file cfe\_sb\_internal\_cfg.h.

**12.178.2.90 DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE  
Definition at line 356 of file cfe\_sb\_internal\_cfg.h.

## 12.179 cfe/modules/sb/fsw/inc/cfe\_sb\_topicids.h File Reference

```
#include "cfe_sb_topicid_values.h"
```

### Macros

- #define CFE\_MISSION\_SB\_CMD\_TOPICID CFE\_MISSION\_SB\_TIDVAL(CMD)
- #define DEFAULT\_CFE\_MISSION\_SB\_CMD\_TOPICID 3
- #define CFE\_MISSION\_SB\_SEND\_HK\_TOPICID CFE\_MISSION\_SB\_TIDVAL(SEND\_HK)
- #define DEFAULT\_CFE\_MISSION\_SB\_SEND\_HK\_TOPICID 11
- #define CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID CFE\_MISSION\_SB\_TIDVAL(SUB\_RPT\_CTRL)
- #define DEFAULT\_CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID 14
- #define CFE\_MISSION\_SB\_HK\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(HK\_TLM)
- #define DEFAULT\_CFE\_MISSION\_SB\_HK\_TLM\_TOPICID 3
- #define CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(STATS\_TLM)
- #define DEFAULT\_CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID 10
- #define CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(ALLSUBS\_TLM)
- #define DEFAULT\_CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID 13
- #define CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(ONESUB\_TLM)
- #define DEFAULT\_CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID 14

### 12.179.1 Detailed Description

CFE Software Bus (CFE\_SB) Application Topic IDs

### 12.179.2 Macro Definition Documentation

**12.179.2.1 CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID** #define CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(\_TLM)  
Definition at line 57 of file cfe\_sb\_topicids.h.

**12.179.2.2 CFE\_MISSION\_SB\_CMD\_TOPICID** #define CFE\_MISSION\_SB\_CMD\_TOPICID CFE\_MISSION\_SB\_TIDVAL(CMD)

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE command messages

**Limits**

Not Applicable

Definition at line 37 of file cfe\_sb\_topicids.h.

**12.179.2.3 CFE\_MISSION\_SB\_HK\_TLM\_TOPICID** #define CFE\_MISSION\_SB\_HK\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(HK↔\_TLM)

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE telemetry messages

**Limits**

Not Applicable

Definition at line 53 of file cfe\_sb\_topicids.h.

**12.179.2.4 CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID** #define CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(\_TLM)

Definition at line 59 of file cfe\_sb\_topicids.h.

**12.179.2.5 CFE\_MISSION\_SB\_SEND\_HK\_TOPICID** #define CFE\_MISSION\_SB\_SEND\_HK\_TOPICID CFE\_MISSION\_SB\_TIDVAL(SEND↔\_HK)

Definition at line 39 of file cfe\_sb\_topicids.h.

**12.179.2.6 CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID** #define CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID CFE\_MISSION\_SB\_TIDVAL(S↔\_TLM)

Definition at line 55 of file cfe\_sb\_topicids.h.

**12.179.2.7 CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID** #define CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID CFE\_MISSION\_SB\_TIDVAL(\_RPT\_CTRL)

Definition at line 41 of file cfe\_sb\_topicids.h.

**12.179.2.8 DEFAULT\_CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID 13

Definition at line 58 of file cfe\_sb\_topicids.h.

**12.179.2.9 DEFAULT\_CFE\_MISSION\_SB\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_SB\_CMD\_TOPICID 3  
 Definition at line 38 of file cfe\_sb\_topicids.h.

**12.179.2.10 DEFAULT\_CFE\_MISSION\_SB\_HK\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_SB\_HK\_TLM\_TOPICID 3  
 Definition at line 54 of file cfe\_sb\_topicids.h.

**12.179.2.11 DEFAULT\_CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID 14  
 Definition at line 60 of file cfe\_sb\_topicids.h.

**12.179.2.12 DEFAULT\_CFE\_MISSION\_SB\_SEND\_HK\_TOPICID** #define DEFAULT\_CFE\_MISSION\_SB\_SEND\_HK\_TOPICID 11  
 Definition at line 40 of file cfe\_sb\_topicids.h.

**12.179.2.13 DEFAULT\_CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID 10  
 Definition at line 56 of file cfe\_sb\_topicids.h.

**12.179.2.14 DEFAULT\_CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID** #define DEFAULT\_CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID 14  
 Definition at line 42 of file cfe\_sb\_topicids.h.

## 12.180 cfe/modules/tbl/config/default\_cfe\_tbl\_extern\_typedefs.h File Reference

```
#include "common_types.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_fs_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.*
- struct [CFE\\_TBL\\_CombinedFileHdr](#)  
*Complete header for CFE table 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.*
- typedef struct [CFE\\_TBL\\_CombinedFileHdr](#) [CFE\\_TBL\\_CombinedFileHdr\\_t](#)  
*Complete header for CFE table files.*
- typedef CFE\_RESOURCEID\_BASE\_TYPE [CFE\\_TBL\\_RegId\\_t](#)

A type for Table Registry IDs.

- `typedef CFE_RESOURCEID_BASE_TYPE CFE_TBL_HandleId_t`

A type for Table Handle IDs.

## Enumerations

- enum `CFE_TBL_BufferSelect { CFE_TBL_BufferSelect_INACTIVE = 0 , CFE_TBL_BufferSelect_ACTIVE = 1 }`

*Label definitions associated with CFE\_TBL\_BufferSelect\_Enum\_t.*

### 12.180.1 Detailed Description

Declarations and prototypes for cfe\_tbl\_extern\_typedefs module

### 12.180.2 Typedef Documentation

#### 12.180.2.1 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 54 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 12.180.2.2 CFE\_TBL\_CombinedFileHdr\_t `typedef struct CFE_TBL_CombinedFileHdr CFE_TBL_CombinedFileHdr_t`

Complete header for CFE table files.

Table files always have a combination of the standard file header and the table-specific file header. This struct just combines the two and makes for an easier item to pass around, simplifying APIs

#### 12.180.2.3 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.

#### 12.180.2.4 CFE\_TBL\_HandleId\_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_TBL_HandleId_t`

A type for Table Handle IDs.

This is the type that is used for any API accepting or returning a table ID

Definition at line 99 of file default\_cfe\_tbl\_extern\_typedefs.h.

#### 12.180.2.5 CFE\_TBL\_RegId\_t `typedef CFE_RESOURCEID_BASE_TYPE CFE_TBL_RegId_t`

A type for Table Registry IDs.

This is the type that is used for any API accepting or returning an Registry ID

Definition at line 92 of file default\_cfe\_tbl\_extern\_typedefs.h.

### 12.180.3 Enumeration Type Documentation

#### 12.180.3.1 CFE\_TBL\_BufferSelect enum CFE\_TBL\_BufferSelect

Label definitions associated with CFE\_TBL\_BufferSelect\_Enum\_t.

Enumerator

CFE_TBL_BufferSelect_INACTIVE	Select the Inactive buffer for validate or dump.
CFE_TBL_BufferSelect_ACTIVE	Select the Active buffer for validate or dump.

Definition at line 36 of file default\_cfe\_tbl\_extern\_typedefs.h.

## 12.181 cfe/modules/tbl/config/default\_cfe\_tbl\_fcncode\_values.h File Reference

### Macros

- #define CFE\_TBL\_CCVAL(x) CFE\_TBL\_FunctionCode\_##x

### Enumerations

- enum CFE\_TBL\_FunctionCode\_ {
 CFE\_TBL\_FunctionCode\_NOOP = 0 , CFE\_TBL\_FunctionCode\_RESET\_COUNTERS = 1 , CFE\_TBL\_FunctionCode\_LOAD = 2 , CFE\_TBL\_FunctionCode\_DUMP = 3 ,
 CFE\_TBL\_FunctionCode\_VALIDATE = 4 , CFE\_TBL\_FunctionCode\_ACTIVATE = 5 , CFE\_TBL\_FunctionCode\_DUMP\_REGISTRY = 6 , CFE\_TBL\_FunctionCode\_SEND\_REGISTRY = 7 ,
 CFE\_TBL\_FunctionCode\_DELETE\_CDS = 8 , CFE\_TBL\_FunctionCode\_ABORT\_LOAD = 9 }

### 12.181.1 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.

### 12.181.2 Macro Definition Documentation

#### 12.181.2.1 CFE\_TBL\_CCVAL #define CFE\_TBL\_CCVAL(

x ) CFE\_TBL\_FunctionCode\_##x

Definition at line 35 of file default\_cfe\_tbl\_fcncode\_values.h.

### 12.181.3 Enumeration Type Documentation

#### 12.181.3.1 CFE\_TBL\_FunctionCode\_ enum CFE\_TBL\_FunctionCode\_

Enumerator

CFE_TBL_FunctionCode_NOOP	
---------------------------	--

**Enumerator**

CFE_TBL_FunctionCode_RESET_COUNTERS	
CFE_TBL_FunctionCode_LOAD	
CFE_TBL_FunctionCode_DUMP	
CFE_TBL_FunctionCode_VALIDATE	
CFE_TBL_FunctionCode_ACTIVATE	
CFE_TBL_FunctionCode_DUMP_REGISTRY	
CFE_TBL_FunctionCode_SEND_REGISTRY	
CFE_TBL_FunctionCode_DELETE_CDS	
CFE_TBL_FunctionCode_ABORT_LOAD	

Definition at line 37 of file default\_cfe\_tbl\_fcncode\_values.h.

**12.182 cfe/modules/tbl/config/default\_cfe\_tbl\_interface\_cfg\_values.h File Reference****Macros**

- #define [CFE\\_MISSION\\_TBL\\_CFGVAL](#)(x) DEFAULT\_CFE\_MISSION\_TBL\_##x

**12.182.1 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.

**12.182.2 Macro Definition Documentation****12.182.2.1 CFE\_MISSION\_TBL\_CFGVAL #define CFE\_MISSION\_TBL\_CFGVAL( x ) DEFAULT\_CFE\_MISSION\_TBL\_##x**

Definition at line 36 of file default\_cfe\_tbl\_interface\_cfg\_values.h.

**12.183 cfe/modules/tbl/config/default\_cfe\_tbl\_internal\_cfg\_values.h File Reference****Macros**

- #define [CFE\\_PLATFORM\\_TBL\\_CFGVAL](#)(x) DEFAULT\_CFE\_PLATFORM\_TBL\_##x

**12.183.1 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.

### 12.183.2 Macro Definition Documentation

**12.183.2.1 CFE\_PLATFORM\_TBL\_CFGVAL** #define CFE\_PLATFORM\_TBL\_CFGVAL( x ) DEFAULT\_CFE\_PLATFORM\_TBL\_##x

Definition at line 36 of file default\_cfe\_tbl\_internal\_cfg\_values.h.

## 12.184 cfe/modules/tbl/config/default\_cfe\_tbl\_mission\_cfg.h File Reference

```
#include "cfe_tbl_interface_cfg_values.h"
```

### Macros

- #define CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH CFE\_MISSION\_TBL\_CFGVAL(MAX\_NAME\_LENGTH)
- #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH 16
- #define CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN CFE\_MISSION\_TBL\_CFGVAL(MAX\_FULL\_NAME\_LEN)
- #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN 40

### 12.184.1 Detailed Description

CFE Event Services (CFE\_TBL) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.184.2 Macro Definition Documentation

**12.184.2.1 CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN** #define CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN CFE\_MISSION\_TBL\_CFG(\_FULL\_NAME\_LEN)

**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 71 of file default\_cfe\_tbl\_mission\_cfg.h.

**12.184.2.2 CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH** #define CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH CFE\_MISSION\_TBL\_CFGVAL  
\_NAME\_LENGTH)

**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 50 of file default\_cfe\_tbl\_mission\_cfg.h.

**12.184.2.3 DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN** #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN 40

Definition at line 74 of file default\_cfe\_tbl\_mission\_cfg.h.

**12.184.2.4 DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH** #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH 16

Definition at line 51 of file default\_cfe\_tbl\_mission\_cfg.h.

## 12.185 cfe/modules/tbl/config/default\_cfe\_tbl\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_tbl_fcncodes.h"
#include "cfe_tbl_msgdefs.h"
#include "cfe_tbl_msgstruct.h"
```

### 12.185.1 Detailed Description

Specification for the CFE Event Services (CFE\_TBL) command and telemetry message data types.

This is a compatibility header for the "cfe\_tbl\_msg.h" file that has traditionally provided the message definitions for cFS apps.

**Note**

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

## 12.186 cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_es_extern_typedefs.h"
#include "cfe_time_extern_typedefs.h"
#include "cfe_tbl_extern_typedefs.h"
#include "cfe_tbl_fcncodes.h"
```

## Data Structures

- struct [CFE\\_TBL\\_LoadCmd\\_Payload](#)  
*Load Table Command Payload.*
- struct [CFE\\_TBL\\_DumpCmd\\_Payload](#)  
*Dump Table Command Payload.*
- struct [CFE\\_TBL\\_ValidateCmd\\_Payload](#)  
*Validate Table Command Payload.*
- struct [CFE\\_TBL\\_ActivateCmd\\_Payload](#)  
*Activate Table Command Payload.*
- struct [CFE\\_TBL\\_DumpRegistryCmd\\_Payload](#)  
*Dump Registry Command Payload.*
- struct [CFE\\_TBL\\_SendRegistryCmd\\_Payload](#)  
*Send Table Registry Command Payload.*
- struct [CFE\\_TBL\\_DelCDSCmd\\_Payload](#)  
*Delete Critical Table CDS Command Payload.*
- struct [CFE\\_TBL\\_AbortLoadCmd\\_Payload](#)  
*Abort Load Command Payload.*
- struct [CFE\\_TBL\\_NotifyCmd\\_Payload](#)  
*Table Management Notification Command Payload.*
- struct [CFE\\_TBL\\_HousekeepingTlm\\_Payload](#)
- struct [CFE\\_TBL\\_TblRegPacket\\_Payload](#)

## Typedefs

- typedef struct [CFE\\_TBL\\_LoadCmd\\_Payload](#) [CFE\\_TBL\\_LoadCmd\\_Payload\\_t](#)  
*Load Table Command Payload.*
- typedef struct [CFE\\_TBL\\_DumpCmd\\_Payload](#) [CFE\\_TBL\\_DumpCmd\\_Payload\\_t](#)  
*Dump Table Command Payload.*
- typedef struct [CFE\\_TBL\\_ValidateCmd\\_Payload](#) [CFE\\_TBL\\_ValidateCmd\\_Payload\\_t](#)  
*Validate Table Command Payload.*
- typedef struct [CFE\\_TBL\\_ActivateCmd\\_Payload](#) [CFE\\_TBL\\_ActivateCmd\\_Payload\\_t](#)  
*Activate Table Command Payload.*
- typedef struct [CFE\\_TBL\\_DumpRegistryCmd\\_Payload](#) [CFE\\_TBL\\_DumpRegistryCmd\\_Payload\\_t](#)  
*Dump Registry Command Payload.*
- typedef struct [CFE\\_TBL\\_SendRegistryCmd\\_Payload](#) [CFE\\_TBL\\_SendRegistryCmd\\_Payload\\_t](#)  
*Send Table Registry Command Payload.*
- typedef struct [CFE\\_TBL\\_DelCDSCmd\\_Payload](#) [CFE\\_TBL\\_DelCDSCmd\\_Payload\\_t](#)  
*Delete Critical Table CDS Command Payload.*
- typedef struct [CFE\\_TBL\\_AbortLoadCmd\\_Payload](#) [CFE\\_TBL\\_AbortLoadCmd\\_Payload\\_t](#)  
*Abort Load Command Payload.*
- typedef struct [CFE\\_TBL\\_NotifyCmd\\_Payload](#) [CFE\\_TBL\\_NotifyCmd\\_Payload\\_t](#)  
*Table Management Notification Command Payload.*
- typedef struct [CFE\\_TBL\\_HousekeepingTlm\\_Payload](#) [CFE\\_TBL\\_HousekeepingTlm\\_Payload\\_t](#)
- typedef struct [CFE\\_TBL\\_TblRegPacket\\_Payload](#) [CFE\\_TBL\\_TblRegPacket\\_Payload\\_t](#)

### 12.186.1 Detailed Description

Specification for the CFE Event Services (CFE\_TBL) command and telemetry message payload and constant definitions.

## 12.186.2 Typedef Documentation

**12.186.2.1 CFE\_TBL\_AbortLoadCmd\_Payload\_t** `typedef struct CFE_TBL_AbortLoadCmd_Payload CFE_TBL_AbortLoadCmd_Payload`  
Abort Load Command Payload.

For command details, see [CFE\\_TBL\\_ABORT\\_LOAD\\_CC](#)

**12.186.2.2 CFE\_TBL\_ActivateCmd\_Payload\_t** `typedef struct CFE_TBL_ActivateCmd_Payload CFE_TBL_ActivateCmd_Payload`  
Activate Table Command Payload.

For command details, see [CFE\\_TBL\\_ACTIVATE\\_CC](#)

**12.186.2.3 CFE\_TBL\_DelCDSCmd\_Payload\_t** `typedef struct CFE_TBL_DelCDSCmd_Payload CFE_TBL_DelCDSCmd_Payload`  
Delete Critical Table CDS Command Payload.

For command details, see [CFE\\_TBL\\_DELETE\\_CDS\\_CC](#)

**12.186.2.4 CFE\_TBL\_DumpCmd\_Payload\_t** `typedef struct CFE_TBL_DumpCmd_Payload CFE_TBL_DumpCmd_Payload`  
Dump Table Command Payload.

For command details, see [CFE\\_TBL\\_DUMP\\_CC](#)

**12.186.2.5 CFE\_TBL\_DumpRegistryCmd\_Payload\_t** `typedef struct CFE_TBL_DumpRegistryCmd_Payload CFE_TBL_DumpRegistryCmd_Payload`  
`CFE_TBL_DumpRegistryCmd_Payload_t`

Dump Registry Command Payload.

For command details, see [CFE\\_TBL\\_DUMP\\_REGISTRY\\_CC](#)

**12.186.2.6 CFE\_TBL\_HousekeepingTlm\_Payload\_t** `typedef struct CFE_TBL_HousekeepingTlm_Payload CFE_TBL_HousekeepingTlm_Payload`  
`CFE_TBL_HousekeepingTlm_Payload_t`

**Name** Table Services Housekeeping Packet

**12.186.2.7 CFE\_TBL\_LoadCmd\_Payload\_t** `typedef struct CFE_TBL_LoadCmd_Payload CFE_TBL_LoadCmd_Payload`  
Load Table Command Payload.

For command details, see [CFE\\_TBL\\_LOAD\\_CC](#)

**12.186.2.8 CFE\_TBL\_NotifyCmd\_Payload\_t** `typedef struct CFE_TBL_NotifyCmd_Payload CFE_TBL_NotifyCmd_Payload`  
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).

**12.186.2.9 CFE\_TBL\_SendRegistryCmd\_Payload\_t** `typedef struct CFE_TBL_SendRegistryCmd_Payload CFE_TBL_SendRegistryCmd_Payload`  
`CFE_TBL_SendRegistryCmd_Payload_t`

Send Table Registry Command Payload.

For command details, see [CFE\\_TBL\\_SEND\\_REGISTRY\\_CC](#)

**12.186.2.10 CFE\_TBL\_TblRegPacket\_Payload\_t** `typedef struct CFE_TBL_TblRegPacket_Payload CFE_TBL_TblRegPacket_Pay`

**Name** Table Registry Info Packet

**12.186.2.11 CFE\_TBL\_ValidateCmd\_Payload\_t** `typedef struct CFE_TBL_ValidateCmd_Payload CFE_TBL_ValidateCmd_Payloa`

Validate Table Command Payload.

For command details, see [CFE\\_TBL\\_VALIDATE\\_CC](#)

**12.187 cfe/modules/tbl/config/default\_cfe\_tbl\_msgid\_values.h File Reference**

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_tbl_topicids.h"
```

**Macros**

- `#define CFE_PLATFORM_TBL_CMD_MIDVAL(x) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TBL_##x##_TOPICID)`
- `#define CFE_PLATFORM_TBL_TLM_MIDVAL(x) CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TBL_##x##_TOPICID)`

**12.187.1 Detailed Description**

CFE Event Services (CFE\_TBL) Application Message IDs

**12.187.2 Macro Definition Documentation****12.187.2.1 CFE\_PLATFORM\_TBL\_CMD\_MIDVAL** `#define CFE_PLATFORM_TBL_CMD_MIDVAL(`

`x ) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TBL_##x##_TOPICID)`

Definition at line 29 of file default\_cfe\_tbl\_msgid\_values.h.

**12.187.2.2 CFE\_PLATFORM\_TBL\_TLM\_MIDVAL** `#define CFE_PLATFORM_TBL_TLM_MIDVAL(`

`x ) CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_MISSION_TBL_##x##_TOPICID)`

Definition at line 30 of file default\_cfe\_tbl\_msgid\_values.h.

**12.188 cfe/modules/tbl/config/default\_cfe\_tbl\_msgids.h File Reference**

```
#include "cfe_tbl_msgid_values.h"
```

**Macros**

- `#define CFE_TBL_CMD_MID CFE_PLATFORM_TBL_CMD_MIDVAL(CMD)`
- `#define CFE_TBL_SEND_HK_MID CFE_PLATFORM_TBL_CMD_MIDVAL(SEND_HK)`
- `#define CFE_TBL_HK_TLM_MID CFE_PLATFORM_TBL_TLM_MIDVAL(HK_TLM)`
- `#define CFE_TBL_REG_TLM_MID CFE_PLATFORM_TBL_TLM_MIDVAL(REG_TLM)`

**12.188.1 Detailed Description**

CFE Event Services (CFE\_TBL) Application Message IDs

## 12.188.2 Macro Definition Documentation

**12.188.2.1 CFE\_TBL\_CMD\_MID** #define CFE\_TBL\_CMD\_MID CFE\_PLATFORM\_TBL\_CMD\_MIDVAL(CMD)  
Definition at line 31 of file default\_cfe\_tbl\_msgids.h.

**12.188.2.2 CFE\_TBL\_HK\_TLM\_MID** #define CFE\_TBL\_HK\_TLM\_MID CFE\_PLATFORM\_TBL\_TLM\_MIDVAL(HK\_TLM)  
Definition at line 37 of file default\_cfe\_tbl\_msgids.h.

**12.188.2.3 CFE\_TBL\_REG\_TLM\_MID** #define CFE\_TBL\_REG\_TLM\_MID CFE\_PLATFORM\_TBL\_TLM\_MIDVAL(REG\_↔  
TLM)  
Definition at line 38 of file default\_cfe\_tbl\_msgids.h.

**12.188.2.4 CFE\_TBL\_SEND\_HK\_MID** #define CFE\_TBL\_SEND\_HK\_MID CFE\_PLATFORM\_TBL\_CMD\_MIDVAL(SEND\_↔  
HK)  
Definition at line 32 of file default\_cfe\_tbl\_msgids.h.

## 12.189 cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h File Reference

```
#include "cfe_tbl_msgdefs.h"
#include "cfe_msg_hdr.h"
```

### Data Structures

- struct [CFE\\_TBL\\_NoopCmd](#)
- struct [CFE\\_TBL\\_ResetCountersCmd](#)
- struct [CFE\\_TBL\\_SendHkCmd](#)
- struct [CFE\\_TBL\\_LoadCmd](#)  
*Load Table Command.*
- struct [CFE\\_TBL\\_DumpCmd](#)
- struct [CFE\\_TBL\\_ValidateCmd](#)  
*Validate Table Command.*
- struct [CFE\\_TBL\\_ActivateCmd](#)  
*Activate Table Command.*
- struct [CFE\\_TBL\\_DumpRegistryCmd](#)  
*Dump Registry Command.*
- struct [CFE\\_TBL\\_SendRegistryCmd](#)  
*Send Table Registry Command.*
- struct [CFE\\_TBL\\_DeleteCDSCmd](#)  
*Delete Critical Table CDS Command.*
- struct [CFE\\_TBL\\_AbortLoadCmd](#)  
*Abort Load Command.*
- struct [CFE\\_TBL\\_NotifyCmd](#)
- struct [CFE\\_TBL\\_HousekeepingTlm](#)
- struct [CFE\\_TBL\\_TableRegistryTlm](#)

## Typedefs

- `typedef struct CFE_TBL_NoopCmd CFE_TBL_NoopCmd_t`
- `typedef struct CFE_TBL_ResetCountersCmd CFE_TBL_ResetCountersCmd_t`
- `typedef struct CFE_TBL_SendHkCmd CFE_TBL_SendHkCmd_t`
- `typedef struct CFE_TBL_LoadCmd CFE_TBL_LoadCmd_t`  
*Load Table Command.*
- `typedef struct CFE_TBL_DumpCmd CFE_TBL_DumpCmd_t`
- `typedef struct CFE_TBL_ValidateCmd CFE_TBL_ValidateCmd_t`  
*Validate Table Command.*
- `typedef struct CFE_TBL_ActivateCmd CFE_TBL_ActivateCmd_t`  
*Activate Table Command.*
- `typedef struct CFE_TBL_DumpRegistryCmd CFE_TBL_DumpRegistryCmd_t`  
*Dump Registry Command.*
- `typedef struct CFE_TBL_SendRegistryCmd CFE_TBL_SendRegistryCmd_t`  
*Send Table Registry Command.*
- `typedef struct CFE_TBL_DeleteCDSCmd CFE_TBL_DeleteCDSCmd_t`  
*Delete Critical Table CDS Command.*
- `typedef struct CFE_TBL_AbortLoadCmd CFE_TBL_AbortLoadCmd_t`  
*Abort Load Command.*
- `typedef struct CFE_TBL_NotifyCmd CFE_TBL_NotifyCmd_t`
- `typedef struct CFE_TBL_HousekeepingTlm CFE_TBL_HousekeepingTlm_t`
- `typedef struct CFE_TBL_TableRegistryTlm CFE_TBL_TableRegistryTlm_t`

### 12.189.1 Detailed Description

Purpose: cFE Executive Services (TBL) Command and Telemetry packet definition file.

### 12.189.2 Typedef Documentation

**12.189.2.1 CFE\_TBL\_AbortLoadCmd\_t** `typedef struct CFE_TBL_AbortLoadCmd CFE_TBL_AbortLoadCmd_t`  
Abort Load Command.

**12.189.2.2 CFE\_TBL\_ActivateCmd\_t** `typedef struct CFE_TBL_ActivateCmd CFE_TBL_ActivateCmd_t`  
Activate Table Command.

**12.189.2.3 CFE\_TBL\_DeleteCDSCmd\_t** `typedef struct CFE_TBL_DeleteCDSCmd CFE_TBL_DeleteCDSCmd_t`  
Delete Critical Table CDS Command.

**12.189.2.4 CFE\_TBL\_DumpCmd\_t** `typedef struct CFE_TBL_DumpCmd CFE_TBL_DumpCmd_t`  
/brief Dump Table Command

**12.189.2.5 CFE\_TBL\_DumpRegistryCmd\_t** `typedef struct CFE_TBL_DumpRegistryCmd CFE_TBL_DumpRegistryCmd_t`  
Dump Registry Command.

**12.189.2.6 CFE\_TBL\_HousekeepingTlm\_t** `typedef struct CFE_TBL_HousekeepingTlm CFE_TBL_HousekeepingTlm_t`

**12.189.2.7 CFE\_TBL\_LoadCmd\_t** `typedef struct CFE_TBL_LoadCmd CFE_TBL_LoadCmd_t`  
Load Table Command.

**12.189.2.8 CFE\_TBL\_NoopCmd\_t** `typedef struct CFE_TBL_NoopCmd CFE_TBL_NoopCmd_t`

**12.189.2.9 CFE\_TBL\_NotifyCmd\_t** `typedef struct CFE_TBL_NotifyCmd CFE_TBL_NotifyCmd_t`  
/brief Table Management Notification Command

**12.189.2.10 CFE\_TBL\_ResetCountersCmd\_t** `typedef struct CFE_TBL_ResetCountersCmd CFE_TBL_ResetCountersCmd_t`

**12.189.2.11 CFE\_TBL\_SendHkCmd\_t** `typedef struct CFE_TBL_SendHkCmd CFE_TBL_SendHkCmd_t`

**12.189.2.12 CFE\_TBL\_SendRegistryCmd\_t** `typedef struct CFE_TBL_SendRegistryCmd CFE_TBL_SendRegistryCmd_t`  
Send Table Registry Command.

**12.189.2.13 CFE\_TBL\_TableRegistryTlm\_t** `typedef struct CFE_TBL_TableRegistryTlm CFE_TBL_TableRegistryTlm_t`

**12.189.2.14 CFE\_TBL\_ValidateCmd\_t** `typedef struct CFE_TBL_ValidateCmd CFE_TBL_ValidateCmd_t`  
Validate Table Command.

## 12.190 cfe/modules/tbl/config/default\_cfe\_tbl\_platform\_cfg.h File Reference

```
#include "cfe_tbl_mission_cfg.h"
#include "cfe_tbl_internal_cfg_values.h"
```

### Macros

- `#define CFE_PLATFORM_TBL_START_TASK_PRIORITY CFE_PLATFORM_TBL_CFGVAL(START_TASK_PRIORITY)`
- `#define DEFAULT_CFE_PLATFORM_TBL_START_TASK_PRIORITY 70`
- `#define CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_TBL_CFGVAL(START_TASK_STACK_SIZE)`
- `#define DEFAULT_CFE_PLATFORM_TBL_START_TASK_STACK_SIZE CFE_PLATFORM_ES_DEFAULT_STACK_SIZE`
- `#define CFE_PLATFORM_TBL_BUF_MEMORY_BYTES CFE_PLATFORM_TBL_CFGVAL(BUF_MEMORY_BYTES)`
- `#define DEFAULT_CFE_PLATFORM_TBL_BUF_MEMORY_BYTES 524288`
- `#define CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE CFE_PLATFORM_TBL_CFGVAL(MAX_DBL_TABLE_SIZE)`
- `#define DEFAULT_CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE 16384`
- `#define CFE_PLATFORM_TBL_MAX_SNGL_TABLE_SIZE CFE_PLATFORM_TBL_CFGVAL(MAX_SNGL_TABLE_SIZE)`

- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE 16384
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_TABLES)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128
- #define CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_CRITICAL\_TABLES)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES 32
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_HANDLES)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256
- #define CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_SIMULTANEOUS\_LOADS)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS 4
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_VALIDATIONS)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS 10
- #define CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE CFE\_PLATFORM\_TBL\_CFGVAL(DEFAULT\_REG\_DUMP\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE "/ram/cfe\_tbl\_reg.log"
- #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_COUNT)
- #define DEFAULT\_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 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_1)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 (0x42)
- #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_2)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_COUNT)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT 0
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_1)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 (1)
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_2)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_3)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 0
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_4)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 0

### 12.190.1 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.

## 12.190.2 Macro Definition Documentation

**12.190.2.1 CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES** #define CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES CFE\_PLATFORM\_TBL\_C←\_MEMORY\_BYTES)

**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 83 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.2 CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE** #define CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE CFE\_PLATFORM\_TBL\_C←\_CFGVAL(DEFAULT\_REG\_DUMP\_FILE)

**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 205 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.3 CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES CFE\_PLATFORM\_TBL\_C←\_CFGVAL(MAX\_CRITICAL\_TABLES)

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

**Description:**

Defines the maximum number of critical tables supported by this processor's Table Services.

**Limits**

This number must be less than 32767. It should be recognized that this parameter determines the size of the Critical Table Registry which is maintained in the Critical Data Store. An excessively high number will waste Critical Data Store memory. Therefore, this number must not exceed the value defined in CFE\_PLATFORM\_ES\_C←\_CDS\_MAX\_NUM\_ENTRIES.

Definition at line 142 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.4 CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE** #define CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_DBL\_TABLE\_SIZE)

**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 96 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.5 CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_HANDLES)

**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 156 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.6 CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_TABLES)

**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 127 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.7 CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS CFE\_PLATFORM\_NUM\_VALIDATIONS)

**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 191 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.8 CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS** #define CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_SIMULTANEOUS\_LOADS)

**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 172 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.9 CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE** #define CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_SNGL\_TABLE\_SIZE)

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

**Description:**

Defines the maximum allowed size (in bytes) of a single buffered table. **NOTE:** This size determines the size of all shared table buffers. Therefore, this size will be multiplied by [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) below when allocating memory for shared tables.

**Limits**

The cFE does not place a limit on the size of this parameter but it must be small enough to allow for [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) number of tables to fit into [CFE\\_PLATFORM\\_TBL\\_BUF\\_MEMORY\\_BYTES](#).

Definition at line 113 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.10 CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY CFE\_PLATFORM\_TBL\_CFGVAL(START\_TASK\_PRIORITY)

**Purpose** Define TBL Task Priority

**Description:**

Defines the cFE\_TBL Task priority.

**Limits**

Not Applicable

Definition at line 50 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.11 CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(START\_TASK\_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 66 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.12 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 229 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.13 CFE\_PLATFORM\_TBL\_VALID\_PRID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_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 282 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.14 CFE\_PLATFORM\_TBL\_VALID\_PRID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_2)

Definition at line 285 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.15 CFE\_PLATFORM\_TBL\_VALID\_PRID\_3** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_3)

Definition at line 288 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.16 CFE\_PLATFORM\_TBL\_VALID\_PRID\_4** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_4)

Definition at line 291 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.17 CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_COUNT)

**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 267 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.18 CFE\_PLATFORM\_TBL\_VALID\_SCID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_1)

**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 244 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.19 CFE\_PLATFORM\_TBL\_VALID\_SCID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_2)

Definition at line 247 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.20 CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT

**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 225 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.21 DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES 524288

Definition at line 84 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.22 DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE** #define DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE "/ram/cfe\_tbl\_reg.log"

Definition at line 206 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.23 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES 32

Definition at line 143 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.24 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE 16384

Definition at line 97 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.25 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256

Definition at line 157 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.26 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128

Definition at line 128 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.27 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS 10

Definition at line 192 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.28 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS 4  
Definition at line 173 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.29 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE 16384  
Definition at line 114 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.30 DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY 70  
Definition at line 51 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.31 DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE  
Definition at line 67 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.32 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 (1)  
Definition at line 283 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.33 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))  
Definition at line 286 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.34 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 0  
Definition at line 289 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.35 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 0  
Definition at line 292 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.36 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT 0  
Definition at line 268 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.37 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 (0x42)  
Definition at line 245 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.38 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_2** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_←  
SCID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))  
Definition at line 248 of file default\_cfe\_tbl\_platform\_cfg.h.

**12.190.2.39 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT** #define DEFAULT\_CFE\_PLATFORM\_TBL\_←  
VALID\_SCID\_COUNT 0  
Definition at line 226 of file default\_cfe\_tbl\_platform\_cfg.h.

## 12.191 cfe/modules/tbl/config/default\_cfe\_tbl\_topicid\_values.h File Reference

### Macros

- #define CFE\_MISSION\_TBL\_TIDVAL(x) DEFAULT\_CFE\_MISSION\_TBL\_##x##\_TOPICID

#### 12.191.1 Detailed Description

CFE Table Services (CFE\_TBL) Application Topic IDs

#### 12.191.2 Macro Definition Documentation

**12.191.2.1 CFE\_MISSION\_TBL\_TIDVAL** #define CFE\_MISSION\_TBL\_TIDVAL(  
x ) DEFAULT\_CFE\_MISSION\_TBL\_##x##\_TOPICID

Definition at line 26 of file default\_cfe\_tbl\_topicid\_values.h.

## 12.192 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_CODEC_ERROR_ERR_EID` 61  
*TBL Encoding/Decoding error Event ID.*
- #define `CFE_TBL_CREATING_DUMP_FILE_ERR_EID` 62  
*TBL Write File Creation Failure Event ID.*
- #define `CFE_TBL_WRITE_CFE_HDR_ERR_EID` 63  
*TBL Write Standard File Header Failure Event ID.*
- #define `CFE_TBL_WRITE_TBL_HDR_ERR_EID` 64  
*TBL Write Table File Header Failure Event ID.*
- #define `CFE_TBL_WRITE_TBL_IMG_ERR_EID` 65  
*TBL Write Table File Data Failure Event ID.*
- #define `CFE_TBL_NO_INACTIVE_BUFFER_ERR_EID` 66  
*TBL Validate Or Write Table Command No Inactive Buffer Event ID.*
- #define `CFE_TBL_TOO_MANY_VALIDATIONS_ERR_EID` 67  
*TBL Validate Table Command Result Storage Exceeded Event ID.*
- #define `CFE_TBL_WRITE_TBL_REG_ERR_EID` 68  
*TBL Write Table Registry File Data Failure Event ID.*
- #define `CFE_TBL_LOAD_ABORT_ERR_EID` 69  
*TBL Abort Table Load No Load Started Event ID.*
- #define `CFE_TBL_ACTIVATE_ERR_EID` 70  
*TBL Activate Table Command No Inactive Buffer Event ID.*
- #define `CFE_TBL_FILE_INCOMPLETE_ERR_EID` 71

- `#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_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.*

### 12.192.1 Detailed Description

cFE Table Services Event IDs

### 12.192.2 Macro Definition Documentation

**12.192.2.1 CFE\_TBL\_ACTIVATE\_DUMP\_ONLY\_ERR\_EID** #define CFE\_TBL\_ACTIVATE\_DUMP\_ONLY\_ERR\_EID 78  
TBL Activate Table Command For Dump Only Table Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to table being dump only.  
Definition at line 556 of file cfe\_tbl\_eventids.h.

**12.192.2.2 CFE\_TBL\_ACTIVATE\_ERR\_EID** #define CFE\_TBL\_ACTIVATE\_ERR\_EID 70  
TBL Activate Table Command No Inactive Buffer Event ID.

Type: ERROR

Cause:

[TBL Activate Table Command](#) failure due to no associated inactive buffer.  
Definition at line 462 of file cfe\_tbl\_eventids.h.

**12.192.2.3 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.

**12.192.2.4 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.

**12.192.2.5 CFE\_TBL\_CDS\_DELETE\_ERR\_EID** #define CFE\_TBL\_CDS\_DELETE\_ERR\_EID 86  
TBL Delete Table CDS Command Internal Error Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to an internal error. See the system log for more information.  
Definition at line 652 of file cfe\_tbl\_eventids.h.

**12.192.2.6 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.

**12.192.2.7 CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID** #define CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID 85  
TBL Delete Table CDS Command Not In CDS Registry Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table name not found in the CDS registry.  
Definition at line 640 of file cfe\_tbl\_eventids.h.

**12.192.2.8 CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID** #define CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID 87  
TBL Delete Table CDS Command App Active Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the owning application being active.  
Definition at line 664 of file cfe\_tbl\_eventids.h.

**12.192.2.9 CFE\_TBL\_CODEC\_ERROR\_ERR\_EID** #define CFE\_TBL\_CODEC\_ERROR\_ERR\_EID 61  
TBL Encoding/Decoding error Event ID.

Type: ERROR

Cause: Encoding/Decoding of the table binary data from the file content failed

This is possibly due to a corrupt or incompatible file.  
Definition at line 358 of file cfe\_tbl\_eventids.h.

**12.192.2.10 CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID** #define CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID 62  
TBL Write File Creation Failure Event ID.

Type: ERROR

Cause:

TBL Write Table or Table Registry File failed to create file. OVERLOADED  
Definition at line 369 of file cfe\_tbl\_eventids.h.

**12.192.2.11 CFE\_TBL\_DUMP\_PENDING\_ERR\_EID** #define CFE\_TBL\_DUMP\_PENDING\_ERR\_EID 77  
TBL Write Table Command Already In Progress Event ID.

Type: ERROR

Cause:

[TBL Write Table Command](#) failure due to a dump already in progress for the same table.  
Definition at line 544 of file cfe\_tbl\_eventids.h.

**12.192.2.12 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.

**12.192.2.13 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 675 of file cfe\_tbl\_eventids.h.

**12.192.2.14 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.

**12.192.2.15 CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID** #define CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID 71  
TBL Load Table Incomplete Load Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to inability to read the size of data specified in the table header from file. OVERLOADED  
Definition at line 474 of file cfe\_tbl\_eventids.h.

**12.192.2.16 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.

**12.192.2.17 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.

**12.192.2.18 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.

**12.192.2.19 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.

**12.192.2.20 CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID** #define CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID 75  
TBL Load Table File Excess Data Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified size of data being smaller than the actual data contained in the file. OVERLOADED

Definition at line 520 of file cfe\_tbl\_eventids.h.

**12.192.2.21 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.

**12.192.2.22 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 818 of file cfe\_tbl\_eventids.h.

**12.192.2.23 CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID** #define CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID 80  
TBL Validate Or Write Table Command Invalid Buffer Event ID.

Type: ERROR

Cause:

[TBL Validate Table Command](#) or [TBL Write Table Command](#) failure due to an invalid buffer selection. OVERLOADED  
Definition at line 580 of file cfe\_tbl\_eventids.h.

**12.192.2.24 CFE\_TBL\_IN\_REGISTRY\_ERR\_EID** #define CFE\_TBL\_IN\_REGISTRY\_ERR\_EID 82  
TBL Delete Table CDS Command For Registered Table Event ID.

Type: ERROR

Cause:

[TBL Delete Table CDS Command](#) failure due to the table being currently registered.  
Definition at line 604 of file cfe\_tbl\_eventids.h.

**12.192.2.25 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.

**12.192.2.26 CFE\_TBL\_LEN\_ERR\_EID** #define CFE\_TBL\_LEN\_ERR\_EID 52  
TBL Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the message ID and command code received on the TBL message pipe.  
Definition at line 257 of file cfe\_tbl\_eventids.h.

**12.192.2.27 CFE\_TBL\_LOAD\_ABORT\_ERR\_EID** #define CFE\_TBL\_LOAD\_ABORT\_ERR\_EID 69  
TBL Abort Table Load No Load Started Event ID.

Type: ERROR

Cause:

[TBL Abort Table Load Command](#) failure due to no load in progress.  
Definition at line 450 of file cfe\_tbl\_eventids.h.

**12.192.2.28 CFE\_TBL\_LOAD\_ABORT\_INF\_EID** #define CFE\_TBL\_LOAD\_ABORT\_INF\_EID 21  
TBL Abort Table Load Success Event ID.

Type: INFORMATION

Cause:

[TBL Abort Table Load Command](#) success.

Definition at line 157 of file cfe\_tbl\_eventids.h.

**12.192.2.29 CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID** #define CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID 72  
TBL Load Table File Exceeds Table Size Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to the file header specified offset and/or size of data exceeding the table size. OVERLOADED  
Definition at line 486 of file cfe\_tbl\_eventids.h.

**12.192.2.30 CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID** #define CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID 101  
TBL Load Table Filename Too Long Event ID.

Type: ERROR

Cause:

Load table filename too long.

Definition at line 796 of file cfe\_tbl\_eventids.h.

**12.192.2.31 CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID** #define CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID 100  
TBL Load Table API Load Already In Progress Event ID.

Type: ERROR

Cause:

[CFE\\_TBL\\_Load](#) API failure due to load already in progress.

Definition at line 785 of file cfe\_tbl\_eventids.h.

**12.192.2.32 CFE\_TBL\_LOAD\_PEND\_REQ\_INF\_EID** #define CFE\_TBL\_LOAD\_PEND\_REQ\_INF\_EID 17  
TBL Load Table Pending Notification Success Event ID.

Type: DEBUG

Cause:

TBL load table pending notification successfully sent.  
Definition at line 134 of file cfe\_tbl\_eventids.h.

**12.192.2.33 CFE\_TBL\_LOAD\_SUCCESS\_INF\_EID** #define CFE\_TBL\_LOAD\_SUCCESS\_INF\_EID 35  
TBL Load Table API Success Event ID.

Type: DEBUG (the first time) and INFORMATION (normally)

Cause:

[CFE\\_TBL\\_Load](#) API success for dump only or normal table. OVERLOADED  
Definition at line 191 of file cfe\_tbl\_eventids.h.

**12.192.2.34 CFE\_TBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID** #define CFE\_TBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID 102  
TBL Load Table Name Mismatch Event ID.

Type: ERROR

Cause:

Load table name in the table file header does not match the specified table name.  
Definition at line 807 of file cfe\_tbl\_eventids.h.

**12.192.2.35 CFE\_TBL\_LOAD\_TYPE\_ERR\_EID** #define CFE\_TBL\_LOAD\_TYPE\_ERR\_EID 94  
TBL Load Table API Invalid Source Type Event ID.

Type: ERROR

Cause:

[CFE\\_TBL\\_Load](#) API valid due to invalid source type.  
Definition at line 730 of file cfe\_tbl\_eventids.h.

**12.192.2.36 CFE\_TBL\_LOAD\_VAL\_ERR\_EID** #define CFE\_TBL\_LOAD\_VAL\_ERR\_EID 93  
TBL Validation Function Invalid Return Code Event ID.

Type: ERROR

Cause:

Invalid table validation function return code.  
Definition at line 719 of file cfe\_tbl\_eventids.h.

**12.192.2.37 CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID** #define CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID 79  
TBL Load Table For Dump Only Table Event ID.

Type: ERROR

Cause:

TBL Load Table failure due to table being dump only. OVERLOADED  
Definition at line 567 of file cfe\_tbl\_eventids.h.

**12.192.2.38 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.

**12.192.2.39 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 415 of file cfe\_tbl\_eventids.h.

**12.192.2.40 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.

**12.192.2.41 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.

**12.192.2.42 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.

**12.192.2.43 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 616 of file cfe\_tbl\_eventids.h.

**12.192.2.44 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 628 of file cfe\_tbl\_eventids.h.

**12.192.2.45 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.

**12.192.2.46 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.

**12.192.2.47 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 508 of file cfe\_tbl\_eventids.h.

**12.192.2.48 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 774 of file cfe\_tbl\_eventids.h.

**12.192.2.49 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 686 of file cfe\_tbl\_eventids.h.

**12.192.2.50 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.

**12.192.2.51 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 697 of file cfe\_tbl\_eventids.h.

**12.192.2.52 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 763 of file cfe\_tbl\_eventids.h.

**12.192.2.53 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.

**12.192.2.54 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 532 of file cfe\_tbl\_eventids.h.

**12.192.2.55 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 427 of file cfe\_tbl\_eventids.h.

**12.192.2.56 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 708 of file cfe\_tbl\_eventids.h.

**12.192.2.57 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 592 of file cfe\_tbl\_eventids.h.

**12.192.2.58 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 741 of file cfe\_tbl\_eventids.h.

**12.192.2.59 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.

**12.192.2.60 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.

**12.192.2.61 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 752 of file cfe\_tbl\_eventids.h.

**12.192.2.62 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.

**12.192.2.63 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 380 of file cfe\_tbl\_eventids.h.

**12.192.2.64 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.

**12.192.2.65 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.

**12.192.2.66 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 391 of file cfe\_tbl\_eventids.h.

**12.192.2.67 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 402 of file cfe\_tbl\_eventids.h.

**12.192.2.68 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 438 of file cfe\_tbl\_eventids.h.

**12.192.2.69 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 497 of file cfe\_tbl\_eventids.h.

## 12.193 cfe/modules/tbl/fsw/inc/cfe\_tbl\_fcncodes.h File Reference

```
#include "cfe_tbl_fcncode_values.h"
```

### Macros

#### Table Services Command Codes

- #define CFE\_TBL\_NOOP\_CC CFE\_TBL\_CCVAL(NOP)
- #define CFE\_TBL\_RESET\_COUNTERS\_CC CFE\_TBL\_CCVAL(RESET\_COUNTERS)
- #define CFE\_TBL\_LOAD\_CC CFE\_TBL\_CCVAL(LOAD)
- #define CFE\_TBL\_DUMP\_CC CFE\_TBL\_CCVAL(DUMP)
- #define CFE\_TBL\_VALIDATE\_CC CFE\_TBL\_CCVAL(VALIDATE)
- #define CFE\_TBL\_ACTIVATE\_CC CFE\_TBL\_CCVAL(ACTIVATE)
- #define CFE\_TBL\_DUMP\_REGISTRY\_CC CFE\_TBL\_CCVAL(DUMP\_REGISTRY)
- #define CFE\_TBL\_SEND\_REGISTRY\_CC CFE\_TBL\_CCVAL(SEND\_REGISTRY)
- #define CFE\_TBL\_DELETE\_CDS\_CC CFE\_TBL\_CCVAL(DELETE\_CDS)
- #define CFE\_TBL\_ABORT\_LOAD\_CC CFE\_TBL\_CCVAL(ABORT\_LOAD)

### 12.193.1 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.

## 12.193.2 Macro Definition Documentation

### 12.193.2.1 CFE\_TBL\_ABORT\_LOAD\_CC #define CFE\_TBL\_ABORT\_LOAD\_CC CFE\_TBL\_CCVAL(ABORT\_LOAD)

**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 463 of file cfe\_tbl\_fcncodes.h.

**12.193.2.2 CFE\_TBL\_ACTIVATE\_CC** #define CFE\_TBL\_ACTIVATE\_CC CFE\_TBL\_CCVAL(activate)**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 301 of file [cfe\\_tbl\\_fcncodes.h](#).

**12.193.2.3 CFE\_TBL\_DELETE\_CDS\_CC** #define CFE\_TBL\_DELETE\_CDS\_CC CFE\_TBL\_CCVAL(delete\_cds)**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 424 of file cfe\_tbl\_fcncodes.h.

## 12.193.2.4 CFE\_TBL\_DUMP\_CC #define CFE\_TBL\_DUMP\_CC [CFE\\_TBL\\_CCVAL](#) (DUMP)

### 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 204 of file `cfe_tbl_fcncodes.h`.

**12.193.2.5 CFE\_TBL\_DUMP\_REGISTRY\_CC** #define CFE\_TBL\_DUMP\_REGISTRY\_CC [CFE\\_TBL\\_CCVAL](#)(DUMP\_↔  
REGISTRY)

**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 345 of file cfe\_tbl\_fcncodes.h.

## 12.193.2.6 CFE\_TBL\_LOAD\_CC #define CFE\_TBL\_LOAD\_CC CFE\_TBL\_CCVAL(LOAD)

**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 161 of file cfe\_tbl\_fcncodes.h.

### 12.193.2.7 CFE\_TBL\_NOOP\_CC #define CFE\_TBL\_NOOP\_CC CFE\_TBL\_CCVAL(NOOP)

**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 70 of file cfe\_tbl\_fcncodes.h.

**12.193.2.8 CFE\_TBL\_RESET\_COUNTERS\_CC** #define CFE\_TBL\_RESET\_COUNTERS\_CC CFE\_TBL\_CCVAL (RESET←\_COUNTERS)

### 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 111 of file cfe\_tbl\_fcncodes.h.

**12.193.2.9 CFE\_TBL\_SEND\_REGISTRY\_CC** #define CFE\_TBL\_SEND\_REGISTRY\_CC CFE\_TBL\_CCVAL (SEND\_←  
REGISTRY)

**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 380 of file cfe\_tbl\_fcncodes.h.

**12.193.2.10 CFE\_TBL\_VALIDATE\_CC** #define CFE\_TBL\_VALIDATE\_CC CFE\_TBL\_CCVAL (VALIDATE)

**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 261 of file `cfe_tbl_fcncodes.h`.

## 12.194 cfe/modules/tbl/fsw/inc/cfe\_tbl\_interface\_cfg.h File Reference

#include "cfe\_tbl\_interface\_cfg\_values.h"

## Macros

- #define CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH CFE\_MISSION\_TBL\_CFGVAL(MAX\_NAME\_LENGTH)
- #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH 16
- #define CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN CFE\_MISSION\_TBL\_CFGVAL(MAX\_FULL\_NAME\_LEN)
- #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN 40

### 12.194.1 Detailed Description

CFE Event Services (CFE\_TBL) Application Mission Configuration Header File

This is a compatibility header for the "mission\_cfg.h" file that has traditionally provided public config definitions for each CFS app.

#### Note

This file may be overridden/superceded by mission-provided definitions either by overriding this header or by generating definitions from a command/data dictionary tool.

### 12.194.2 Macro Definition Documentation

#### 12.194.2.1 CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN #define CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN CFE\_MISSION\_TBL\_CFGVAL(FULL\_NAME\_LEN)

**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 71 of file cfe\_tbl\_interface\_cfg.h.

#### 12.194.2.2 CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH #define CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH CFE\_MISSION\_TBL\_CFGVAL(NAME\_LENGTH)

**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 50 of file cfe\_tbl\_interface\_cfg.h.

**12.194.2.3 DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN** #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN 40

Definition at line 74 of file cfe\_tbl\_interface\_cfg.h.

**12.194.2.4 DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH** #define DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH 16

Definition at line 51 of file cfe\_tbl\_interface\_cfg.h.

## 12.195 cfe/modules/tbl/fsw/inc/cfe\_tbl\_internal\_cfg.h File Reference

```
#include "cfe_tbl_mission_cfg.h"
#include "cfe_tbl_internal_cfg_values.h"
```

### Macros

- #define CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY CFE\_PLATFORM\_TBL\_CFGVAL(START\_TASK\_PRIORITY)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY 70
- #define CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(START\_TASK\_STACK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES CFE\_PLATFORM\_TBL\_CFGVAL(BUF\_MEMORY\_BYTES)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES 524288
- #define CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_DBL\_TABLE\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE 16384
- #define CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_SNGL\_TABLE\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE 16384
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_TABLES)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128
- #define CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_CRITICAL\_TABLES)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES 32
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_HANDLES)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256
- #define CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_SIMULTANEOUS\_LOADS)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS 4
- #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_NUM\_VALIDATIONS)

- #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS 10
- #define CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE CFE\_PLATFORM\_TBL\_CFGVAL(DEFAULT\_←  
REG\_DUMP\_FILE)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE "/ram/cfe\_tbl\_reg.log"
- #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_←  
COUNT)
- #define DEFAULT\_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 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_1)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 (0x42)
- #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_2)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b',  
'c', 'd'))
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_←  
COUNT)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT 0
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_1)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 (1)
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_2)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b',  
'c', 'd'))
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_3)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 0
- #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_4)
- #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 0

### 12.195.1 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.

### 12.195.2 Macro Definition Documentation

#### 12.195.2.1 CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES #define CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES CFE\_PLATFORM\_TBL\_← \_MEMORY\_BYTES)

**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 83 of file `cfe_tbl_internal_cfg.h`.

**12.195.2.2 CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE** `#define CFE_PLATFORM_TBL_DEFAULT_REG_DUMP_FILE CFE_PLATFORM_TBL_CFGVAL(DEFAULT_REG_DUMP_FILE)`

**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 205 of file `cfe_tbl_internal_cfg.h`.

**12.195.2.3 CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES** `#define CFE_PLATFORM_TBL_MAX_CRITICAL_TABLES CFE_PLATFORM_TBL_CFGVAL(MAX_CRITICAL_TABLES)`

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

## Description:

Defines the maximum number of critical tables supported by this processor's Table Services.

## Limits

This number must be less than 32767. It should be recognized that this parameter determines the size of the Critical Table Registry which is maintained in the Critical Data Store. An excessively high number will waste Critical Data Store memory. Therefore, this number must not exceed the value defined in `CFE_PLATFORM_ES_CDS_MAX_NUM_ENTRIES`.

Definition at line 142 of file `cfe_tbl_internal_cfg.h`.

**12.195.2.4 CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE** `#define CFE_PLATFORM_TBL_MAX_DBL_TABLE_SIZE CFE_PLATFORM_TBL_CFGVAL(MAX_DBL_TABLE_SIZE)`

**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 96 of file `cfe_tbl_internal_cfg.h`.

**12.195.2.5 CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES CFE\_PLATFORM\_TBL\_CFGVA\_NUM\_HANDLES)

**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 156 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.6 CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES CFE\_PLATFORM\_TBL\_CFGVA\_NUM\_TABLES)

**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 127 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.7 CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS** #define CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS CFE\_PLATFORM\_TBL\_CFGVA\_NUM\_VALIDATIONS)

**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 191 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.8 CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS** #define CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_SIMULTANEOUS\_LOADS)

**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 172 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.9 CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE** #define CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(MAX\_SNGL\_TABLE\_SIZE)

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

**Description:**

Defines the maximum allowed size (in bytes) of a single buffered table. **NOTE:** This size determines the size of all shared table buffers. Therefore, this size will be multiplied by [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) below when allocating memory for shared tables.

**Limits**

The cFE does not place a limit on the size of this parameter but it must be small enough to allow for [CFE\\_PLATFORM\\_TBL\\_MAX\\_SIMULTANEOUS\\_LOADS](#) number of tables to fit into [CFE\\_PLATFORM\\_TBL\\_BUF\\_MEMORY\\_BYTES](#).

Definition at line 113 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.10 CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY CFE\_PLATFORM\_TBL\_CFGVAL(START\_TASK\_PRIORITY)

**Purpose** Define TBL Task Priority

**Description:**

Defines the cFE\_TBL Task priority.

**Limits**

Not Applicable

Definition at line 50 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.11 CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TBL\_START\_TASK\_←  
STACK\_SIZE CFE\_PLATFORM\_TBL\_CFGVAL(START\_TASK\_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 66 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.12 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 229 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.13 CFE\_PLATFORM\_TBL\_VALID\_PRID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_←  
\_PRID\_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 282 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.14 CFE\_PLATFORM\_TBL\_VALID\_PRID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_←  
\_PRID\_2)

Definition at line 285 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.15 CFE\_PLATFORM\_TBL\_VALID\_PRID\_3** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_←  
\_PRID\_3)

Definition at line 288 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.16 CFE\_PLATFORM\_TBL\_VALID\_PRID\_4** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_4)

Definition at line 291 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.17 CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT** #define CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_PRID\_COUNT)

**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 267 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.18 CFE\_PLATFORM\_TBL\_VALID\_SCID\_1** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_1)

**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 244 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.19 CFE\_PLATFORM\_TBL\_VALID\_SCID\_2** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_2)

Definition at line 247 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.20 CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT** #define CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT CFE\_PLATFORM\_TBL\_CFGVAL(VALID\_SCID\_COUNT)

**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 225 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.21 DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES 524288

Definition at line 84 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.22 DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE** #define DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE "/ram/cfe\_tbl\_reg.log"

Definition at line 206 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.23 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES 32

Definition at line 143 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.24 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE 16384

Definition at line 97 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.25 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES 256

Definition at line 157 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.26 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES 128

Definition at line 128 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.27 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS 10

Definition at line 192 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.28 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS** #define DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS 4

Definition at line 173 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.29 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_↪  
TBL\_MAX\_SNGL\_TABLE\_SIZE 16384  
Definition at line 114 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.30 DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_↪  
TBL\_START\_TASK\_PRIORITY 70  
Definition at line 51 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.31 DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_↪  
\_TBL\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE  
Definition at line 67 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.32 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_↪  
PRID\_1 (1)  
Definition at line 283 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.33 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_↪  
PRID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))  
Definition at line 286 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.34 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_↪  
PRID\_3 0  
Definition at line 289 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.35 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_↪  
PRID\_4 0  
Definition at line 292 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.36 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT** #define DEFAULT\_CFE\_PLATFORM\_TBL\_↪  
VALID\_PRID\_COUNT 0  
Definition at line 268 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.37 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_↪  
SCID\_1 (0x42)  
Definition at line 245 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.38 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_2** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_↪  
SCID\_2 (CFE\_PLATFORM\_TBL\_U32FROM4CHARS('a', 'b', 'c', 'd'))  
Definition at line 248 of file cfe\_tbl\_internal\_cfg.h.

**12.195.2.39 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT** #define DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT 0  
Definition at line 226 of file cfe\_tbl\_internal\_cfg.h.

## 12.196 cfe/modules/tbl/fsw/inc/cfe\_tbl\_topicids.h File Reference

```
#include "cfe_tbl_topicid_values.h"
```

### Macros

- #define CFE\_MISSION\_TBL\_CMD\_TOPICID CFE\_MISSION\_TBL\_TIDVAL(CMD)
- #define DEFAULT\_CFE\_MISSION\_TBL\_CMD\_TOPICID 4
- #define CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID CFE\_MISSION\_TBL\_TIDVAL(SEND\_HK)
- #define DEFAULT\_CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID 12
- #define CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID CFE\_MISSION\_TBL\_TIDVAL(HK\_TLM)
- #define DEFAULT\_CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID 4
- #define CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID CFE\_MISSION\_TBL\_TIDVAL(REG\_TLM)
- #define DEFAULT\_CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID 12

### 12.196.1 Detailed Description

CFE Table Services (CFE\_TBL) Application Topic IDs

### 12.196.2 Macro Definition Documentation

**12.196.2.1 CFE\_MISSION\_TBL\_CMD\_TOPICID** #define CFE\_MISSION\_TBL\_CMD\_TOPICID CFE\_MISSION\_TBL\_TIDVAL (CMD)

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE command messages

**Limits**

Not Applicable

Definition at line 37 of file cfe\_tbl\_topicids.h.

**12.196.2.2 CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID** #define CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID CFE\_MISSION\_TBL\_TIDVAL (HK\_TLM)

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE telemetry messages

**Limits**

Not Applicable

Definition at line 52 of file cfe\_tbl\_topicids.h.

**12.196.2.3 CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID** #define CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID CFE\_MISSION\_TBL\_TIDVAL (RE\_TLM)  
Definition at line 55 of file cfe\_tbl\_topicids.h.

**12.196.2.4 CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID** #define CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID CFE\_MISSION\_TBL\_TIDVAL (SE\_HK)  
Definition at line 40 of file cfe\_tbl\_topicids.h.

**12.196.2.5 DEFAULT\_CFE\_MISSION\_TBL\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TBL\_CMD\_TOPICID 4  
Definition at line 38 of file cfe\_tbl\_topicids.h.

**12.196.2.6 DEFAULT\_CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID 4  
Definition at line 53 of file cfe\_tbl\_topicids.h.

**12.196.2.7 DEFAULT\_CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID 12  
Definition at line 56 of file cfe\_tbl\_topicids.h.

**12.196.2.8 DEFAULT\_CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID 12  
Definition at line 41 of file cfe\_tbl\_topicids.h.

## 12.197 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.*

### 12.197.1 Detailed Description

Declarations and prototypes for cfe\_time\_extern\_typedefs module

### 12.197.2 Typedef Documentation

#### 12.197.2.1 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.

**12.197.2.2 CFE\_TIME\_ClockState\_Enum\_t** `typedef int16 CFE_TIME_ClockState_Enum_t`  
Enumerated types identifying the quality of the current time.

See also

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.

**12.197.2.3 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.

**12.197.2.4 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.

**12.197.2.5 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.

**12.197.2.6 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.

**12.197.2.7 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.

**12.197.2.8 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.

### 12.197.3 Enumeration Type Documentation

**12.197.3.1 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.

**12.197.3.2 CFE\_TIME\_ClockState** `enum CFE_TIME_ClockState`  
 Label definitions associated with `CFE_TIME_ClockState_Enum_t`.

#### Enumerator

<code>CFE_TIME_ClockState_INVALID</code>	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.
<code>CFE_TIME_ClockState_VALID</code>	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.
<code>CFE_TIME_ClockState_FLYWHEEL</code>	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.

**12.197.3.3 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.

**12.197.3.4 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.

**12.197.3.5 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.

**12.197.3.6 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.

### 12.197.3.7 CFE\_TIME\_ToneSignalSelect enum `CFE_TIME_ToneSignalSelect`

Label definitions associated with `CFE_TIME_ToneSignalSelect_Enum_t`.

Enumerator

<code>CFE_TIME_ToneSignalSelect_PRIMARY</code>	Primary Source.
<code>CFE_TIME_ToneSignalSelect_REDUNDANT</code>	Redundant Source.

Definition at line 193 of file default\_cfe\_time\_extern\_typedefs.h.

## 12.198 cfe/modules/time/config/default\_cfe\_time\_fcncode\_values.h File Reference

### Macros

- `#define CFE_TIME_CCVAL(x) CFE_TIME_FunctionCode_##x`

### Enumerations

- enum `CFE_TIME_FunctionCode_` {
   
`CFE_TIME_FunctionCode_NOOP = 0, CFE_TIME_FunctionCode_RESET_COUNTERS = 1, CFE_TIME_FunctionCode_SEND_D`
  
`= 2, CFE_TIME_FunctionCode_SET_SOURCE = 3,`
  
`CFE_TIME_FunctionCode_SET_STATE = 4, CFE_TIME_FunctionCode_ADD_DELAY = 5, CFE_TIME_FunctionCode_SUB_DELA`
  
`= 6, CFE_TIME_FunctionCode_SET_TIME = 7,`
  
`CFE_TIME_FunctionCode_SET_MET = 8, CFE_TIME_FunctionCode_SET_STCF = 9, CFE_TIME_FunctionCode_SET_LEAP_S`
  
`= 10, CFE_TIME_FunctionCode_ADD_ADJUST = 11,`
  
`CFE_TIME_FunctionCode_SUB_ADJUST = 12, CFE_TIME_FunctionCode_ADD_ONE_HZ_ADJUSTMENT =`
  
`13, CFE_TIME_FunctionCode_SUB_ONE_HZ_ADJUSTMENT = 14, CFE_TIME_FunctionCode_SET_SIGNAL`
  
`= 15 }`

### 12.198.1 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.

### 12.198.2 Macro Definition Documentation

#### 12.198.2.1 CFE\_TIME\_CCVAL `#define CFE_TIME_CCVAL(`

`x ) CFE_TIME_FunctionCode_##x`

Definition at line 35 of file default\_cfe\_time\_fcncode\_values.h.

### 12.198.3 Enumeration Type Documentation

#### 12.198.3.1 CFE\_TIME\_FunctionCode\_ enum `CFE_TIME_FunctionCode_`

**Enumerator**

CFE_TIME_FunctionCode_NOOP
CFE_TIME_FunctionCode_RESET_COUNTERS
CFE_TIME_FunctionCode_SEND_DIAGNOSTIC
CFE_TIME_FunctionCode_SET_SOURCE
CFE_TIME_FunctionCode_SET_STATE
CFE_TIME_FunctionCode_ADD_DELAY
CFE_TIME_FunctionCode_SUB_DELAY
CFE_TIME_FunctionCode_SET_TIME
CFE_TIME_FunctionCode_SET_MET
CFE_TIME_FunctionCode_SET_STCF
CFE_TIME_FunctionCode_SET_LEAP_SECONDS
CFE_TIME_FunctionCode_ADD_ADJUST
CFE_TIME_FunctionCode_SUB_ADJUST
CFE_TIME_FunctionCode_ADD_ONE_HZ_ADJUSTMENT
CFE_TIME_FunctionCode_SUB_ONE_HZ_ADJUSTMENT
CFE_TIME_FunctionCode_SET_SIGNAL

Definition at line 37 of file default\_cfe\_time\_fcncode\_values.h.

## **12.199 cfe/modules/time/config/default\_cfe\_time\_interface\_cfg\_values.h File Reference**

### **Macros**

- #define CFE\_MISSION\_TIME\_CFGVAL(x) DEFAULT\_CFE\_MISSION\_TIME\_##x

#### **12.199.1 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.

#### **12.199.2 Macro Definition Documentation**

**12.199.2.1 CFE\_MISSION\_TIME\_CFGVAL** #define CFE\_MISSION\_TIME\_CFGVAL(  
  x ) DEFAULT\_CFE\_MISSION\_TIME\_##x

Definition at line 36 of file default\_cfe\_time\_interface\_cfg\_values.h.

## **12.200 cfe/modules/time/config/default\_cfe\_time\_internal\_cfg\_values.h File Reference**

### **Macros**

- #define CFE\_PLATFORM\_TIME\_CFGVAL(x) DEFAULT\_CFE\_PLATFORM\_TIME\_##x

### 12.200.1 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.

### 12.200.2 Macro Definition Documentation

**12.200.2.1 CFE\_PLATFORM\_TIME\_CFGVAL** #define CFE\_PLATFORM\_TIME\_CFGVAL ( x ) DEFAULT\_CFE\_PLATFORM\_TIME\_##x

Definition at line 36 of file default\_cfe\_time\_internal\_cfg\_values.h.

## 12.201 cfe/modules/time/config/default\_cfe\_time\_mission\_cfg.h File Reference

```
#include "cfe_time_interface_cfg.h"
```

### 12.201.1 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.

## 12.202 cfe/modules/time/config/default\_cfe\_time\_msg.h File Reference

```
#include "cfe_mission_cfg.h"
#include "cfe_time_fcncodes.h"
#include "cfe_time_msgdefs.h"
#include "cfe_time_msgstruct.h"
```

### 12.202.1 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.

## 12.203 cfe/modules/time/config/default\_cfe\_time\_msgdefs.h File Reference

```
#include "common_types.h"
#include "cfe_mission_cfg.h"
#include "cfe_time_extern_typedefs.h"
#include "cfe_time_fcncodes.h"
```

### Data Structures

- struct [CFE\\_TIME\\_LeapsCmd\\_Payload](#)  
*Set leap seconds command payload.*
- struct [CFE\\_TIME\\_StateCmd\\_Payload](#)  
*Set clock state command payload.*
- struct [CFE\\_TIME\\_SourceCmd\\_Payload](#)  
*Set time data source command payload.*
- struct [CFE\\_TIME\\_SignalCmd\\_Payload](#)  
*Set tone signal source command payload.*
- struct [CFE\\_TIME\\_TimeCmd\\_Payload](#)  
*Generic seconds, microseconds command payload.*
- struct [CFE\\_TIME\\_OneHzAdjustmentCmd\\_Payload](#)  
*Generic seconds, subseconds command payload.*
- struct [CFE\\_TIME\\_ToneDataCmd\\_Payload](#)  
*Time at tone data command payload.*
- struct [CFE\\_TIME\\_HousekeepingTlm\\_Payload](#)
- struct [CFE\\_TIME\\_DiagnosticTlm\\_Payload](#)

### Macros

- #define [CFE\\_TIME\\_FLAG\\_CLKSET](#) 0x8000  
*The spacecraft time has been set.*
- #define [CFE\\_TIME\\_FLAG\\_FLYING](#) 0x4000  
*This instance of Time Services is flywheeling.*
- #define [CFE\\_TIME\\_FLAG\\_SRCINT](#) 0x2000  
*The clock source is set to "internal".*
- #define [CFE\\_TIME\\_FLAG\\_SIGPRI](#) 0x1000  
*The clock signal is set to "primary".*
- #define [CFE\\_TIME\\_FLAG\\_SRVFLY](#) 0x0800  
*The Time Server is in flywheel mode.*
- #define [CFE\\_TIME\\_FLAG\\_CMDFLY](#) 0x0400  
*This instance of Time Services was commanded into flywheel mode.*
- #define [CFE\\_TIME\\_FLAG\\_ADDADJ](#) 0x0200  
*One time STCF Adjustment is to be done in positive direction.*
- #define [CFE\\_TIME\\_FLAG\\_ADD1HZ](#) 0x0100  
*1 Hz STCF Adjustment is to be done in a positive direction*
- #define [CFE\\_TIME\\_FLAG\\_ADDTCL](#) 0x0080  
*Time Client Latency is applied in a positive direction.*
- #define [CFE\\_TIME\\_FLAG\\_SERVER](#) 0x0040  
*This instance of Time Services is a Time Server.*
- #define [CFE\\_TIME\\_FLAG\\_GDTONE](#) 0x0020

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

- #define CFE\_TIME\_FLAG\_REFERR 0x0010
  - GetReference read error, will be set if unable to get a consistent ref value.*
- #define CFE\_TIME\_FLAG\_UNUSED 0x000F
  - Reserved flags - should be zero.*

## Typedefs

- typedef struct CFE\_TIME\_LeapsCmd\_Payload CFE\_TIME\_LeapsCmd\_Payload\_t
  - Set leap seconds command payload.*
- typedef struct CFE\_TIME\_StateCmd\_Payload CFE\_TIME\_StateCmd\_Payload\_t
  - Set clock state command payload.*
- typedef struct CFE\_TIME\_SourceCmd\_Payload CFE\_TIME\_SourceCmd\_Payload\_t
  - Set time data source command payload.*
- typedef struct CFE\_TIME\_SignalCmd\_Payload CFE\_TIME\_SignalCmd\_Payload\_t
  - Set tone signal source command payload.*
- typedef struct CFE\_TIME\_TimeCmd\_Payload CFE\_TIME\_TimeCmd\_Payload\_t
  - Generic seconds, microseconds command payload.*
- typedef struct CFE\_TIME\_OneHzAdjustmentCmd\_Payload CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t
  - Generic seconds, subseconds command payload.*
- typedef struct CFE\_TIME\_ToneDataCmd\_Payload CFE\_TIME\_ToneDataCmd\_Payload\_t
  - Time at tone data command payload.*
- typedef struct CFE\_TIME\_HousekeepingTlm\_Payload CFE\_TIME\_HousekeepingTlm\_Payload\_t
- typedef struct CFE\_TIME\_DiagnosticTlm\_Payload CFE\_TIME\_DiagnosticTlm\_Payload\_t

### 12.203.1 Detailed Description

Specification for the CFE Time Services (CFE\_TIME) command and telemetry message payloads and constant definitions.

### 12.203.2 Typedef Documentation

#### 12.203.2.1 CFE\_TIME\_DiagnosticTlm\_Payload\_t `typedef struct CFE_TIME_DiagnosticTlm_Payload CFE_TIME_DiagnosticTlm_Payload_t`

**Name** Time Services Diagnostics Packet

#### 12.203.2.2 CFE\_TIME\_HousekeepingTlm\_Payload\_t `typedef struct CFE_TIME_HousekeepingTlm_Payload CFE_TIME_HousekeepingTlm_Payload_t`

**Name** Time Services Housekeeping Packet

#### 12.203.2.3 CFE\_TIME\_LeapsCmd\_Payload\_t `typedef struct CFE_TIME_LeapsCmd_Payload CFE_TIME_LeapsCmd_Payload_t`

Set leap seconds command payload.

**12.203.2.4 CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t** `typedef struct CFE_TIME_OneHzAdjustmentCmd_Payload CFE_TIME_OneHzAdjustmentCmd_Payload_t`  
Generic seconds, subseconds command payload.

**12.203.2.5 CFE\_TIME\_SignalCmd\_Payload\_t** `typedef struct CFE_TIME_SignalCmd_Payload CFE_TIME_SignalCmd_Payload_t`  
Set tone signal source command payload.

**12.203.2.6 CFE\_TIME\_SourceCmd\_Payload\_t** `typedef struct CFE_TIME_SourceCmd_Payload CFE_TIME_SourceCmd_Payload_t`  
Set time data source command payload.

**12.203.2.7 CFE\_TIME\_StateCmd\_Payload\_t** `typedef struct CFE_TIME_StateCmd_Payload CFE_TIME_StateCmd_Payload_t`  
Set clock state command payload.

**12.203.2.8 CFE\_TIME\_TimeCmd\_Payload\_t** `typedef struct CFE_TIME_TimeCmd_Payload CFE_TIME_TimeCmd_Payload_t`  
Generic seconds, microseconds command payload.

**12.203.2.9 CFE\_TIME\_ToneDataCmd\_Payload\_t** `typedef struct CFE_TIME_ToneDataCmd_Payload CFE_TIME_ToneDataCmd_Payload_t`  
Time at tone data command payload.

## 12.204 cfe/modules/time/config/default\_cfe\_time\_msgid\_values.h File Reference

```
#include "cfe_core_api_base_msgids.h"
#include "cfe_time_topicids.h"
```

### Macros

- `#define CFE_PLATFORM_TIME_CMD_MIDVAL(x) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_`  
`MISSION_TIME_##x##_TOPICID)`
- `#define CFE_PLATFORM_TIME_TLM_MIDVAL(x) CFE_PLATFORM_TLM_TOPICID_TO_MIDV(CFE_`  
`MISSION_TIME_##x##_TOPICID)`
- `#define CFE_PLATFORM_TIME_GLBCMD_MIDVAL(x) CFE_GLOBAL_CMD_TOPICID_TO_MIDV(CFE_`  
`MISSION_TIME_##x##_TOPICID)`

### 12.204.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Message IDs

### 12.204.2 Macro Definition Documentation

**12.204.2.1 CFE\_PLATFORM\_TIME\_CMD\_MIDVAL** `#define CFE_PLATFORM_TIME_CMD_MIDVAL(`  
`x ) CFE_PLATFORM_CMD_TOPICID_TO_MIDV(CFE_MISSION_TIME_##x##_TOPICID)`  
Definition at line 29 of file default\_cfe\_time\_msgid\_values.h.

**12.204.2.2 CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL** #define CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL( x ) CFE\_GLOBAL\_CMD\_TOPICID\_TO\_MIDV(CFE\_MISSION\_TIME\_##x##\_TOPICID)

Definition at line 33 of file default\_cfe\_time\_msgid\_values.h.

**12.204.2.3 CFE\_PLATFORM\_TIME\_TLM\_MIDVAL** #define CFE\_PLATFORM\_TIME\_TLM\_MIDVAL( x ) CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV(CFE\_MISSION\_TIME\_##x##\_TOPICID)

Definition at line 30 of file default\_cfe\_time\_msgid\_values.h.

## 12.205 cfe/modules/time/config/default\_cfe\_time\_msgids.h File Reference

```
#include "cfe_time_msgid_values.h"
```

### Macros

- #define CFE\_TIME\_CMD\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL(CMD)
- #define CFE\_TIME\_SEND\_HK\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL(SEND\_HK)
- #define CFE\_TIME\_TONE\_CMD\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL(TONE\_CMD)
- #define CFE\_TIME\_ONEHZ\_CMD\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL(ONEHZ\_CMD)
- #define CFE\_TIME\_DATA\_CMD\_MID CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL(DATA\_CMD)
- #define CFE\_TIME\_SEND\_CMD\_MID CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL(SEND\_CMD)
- #define CFE\_TIME\_HK\_TLM\_MID CFE\_PLATFORM\_TIME\_TLM\_MIDVAL(HK\_TLM)
- #define CFE\_TIME\_DIAG\_TLM\_MID CFE\_PLATFORM\_TIME\_TLM\_MIDVAL(DIAG\_TLM)
- #define CFE\_TIME\_1HZ\_CMD\_MID CFE\_TIME\_ONEHZ\_CMD\_MID

### 12.205.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Message IDs

### 12.205.2 Macro Definition Documentation

**12.205.2.1 CFE\_TIME\_1HZ\_CMD\_MID** #define CFE\_TIME\_1HZ\_CMD\_MID CFE\_TIME\_ONEHZ\_CMD\_MID  
Definition at line 54 of file default\_cfe\_time\_msgids.h.

**12.205.2.2 CFE\_TIME\_CMD\_MID** #define CFE\_TIME\_CMD\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL(CMD)  
Definition at line 31 of file default\_cfe\_time\_msgids.h.

**12.205.2.3 CFE\_TIME\_DATA\_CMD\_MID** #define CFE\_TIME\_DATA\_CMD\_MID CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL(DATA↔\_CMD)

Definition at line 39 of file default\_cfe\_time\_msgids.h.

**12.205.2.4 CFE\_TIME\_DIAG\_TLM\_MID** #define CFE\_TIME\_DIAG\_TLM\_MID CFE\_PLATFORM\_TIME\_TLM\_MIDVAL(DIAG↔\_TLM)

Definition at line 46 of file default\_cfe\_time\_msgids.h.

**12.205.2.5 CFE\_TIME\_HK\_TLM\_MID** #define CFE\_TIME\_HK\_TLM\_MID CFE\_PLATFORM\_TIME\_TLM\_MIDVAL (HK↔TLM)

Definition at line 45 of file default\_cfe\_time\_msgids.h.

**12.205.2.6 CFE\_TIME\_ONEHZ\_CMD\_MID** #define CFE\_TIME\_ONEHZ\_CMD\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL (ONEHZ↔\_CMD)

Definition at line 34 of file default\_cfe\_time\_msgids.h.

**12.205.2.7 CFE\_TIME\_SEND\_CMD\_MID** #define CFE\_TIME\_SEND\_CMD\_MID CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL (SEND↔\_CMD)

Definition at line 40 of file default\_cfe\_time\_msgids.h.

**12.205.2.8 CFE\_TIME\_SEND\_HK\_MID** #define CFE\_TIME\_SEND\_HK\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL (SEND↔\_HK)

Definition at line 32 of file default\_cfe\_time\_msgids.h.

**12.205.2.9 CFE\_TIME\_TONE\_CMD\_MID** #define CFE\_TIME\_TONE\_CMD\_MID CFE\_PLATFORM\_TIME\_CMD\_MIDVAL (TONE↔\_CMD)

Definition at line 33 of file default\_cfe\_time\_msgids.h.

## 12.206 cfe/modules/time/config/default\_cfe\_time\_msgstruct.h File Reference

```
#include "cfe_time_msgdefs.h"
#include "cfe_msg_hdr.h"
```

### Data Structures

- struct [CFE\\_TIME\\_NoopCmd](#)
- struct [CFE\\_TIME\\_ResetCountersCmd](#)
- struct [CFE\\_TIME\\_SendDiagnosticCmd](#)
- struct [CFE\\_TIME\\_OneHzCmd](#)
- struct [CFE\\_TIME\\_ToneSignalCmd](#)
- struct [CFE\\_TIME\\_FakeToneCmd](#)
- struct [CFE\\_TIME\\_SendHkCmd](#)
- struct [CFE\\_TIME\\_SetLeapSecondsCmd](#)  
*Set leap seconds command.*
- struct [CFE\\_TIME\\_SetStateCmd](#)  
*Set clock state command.*
- struct [CFE\\_TIME\\_SetSourceCmd](#)  
*Set time data source command.*
- struct [CFE\\_TIME\\_SetSignalCmd](#)  
*Set tone signal source command.*
- struct [CFE\\_TIME\\_AddDelayCmd](#)
- struct [CFE\\_TIME\\_SubDelayCmd](#)
- struct [CFE\\_TIME\\_SetMETCCmd](#)
- struct [CFE\\_TIME\\_SetSTCFCmd](#)

- struct [CFE\\_TIME\\_AddAdjustCmd](#)
- struct [CFE\\_TIME\\_SubAdjustCmd](#)
- struct [CFE\\_TIME\\_SetTimeCmd](#)
- struct [CFE\\_TIME\\_AddOneHzAdjustmentCmd](#)
- struct [CFE\\_TIME\\_SubOneHzAdjustmentCmd](#)
- struct [CFE\\_TIME\\_ToneDataCmd](#)  
*Time at tone data command.*
- struct [CFE\\_TIME\\_HousekeepingTlm](#)
- struct [CFE\\_TIME\\_DiagnosticTlm](#)

## Typedefs

- typedef struct [CFE\\_TIME\\_NoopCmd](#) CFE\_TIME\_NoopCmd\_t
- typedef struct [CFE\\_TIME\\_ResetCountersCmd](#) CFE\_TIME\_ResetCountersCmd\_t
- typedef struct [CFE\\_TIME\\_SendDiagnosticCmd](#) CFE\_TIME\_SendDiagnosticCmd\_t
- typedef struct [CFE\\_TIME\\_OneHzCmd](#) CFE\_TIME\_OneHzCmd\_t
- typedef struct [CFE\\_TIME\\_ToneSignalCmd](#) CFE\_TIME\_ToneSignalCmd\_t
- typedef struct [CFE\\_TIME\\_FakeToneCmd](#) CFE\_TIME\_FakeToneCmd\_t
- typedef struct [CFE\\_TIME\\_SendHkCmd](#) CFE\_TIME\_SendHkCmd\_t
- typedef struct [CFE\\_TIME\\_SetLeapSecondsCmd](#) CFE\_TIME\_SetLeapSecondsCmd\_t  
*Set leap seconds command.*
- typedef struct [CFE\\_TIME\\_SetStateCmd](#) CFE\_TIME\_SetStateCmd\_t  
*Set clock state command.*
- typedef struct [CFE\\_TIME\\_SetSourceCmd](#) CFE\_TIME\_SetSourceCmd\_t  
*Set time data source command.*
- typedef struct [CFE\\_TIME\\_SetSignalCmd](#) CFE\_TIME\_SetSignalCmd\_t  
*Set tone signal source command.*
- typedef struct [CFE\\_TIME\\_AddDelayCmd](#) CFE\_TIME\_AddDelayCmd\_t
- typedef struct [CFE\\_TIME\\_SubDelayCmd](#) CFE\_TIME\_SubDelayCmd\_t
- typedef struct [CFE\\_TIME\\_SetMETCmd](#) CFE\_TIME\_SetMETCmd\_t
- typedef struct [CFE\\_TIME\\_SetSTCFCmd](#) CFE\_TIME\_SetSTCFCmd\_t
- typedef struct [CFE\\_TIME\\_AddAdjustCmd](#) CFE\_TIME\_AddAdjustCmd\_t
- typedef struct [CFE\\_TIME\\_SubAdjustCmd](#) CFE\_TIME\_SubAdjustCmd\_t
- typedef struct [CFE\\_TIME\\_SetTimeCmd](#) CFE\_TIME\_SetTimeCmd\_t
- typedef struct [CFE\\_TIME\\_AddOneHzAdjustmentCmd](#) CFE\_TIME\_AddOneHzAdjustmentCmd\_t
- typedef struct [CFE\\_TIME\\_SubOneHzAdjustmentCmd](#) CFE\_TIME\_SubOneHzAdjustmentCmd\_t
- typedef struct [CFE\\_TIME\\_ToneDataCmd](#) CFE\_TIME\_ToneDataCmd\_t  
*Time at tone data command.*
- typedef struct [CFE\\_TIME\\_HousekeepingTlm](#) CFE\_TIME\_HousekeepingTlm\_t
- typedef struct [CFE\\_TIME\\_DiagnosticTlm](#) CFE\_TIME\_DiagnosticTlm\_t

### 12.206.1 Detailed Description

cFE Executive Services (TIME) Command and Telemetry packet definition file.

### 12.206.2 Typedef Documentation

#### 12.206.2.1 CFE\_TIME\_AddAdjustCmd\_t

**12.206.2.2 CFE\_TIME\_AddDelayCmd\_t** `typedef struct CFE_TIME_AddDelayCmd CFE_TIME_AddDelayCmd_t`

**12.206.2.3 CFE\_TIME\_AddOneHzAdjustmentCmd\_t** `typedef struct CFE_TIME_AddOneHzAdjustmentCmd CFE_TIME_AddOneHzAdjustmentCmd_t`

**12.206.2.4 CFE\_TIME\_DiagnosticTlm\_t** `typedef struct CFE_TIME_DiagnosticTlm CFE_TIME_DiagnosticTlm_t`

**12.206.2.5 CFE\_TIME\_FakeToneCmd\_t** `typedef struct CFE_TIME_FakeToneCmd CFE_TIME_FakeToneCmd_t`

**12.206.2.6 CFE\_TIME\_HousekeepingTlm\_t** `typedef struct CFE_TIME_HousekeepingTlm CFE_TIME_HousekeepingTlm_t`

**12.206.2.7 CFE\_TIME\_NoopCmd\_t** `typedef struct CFE_TIME_NoopCmd CFE_TIME_NoopCmd_t`

**12.206.2.8 CFE\_TIME\_OneHzCmd\_t** `typedef struct CFE_TIME_OneHzCmd CFE_TIME_OneHzCmd_t`

**12.206.2.9 CFE\_TIME\_ResetCountersCmd\_t** `typedef struct CFE_TIME_ResetCountersCmd CFE_TIME_ResetCountersCmd_t`

**12.206.2.10 CFE\_TIME\_SendDiagnosticCmd\_t** `typedef struct CFE_TIME_SendDiagnosticCmd CFE_TIME_SendDiagnosticCmd_t`

**12.206.2.11 CFE\_TIME\_SendHkCmd\_t** `typedef struct CFE_TIME_SendHkCmd CFE_TIME_SendHkCmd_t`

**12.206.2.12 CFE\_TIME\_SetLeapSecondsCmd\_t** `typedef struct CFE_TIME_SetLeapSecondsCmd CFE_TIME_SetLeapSecondsCmd_t`  
Set leap seconds command.

**12.206.2.13 CFE\_TIME\_SetMETCmd\_t** `typedef struct CFE_TIME_SetMETCmd CFE_TIME_SetMETCmd_t`

**12.206.2.14 CFE\_TIME\_SetSignalCmd\_t** `typedef struct CFE_TIME_SetSignalCmd CFE_TIME_SetSignalCmd_t`  
Set tone signal source command.

**12.206.2.15 CFE\_TIME\_SetSourceCmd\_t** `typedef struct CFE_TIME_SetSourceCmd CFE_TIME_SetSourceCmd_t`  
Set time data source command.

**12.206.2.16 CFE\_TIME\_SetStateCmd\_t** `typedef struct CFE_TIME_SetStateCmd CFE_TIME_SetStateCmd_t`  
Set clock state command.

**12.206.2.17 CFE\_TIME\_SetSTCFCmd\_t** `typedef struct CFE_TIME_SetSTCFCmd CFE_TIME_SetSTCFCmd_t`

**12.206.2.18 CFE\_TIME\_SetTimeCmd\_t** `typedef struct CFE_TIME_SetTimeCmd CFE_TIME_SetTimeCmd_t`

**12.206.2.19 CFE\_TIME\_SubAdjustCmd\_t** `typedef struct CFE_TIME_SubAdjustCmd CFE_TIME_SubAdjustCmd_t`

**12.206.2.20 CFE\_TIME\_SubDelayCmd\_t** `typedef struct CFE_TIME_SubDelayCmd CFE_TIME_SubDelayCmd_t`

**12.206.2.21 CFE\_TIME\_SubOneHzAdjustmentCmd\_t** `typedef struct CFE_TIME_SubOneHzAdjustmentCmd CFE_TIME_SubOneHzAdjustmentCmd_t`

**12.206.2.22 CFE\_TIME\_ToneDataCmd\_t** `typedef struct CFE_TIME_ToneDataCmd CFE_TIME_ToneDataCmd_t`  
Time at tone data command.

**12.206.2.23 CFE\_TIME\_ToneSignalCmd\_t** `typedef struct CFE_TIME_ToneSignalCmd CFE_TIME_ToneSignalCmd_t`

## 12.207 cfe/modules/time/config/default\_cfe\_time\_platform\_cfg.h File Reference

```
#include "cfe_time_mission_cfg.h"
#include "cfe_time_internal_cfg.h"
```

### 12.207.1 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.

## 12.208 cfe/modules/time/config/default\_cfe\_time\_topicid\_values.h File Reference

### Macros

- `#define CFE_MISSION_TIME_TIDVAL(x) DEFAULT_CFE_MISSION_TIME_##x##_TOPICID`

### 12.208.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Topic IDs

### 12.208.2 Macro Definition Documentation

**12.208.2.1 CFE\_MISSION\_TIME\_TIDVAL** #define CFE\_MISSION\_TIME\_TIDVAL(

x ) DEFAULT\_CFE\_MISSION\_TIME\_##x##\_TOPICID

Definition at line 26 of file default\_cfe\_time\_topicid\_values.h.

**12.209 cfe/modules/time/fsw/inc/cfe\_time\_eventids.h File Reference****Macros****TIME event IDs**

- #define **CFE\_TIME\_INIT\_EID** 1  
*TIME Initialization Event ID.*
- #define **CFE\_TIME\_NOOP\_EID** 4  
*TIME No-op Command Success Event ID.*
- #define **CFE\_TIME\_RESET\_EID** 5  
*TIME Reset Counters Command Success Event ID.*
- #define **CFE\_TIME\_DIAG\_EID** 6  
*TIME Request Diagnostics Command Success Event ID.*
- #define **CFE\_TIME\_STATE\_EID** 7  
*TIME Set Time State Command Success Event ID.*
- #define **CFE\_TIME\_SOURCE\_EID** 8  
*TIME Set Time Source Command Success Event ID.*
- #define **CFE\_TIME\_SIGNAL\_EID** 9  
*TIME Set Tone Source Command Success Event ID.*
- #define **CFE\_TIME\_DELAY\_EID** 11  
*TIME Add or Subtract Delay Command Success Event ID.*
- #define **CFE\_TIME\_TIME\_EID** 12  
*TIME Set Time Command Success Event ID.*
- #define **CFE\_TIME\_MET\_EID** 13  
*TIME Set Mission Elapsed Time Command Success Event ID.*
- #define **CFE\_TIME\_STCF\_EID** 14  
*TIME Set Spacecraft Time Correlation Factor Command Success Event ID.*
- #define **CFE\_TIME\_DELTA\_EID** 15  
*TIME Add or Subtract Single STCF Adjustment Command Success Event ID.*
- #define **CFE\_TIME\_ONEHZ\_EID** 16  
*TIME Add or Subtract STCF Adjustment Each Second Command Success Event ID.*
- #define **CFE\_TIME\_LEAPS\_EID** 17  
*TIME Set Leap Seconds Command Success Event ID.*
- #define **CFE\_TIME\_FLY\_ON\_EID** 20  
*TIME Entered FLYWHEEL Mode Event ID.*
- #define **CFE\_TIME\_FLY\_OFF\_EID** 21  
*TIME Exited FLYWHEEL Mode Event ID.*
- #define **CFE\_TIME\_ID\_ERR\_EID** 26  
*TIME Invalid Message ID Received Event ID.*
- #define **CFE\_TIME\_CC\_ERR\_EID** 27  
*TIME Invalid Command Code Received Event ID.*
- #define **CFE\_TIME\_STATE\_ERR\_EID** 30  
*TIME Set Clock State Command Invalid State Event ID.*
- #define **CFE\_TIME\_SOURCE\_ERR\_EID** 31  
*TIME Set Clock Source Command Invalid Source Event ID.*
- #define **CFE\_TIME\_SIGNAL\_ERR\_EID** 32  
*TIME Set Clock Tone Source Command Invalid Source Event ID.*
- #define **CFE\_TIME\_DELAY\_ERR\_EID** 33  
*TIME Add or Subtract Tone Delay Command Invalid Time Value Event ID.*
- #define **CFE\_TIME\_TIME\_ERR\_EID** 34

- `#define CFE_TIME_MET_ERR_EID 35`  
*TIME Set Mission Elapsed Time Command Invalid Time Value Event ID.*
- `#define CFE_TIME_STCF_ERR_EID 36`  
*TIME Set Spacecraft Time Correlation Factor Command Invalid Time Value Event ID.*
- `#define CFE_TIME_DELTA_ERR_EID 37`  
*TIME Add or Subtract Single STCF Adjustment Command Invalid Time Value Event ID.*
- `#define CFE_TIME_SOURCE_CFG_EID 40`  
*TIME Set Clock Source Command Incompatible Mode Event ID.*
- `#define CFE_TIME_SIGNAL_CFG_EID 41`  
*TIME Set Clock Signal Command Incompatible Mode Event ID.*
- `#define CFE_TIME_DELAY_CFG_EID 42`  
*TIME Add or Subtract Tone Delay Command Incompatible Mode Event ID.*
- `#define CFE_TIME_TIME_CFG_EID 43`  
*TIME Set Spacecraft Time Command Incompatible Mode Event ID.*
- `#define CFE_TIME_MET_CFG_EID 44`  
*TIME Set Mission Elapsed Time Command Incompatible Mode Event ID.*
- `#define CFE_TIME_STCF_CFG_EID 45`  
*TIME Set Spacecraft Time Correlation Factor Command Incompatible Mode Event ID.*
- `#define CFE_TIME_LEAPS_CFG_EID 46`  
*TIME Set Leap Seconds Command Incompatible Mode Event ID.*
- `#define CFE_TIME_DELTA_CFG_EID 47`  
*TIME Add or Subtract Single STCF Adjustment Command Incompatible Mode Event ID.*
- `#define CFE_TIME_ONEHZ_CFG_EID 48`  
*TIME Add or Subtract STCF Adjustment Each Second Command Incompatible Mode Event ID.*
- `#define CFE_TIME_LEN_ERR_EID 49`  
*TIME Invalid Command Length Event ID.*

### 12.209.1 Detailed Description

cFE Time Services Event IDs

### 12.209.2 Macro Definition Documentation

**12.209.2.1 CFE\_TIME\_CC\_ERR\_EID** `#define CFE_TIME_CC_ERR_EID 27`  
TIME Invalid Command Code Received Event ID.

Type: ERROR

Cause:

Invalid command code for message ID `CFE_TIME_CMD_MID` received on the TIME message pipe.  
Definition at line 232 of file `cfe_time_eventids.h`.

**12.209.2.2 CFE\_TIME\_DELAY\_CFG\_EID** `#define CFE_TIME_DELAY_CFG_EID 42`  
TIME Add or Subtract Tone Delay Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Add Tone Delay Command](#) OR [TIME Subtract Tone Delay Command](#) failure due to being in an incompatible mode.

Definition at line 364 of file cfe\_time\_eventids.h.

**12.209.2.3 CFE\_TIME\_DELAY\_EID** #define CFE\_TIME\_DELAY\_EID 11  
TIME Add or Subtract Delay Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Add Time Delay Command](#) OR a [Subtract Time Delay Command](#) success.

Definition at line 120 of file cfe\_time\_eventids.h.

**12.209.2.4 CFE\_TIME\_DELAY\_ERR\_EID** #define CFE\_TIME\_DELAY\_ERR\_EID 33  
TIME Add or Subtract Tone Delay Command Invalid Time Value Event ID.

Type: ERROR

Cause:

[TIME Add Tone Delay Command](#) OR [TIME Subtract Tone Delay Command](#) failure due to an invalid time value.

Definition at line 278 of file cfe\_time\_eventids.h.

**12.209.2.5 CFE\_TIME\_DELTA\_CFG\_EID** #define CFE\_TIME\_DELTA\_CFG\_EID 47  
TIME Add or Subtract Single STCF Adjustment Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Add Single STCF Adjustment Command](#) OR [TIME Subtract Single STCF Adjustment Command](#) failure due to being in an incompatible mode.

Definition at line 425 of file cfe\_time\_eventids.h.

**12.209.2.6 CFE\_TIME\_DELTA\_EID** #define CFE\_TIME\_DELTA\_EID 15  
TIME Add or Subtract Single STCF Adjustment Command Success Event ID.

Type: INFORMATION

Cause:

TIME Add Single STCF Adjustment Command OR TIME Subtract Single STCF Adjustment Command success.  
Definition at line 165 of file cfe\_time\_eventids.h.

**12.209.2.7 CFE\_TIME\_DELTA\_ERR\_EID** #define CFE\_TIME\_DELTA\_ERR\_EID 37  
TIME Add or Subtract Single STCF Adjustment Command Invalid Time Value Event ID.

Type: ERROR

Cause:

TIME Add Single STCF Adjustment Command OR TIME Subtract Single STCF Adjustment Command failure due to  
an invalid time value.  
Definition at line 327 of file cfe\_time\_eventids.h.

**12.209.2.8 CFE\_TIME\_DIAG\_EID** #define CFE\_TIME\_DIAG\_EID 6  
TIME Request Diagnostics Command Success Event ID.

Type: DEBUG

Cause:

TIME Request Diagnostics Command success.  
Definition at line 75 of file cfe\_time\_eventids.h.

**12.209.2.9 CFE\_TIME\_FLY\_OFF\_EID** #define CFE\_TIME\_FLY\_OFF\_EID 21  
TIME Exited FLYWHEEL Mode Event ID.

Type: INFORMATION

Cause:

TIME Exited FLYWHEEL Mode.  
Definition at line 210 of file cfe\_time\_eventids.h.

**12.209.2.10 CFE\_TIME\_FLY\_ON\_EID** #define CFE\_TIME\_FLY\_ON\_EID 20  
TIME Entered FLYWHEEL Mode Event ID.

Type: INFORMATION

Cause:

TIME Entered FLYWHEEL Mode.  
Definition at line 199 of file cfe\_time\_eventids.h.

**12.209.2.11 CFE\_TIME\_ID\_ERR\_EID** #define CFE\_TIME\_ID\_ERR\_EID 26  
TIME Invalid Message ID Received Event ID.

Type: ERROR

Cause:

Invalid message ID received on the TIME message pipe.  
Definition at line 221 of file cfe\_time\_eventids.h.

**12.209.2.12 CFE\_TIME\_INIT\_EID** #define CFE\_TIME\_INIT\_EID 1  
TIME Initialization Event ID.

Type: INFORMATION

Cause:

Time Services Task Initialization complete.  
Definition at line 42 of file cfe\_time\_eventids.h.

**12.209.2.13 CFE\_TIME\_LEAPS\_CFG\_EID** #define CFE\_TIME\_LEAPS\_CFG\_EID 46  
TIME Set Leap Seconds Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Leap Seconds Command](#) failure due to being in an incompatible mode.  
Definition at line 412 of file cfe\_time\_eventids.h.

**12.209.2.14 CFE\_TIME\_LEAPS\_EID** #define CFE\_TIME\_LEAPS\_EID 17  
TIME Set Leap Seconds Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Leap Seconds Command](#) success.  
Definition at line 188 of file cfe\_time\_eventids.h.

**12.209.2.15 CFE\_TIME\_LEN\_ERR\_EID** #define CFE\_TIME\_LEN\_ERR\_EID 49  
TIME Invalid Command Length Event ID.

Type: ERROR

Cause:

Invalid length for the command code in message ID [CFE\\_TIME\\_CMD\\_MID](#) received on the TIME message pipe.  
Definition at line 450 of file cfe\_time\_eventids.h.

**12.209.2.16 CFE\_TIME\_MET\_CFG\_EID** #define CFE\_TIME\_MET\_CFG\_EID 44  
TIME Set Mission Elapsed Time Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Set Mission Elapsed Time Command](#) failure due to being in an incompatible mode.  
Definition at line 388 of file cfe\_time\_eventids.h.

**12.209.2.17 CFE\_TIME\_MET\_EID** #define CFE\_TIME\_MET\_EID 13  
TIME Set Mission Elapsed Time Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Set Mission Elapsed Time Command](#) success.  
Definition at line 142 of file cfe\_time\_eventids.h.

**12.209.2.18 CFE\_TIME\_MET\_ERR\_EID** #define CFE\_TIME\_MET\_ERR\_EID 35  
TIME Set Mission Elapsed Time Command Invalid Time Value Event ID.

Type: ERROR

Cause:

[TIME Set Mission Elapsed Time Command](#) failure due to an invalid time value.  
Definition at line 302 of file cfe\_time\_eventids.h.

**12.209.2.19 CFE\_TIME\_NOOP\_EID** #define CFE\_TIME\_NOOP\_EID 4  
TIME No-op Command Success Event ID.

Type: INFORMATION

Cause:

[TIME NO-OP Command](#) success.  
Definition at line 53 of file cfe\_time\_eventids.h.

**12.209.2.20 CFE\_TIME\_ONEHZ\_CFG\_EID** #define CFE\_TIME\_ONEHZ\_CFG\_EID 48  
TIME Add or Subtract STCF Adjustment Each Second Command Incompatible Mode Event ID.

Type: ERROR

Cause:

[TIME Add STCF Adjustment Each Second Command](#) OR [TIME Subtract STCF Adjustment Each Second Command](#) failure due to being in an incompatible mode.  
Definition at line 438 of file cfe\_time\_eventids.h.

**12.209.2.21 CFE\_TIME\_ONEHZ\_EID** #define CFE\_TIME\_ONEHZ\_EID 16  
TIME Add or Subtract STCF Adjustment Each Second Command Success Event ID.

Type: INFORMATION

Cause:

[TIME Add STCF Adjustment Each Second Command](#) OR [TIME Subtract STCF Adjustment Each Second Command](#) success.  
Definition at line 177 of file cfe\_time\_eventids.h.

**12.209.2.22 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.

**12.209.2.23 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.

**12.209.2.24 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.

**12.209.2.25 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.

**12.209.2.26 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.

**12.209.2.27 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.

**12.209.2.28 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.

**12.209.2.29 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.

**12.209.2.30 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.

**12.209.2.31 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.

**12.209.2.32 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.

**12.209.2.33 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.

**12.209.2.34 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.

**12.209.2.35 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.

**12.209.2.36 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.

## 12.210 cfe/modules/time/fsw/inc/cfe\_time\_fcncodes.h File Reference

```
#include "cfe_time_fcncode_values.h"
```

### Macros

#### Time Services Command Codes

- #define CFE\_TIME\_NOOP\_CC CFE\_TIME\_CCVAL(NOP)
- #define CFE\_TIME\_RESET\_COUNTERS\_CC CFE\_TIME\_CCVAL(RESET\_COUNTERS)
- #define CFE\_TIME\_SEND\_DIAGNOSTIC\_CC CFE\_TIME\_CCVAL(SEND\_DIAGNOSTIC)
- #define CFE\_TIME\_SET\_SOURCE\_CC CFE\_TIME\_CCVAL(SET\_SOURCE)
- #define CFE\_TIME\_SET\_STATE\_CC CFE\_TIME\_CCVAL(SET\_STATE)
- #define CFE\_TIME\_ADD\_DELAY\_CC CFE\_TIME\_CCVAL(ADD\_DELAY)

- #define CFE\_TIME\_SUB\_DELAY\_CC CFE\_TIME\_CCVAL(SUB\_DELAY)
- #define CFE\_TIME\_SET\_TIME\_CC CFE\_TIME\_CCVAL(SET\_TIME)
- #define CFE\_TIME\_SET\_MET\_CC CFE\_TIME\_CCVAL(SET\_MET)
- #define CFE\_TIME\_SET\_STCF\_CC CFE\_TIME\_CCVAL(SET\_STCF)
- #define CFE\_TIME\_SET\_LEAP\_SECONDS\_CC CFE\_TIME\_CCVAL(SET\_LEAP\_SECONDS)
- #define CFE\_TIME\_ADD\_ADJUST\_CC CFE\_TIME\_CCVAL(ADD\_ADJUST)
- #define CFE\_TIME\_SUB\_ADJUST\_CC CFE\_TIME\_CCVAL(SUB\_ADJUST)
- #define CFE\_TIME\_ADD\_ONE\_HZ\_ADJUSTMENT\_CC CFE\_TIME\_CCVAL(ADD\_ONE\_HZ\_ADJUSTMENT)
- #define CFE\_TIME\_SUB\_ONE\_HZ\_ADJUSTMENT\_CC CFE\_TIME\_CCVAL(SUB\_ONE\_HZ\_ADJUSTMENT)
- #define CFE\_TIME\_SET\_SIGNAL\_CC CFE\_TIME\_CCVAL(SET\_SIGNAL)

### 12.210.1 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.

### 12.210.2 Macro Definition Documentation

#### 12.210.2.1 CFE\_TIME\_ADD\_ADJUST\_CC #define CFE\_TIME\_ADD\_ADJUST\_CC CFE\_TIME\_CCVAL(ADD\_ADJUST)

**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\_STCF - Housekeeping Telemetry point indicating new STCF value
- The CFE\_TIME\_DELTA\_EID informational event message will be generated

#### Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- \$sc\_\$cpu\_TIME\_CMDEC - command error counter will increment
- Error specific event messages will be issued (CFE\_TIME\_DELTA\_ERR\_EID or CFE\_TIME\_DELTA\_CFG\_EID)

### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

### See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#), [CFE\\_TIME\\_ADD\\_ONE\\_HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SUB\\_ONE\\_HZ\\_ADJUSTMENT\\_CC](#)

Definition at line 519 of file cfe\_time\_fcncodes.h.

### 12.210.2.2 CFE\_TIME\_ADD\_DELAY\_CC #define CFE\_TIME\_ADD\_DELAY\_CC CFE\_TIME\_CCVAL(ADD\_DELAY)

**Name** Add Time to Tone Time Delay

#### Description

This command is used to factor out a known, predictable latency between the Time Server and a particular Time Client. The correction is applied (added) to the current time calculation for Time Clients, so this command has no meaning for Time Servers. Each Time Client can have a unique latency setting. The latency value is a positive number of seconds and microseconds that represent the deviation from the time maintained by the Time Server.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_AddClockLat

#### Command Structure

[CFE\\_TIME\\_AddDelayCmd\\_t](#)

#### Command Verification

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

- **\$sc\_\$cpu\_TIME\_CMDPC** - command execution counter will increment
- **\$sc\_\$cpu\_TIME\_DLAtentS**, **\$sc\_\$cpu\_TIME\_DLAtentSs** - Housekeeping Telemetry point indicating command specified values
- **\$sc\_\$cpu\_TIME\_DLAtentDir** - Diagnostic Telemetry point indicating commanded latency direction
- The [CFE\\_TIME\\_DELAY\\_EID](#) informational event message will be generated

#### Error Conditions

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Client

Evidence of Failure may be found in the following telemetry:

- **\$sc\_\$cpu\_TIME\_CMDEC** - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_DELAY\\_CFG\\_EID](#) or [CFE\\_TIME\\_DELAY\\_ERR\\_EID](#))

### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

### See also

[CFE\\_TIME\\_SUB\\_DELAY\\_CC](#)

Definition at line 292 of file cfe\_time\_fcncodes.h.

**12.210.2.3 CFE\_TIME\_ADD\_ONE\_HZ\_ADJUSTMENT\_CC** #define CFE\_TIME\_ADD\_ONE\_HZ\_ADJUSTMENT\_←  
CC CFE\_TIME\_CCVAL(ADD\_ONE\_HZ\_ADJUSTMENT)

**Name** Add Delta to Spacecraft Time Correlation Factor each 1Hz

#### Description

This command has been updated to take actual sub-seconds ( $1/2^{32}$  seconds) rather than micro-seconds as an input argument. This change occurred after the determination was made that one micro-second is too large an increment for a constant 1Hz adjustment.

This command continuously adjusts the Spacecraft Time Correlation Factor (STCF) every second, by adding the specified value. The adjustment to the STCF is applied in the Time Service local 1Hz interrupt handler. As the local 1Hz interrupt is not synchronized to the tone signal, one cannot say when the adjustment will occur, other than once a second, at about the same time relative to the tone.

There was some debate about whether the maximum 1Hz clock drift correction factor would ever need to exceed some small fraction of a second. But, the decision was made to provide the capability to make 1Hz adjustments greater than one second and leave it to the ground system to provide mission specific limits.

**Command Mnemonic(s)** \$sc\_\$cpu\_TIME\_Add1HzSTCF

#### Command Structure

[CFE\\_TIME\\_AddOneHzAdjustmentCmd\\_t](#)

#### Command Verification

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

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_STCF](#) - Housekeeping Telemetry point indicating new STCF value
- The [CFE\\_TIME\\_ONEHZ\\_EID](#) informational event message will be generated

#### Error Conditions

- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will increment
- Error specific event message will be issued ([CFE\\_TIME\\_ONEHZ\\_CFG\\_EID](#))

#### Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

#### See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ONE\\_HZ\\_ADJUSTMENT\\_CC](#)

Definition at line 597 of file `cfe_time_fcncodes.h`.

**12.210.2.4 CFE\_TIME\_NOOP\_CC** #define CFE\_TIME\_NOOP\_CC CFE\_TIME\_CCVAL(NOOP)**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 68 of file cfe\_time\_fcncodes.h.

**12.210.2.5 CFE\_TIME\_RESET\_COUNTERS\_CC** #define CFE\_TIME\_RESET\_COUNTERS\_CC CFE\_TIME\_CCVAL(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 113 of file [cfe\\_time\\_fcncodes.h](#).

**12.210.2.6 CFE\_TIME\_SEND\_DIAGNOSTIC\_CC** #define CFE\_TIME\_SEND\_DIAGNOSTIC\_CC [CFE\\_TIME\\_CCVAL](#) (SEND←\_DIAGNOSTIC)

**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 147 of file cfe\_time\_fcncodes.h.

**12.210.2.7 CFE\_TIME\_SET\_LEAP\_SECONDS\_CC** #define CFE\_TIME\_SET\_LEAP\_SECONDS\_CC CFE\_TIME\_CCVAL (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 484 of file `cfe_time_fcncodes.h`.

#### 12.210.2.8 CFE\_TIME\_SET\_MET\_CC `#define CFE_TIME_SET_MET_CC CFE_TIME_CCVAL(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_MET` - Housekeeping Telemetry point indicating new MET 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 cfe\_time\_fcncodes.h.

### **12.210.2.9 CFE\_TIME\_SET\_SIGNAL\_CC** #define CFE\_TIME\_SET\_SIGNAL\_CC CFE\_TIME\_CCVAL(SET\_SIGNAL)

**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 686 of file cfe\_time\_fcncodes.h.

**12.210.2.10 CFE\_TIME\_SET\_SOURCE\_CC** #define CFE\_TIME\_SET\_SOURCE\_CC CFE\_TIME\_CCVAL(SET\_SOURCE)**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 197 of file `cfe_time_fcncodes.h`.

**12.210.2.11 CFE\_TIME\_SET\_STATE\_CC** #define CFE\_TIME\_SET\_STATE\_CC CFE\_TIME\_CCVAL(SET\_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 VALID (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 254 of file cfe\_time\_fcncodes.h.

#### 12.210.2.12 CFE\_TIME\_SET\_STCF\_CC #define CFE\_TIME\_SET\_STCF\_CC CFE\_TIME\_CCVAL(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\\_STCF](#) - Housekeeping Telemetry point indicating new STCF 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 449 of file cfe\_time\_fcncodes.h.

**12.210.2.13 CFE\_TIME\_SET\_TIME\_CC** #define CFE\_TIME\_SET\_TIME\_CC CFE\_TIME\_CCVAL(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\_STCF** - Housekeeping Telemetry point indicating newly calculated STCF 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 374 of file cfe\_time\_fcncodes.h.

**12.210.2.14 CFE\_TIME\_SUB\_ADJUST\_CC** #define CFE\_TIME\_SUB\_ADJUST\_CC CFE\_TIME\_CCVAL(SUB\_ADJUST)**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\\_STCF](#) - Housekeeping Telemetry point indicating new STCF value
- The [CFE\\_TIME\\_DELTA\\_EID](#) informational event message will be generated

**Error Conditions**

- An invalid number of microseconds was specified (must be less than 1 million)
- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will increment
- Error specific event messages will be issued ([CFE\\_TIME\\_DELTA\\_ERR\\_EID](#) or [CFE\\_TIME\\_DELTA\\_CFG\\_EID](#))

**Criticality**

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

**See also**[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_ADD\\_ONE\\_HZ\\_ADJUSTMENT\\_CC](#), [CFE\\_TIME\\_SUB\\_ONE\\_HZ\\_ADJUSTMENT\\_CC](#)

Definition at line 552 of file `cfe_time_fcncodes.h`.

**12.210.2.15 CFE\_TIME\_SUB\_DELAY\_CC** #define CFE\_TIME\_SUB\_DELAY\_CC CFE\_TIME\_CCVAL(SUB\_DELAY)**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 330 of file `cfe_time_fcncodes.h`.

**12.210.2.16 CFE\_TIME\_SUB\_ONE\_HZ\_ADJUSTMENT\_CC** #define CFE\_TIME\_SUB\_ONE\_HZ\_ADJUSTMENT\_←  
CC CFE\_TIME\_CCVAL(SUB\_ONE\_HZ\_ADJUSTMENT)

**Name** Subtract Delta from Spacecraft Time Correlation Factor each 1Hz

## Description

This command has been updated to take actual sub-seconds ( $1/2^{32}$  seconds) rather than micro-seconds as an input argument. This change occurred after the determination was made that one micro-second is too large an increment for a constant 1Hz adjustment.

This command continuously adjusts the Spacecraft Time Correlation Factor (STCF) every second, by subtracting the specified value. The adjustment to the STCF is applied in the Time Service local 1Hz interrupt handler. As the local 1Hz interrupt is not synchronized to the tone signal, one cannot say when the adjustment will occur, other than once a second, at about the same time relative to the tone.

There was some debate about whether the maximum 1Hz clock drift correction factor would ever need to exceed some small fraction of a second. But, the decision was made to provide the capability to make 1Hz adjustments greater than one second and leave it to the ground system to provide mission specific limits.

## Command Mnemonic(s)

\$sc\_\$cpu\_TIME\_Sub1HzSTCF

## Command Structure

[CFE\\_TIME\\_SubOneHzAdjustmentCmd\\_t](#)

## Command Verification

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

- [\\$sc\\_\\$cpu\\_TIME\\_CMDPC](#) - command execution counter will increment
- [\\$sc\\_\\$cpu\\_TIME\\_STCF](#) - Housekeeping Telemetry point indicating new STCF value
- The [CFE\\_TIME\\_ONEHZ\\_EID](#) informational event message will be generated

## Error Conditions

- Platform receiving the command is not a Time Server

Evidence of Failure may be found in the following telemetry:

- [\\$sc\\_\\$cpu\\_TIME\\_CMDEC](#) - command error counter will increment
- Error specific event message will be issued ([CFE\\_TIME\\_ONEHZ\\_CFG\\_EID](#))

## Criticality

Inappropriately setting the clock may result in other sub-systems performing incorrect time based calculations. The specific risk is dependent upon the behavior of those sub-systems.

## See also

[CFE\\_TIME\\_ADD\\_ADJUST\\_CC](#), [CFE\\_TIME\\_SUB\\_ADJUST\\_CC](#), [CFE\\_TIME\\_ADD\\_ONE\\_HZ\\_ADJUSTMENT\\_CC](#)

Definition at line 644 of file `cfe_time_fcncodes.h`.

## 12.211 cfe/modules/time/fsw/inc/cfe\_time\_interface\_cfg.h File Reference

```
#include "cfe_time_interface_cfg_values.h"
```

**Macros**

- #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI CFE\_MISSION\_TIME\_CFGVAL(CFG\_DEFAULT\_TAI)
- #define DEFAULT\_CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI true
- #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC CFE\_MISSION\_TIME\_CFGVAL(CFG\_DEFAULT\_UTC)
- #define DEFAULT\_CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC false
- #define CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE CFE\_MISSION\_TIME\_CFGVAL(CFG\_FAKE\_TONE)
- #define DEFAULT\_CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE true
- #define CFE\_MISSION\_TIME\_AT\_TONE\_WAS CFE\_MISSION\_TIME\_CFGVAL(AT\_TONE\_WAS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_AT\_TONE\_WAS true
- #define CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE CFE\_MISSION\_TIME\_CFGVAL(AT\_TONE\_WILL\_BE)
- #define DEFAULT\_CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE false
- #define CFE\_MISSION\_TIME\_MIN\_ELAPSED CFE\_MISSION\_TIME\_CFGVAL(MIN\_ELAPSED)
- #define DEFAULT\_CFE\_MISSION\_TIME\_MIN\_ELAPSED 0
- #define CFE\_MISSION\_TIME\_MAX\_ELAPSED CFE\_MISSION\_TIME\_CFGVAL(MAX\_ELAPSED)
- #define DEFAULT\_CFE\_MISSION\_TIME\_MAX\_ELAPSED 200000
- #define CFE\_MISSION\_TIME\_DEF\_MET\_SECS CFE\_MISSION\_TIME\_CFGVAL(DEF\_MET\_SECS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_MET\_SECS 1000
- #define CFE\_MISSION\_TIME\_DEF\_MET\_SUBS CFE\_MISSION\_TIME\_CFGVAL(DEF\_MET\_SUBS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_MET\_SUBS 0
- #define CFE\_MISSION\_TIME\_DEF\_STCF\_SECS CFE\_MISSION\_TIME\_CFGVAL(DEF\_STCF\_SECS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_STCF\_SECS 1000000
- #define CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS CFE\_MISSION\_TIME\_CFGVAL(DEF\_STCF\_SUBS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS 0
- #define CFE\_MISSION\_TIME\_DEF\_LEAPS CFE\_MISSION\_TIME\_CFGVAL(DEF\_LEAPS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_LEAPS 37
- #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS CFE\_MISSION\_TIME\_CFGVAL(DEF\_DELAY\_SECS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS 0
- #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS CFE\_MISSION\_TIME\_CFGVAL(DEF\_DELAY\_SUBS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS 1000
- #define CFE\_MISSION\_TIME\_EPOCH\_YEAR CFE\_MISSION\_TIME\_CFGVAL(EPOCH\_YEAR)
- #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_YEAR 1980
- #define CFE\_MISSION\_TIME\_EPOCH\_DAY CFE\_MISSION\_TIME\_CFGVAL(EPOCH\_DAY)
- #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_DAY 1
- #define CFE\_MISSION\_TIME\_EPOCH\_HOUR CFE\_MISSION\_TIME\_CFGVAL(EPOCH\_HOUR)
- #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_HOUR 0
- #define CFE\_MISSION\_TIME\_EPOCH\_MINUTE CFE\_MISSION\_TIME\_CFGVAL(EPOCH\_MINUTE)
- #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_MINUTE 0
- #define CFE\_MISSION\_TIME\_EPOCH\_SECOND CFE\_MISSION\_TIME\_CFGVAL(EPOCH\_SECOND)
- #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_SECOND 0
- #define CFE\_MISSION\_TIME\_EPOCH\_MICROS CFE\_MISSION\_TIME\_CFGVAL(EPOCH\_MICROS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_MICROS 0
- #define CFE\_MISSION\_TIME\_EPOCH\_SECONDS CFE\_MISSION\_TIME\_CFGVAL(EPOCH\_SECONDS)
- #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_SECONDS 315532800
- #define CFE\_MISSION\_TIME\_FS\_FACTOR CFE\_MISSION\_TIME\_CFGVAL(FS\_FACTOR)
- #define DEFAULT\_CFE\_MISSION\_TIME\_FS\_FACTOR 789004800

### 12.211.1 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.

### 12.211.2 Macro Definition Documentation

#### 12.211.2.1 CFE\_MISSION\_TIME\_AT\_TONE\_WAS `#define CFE_MISSION_TIME_AT_TONE_WAS CFE_MISSION_TIME_CFGVAL(AT←_TONE_WAS)`

**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 93 of file cfe\_time\_interface\_cfg.h.

#### 12.211.2.2 CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE `#define CFE_MISSION_TIME_AT_TONE_WILL_BE CFE_MISSION_TIME_CFGVAL(_TONE_WILL_BE)`

Definition at line 96 of file cfe\_time\_interface\_cfg.h.

#### 12.211.2.3 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI `#define CFE_MISSION_TIME_CFG_DEFAULT_TAI CFE_MISSION_TIME_CFGVAL(_DEFAULT_TAI)`

**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 53 of file cfe\_time\_interface\_cfg.h.

**12.211.2.4 CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC** #define CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC CFE\_MISSION\_TIME\_CFGVAL(\_DEFAULT\_UTC)

Definition at line 56 of file cfe\_time\_interface\_cfg.h.

**12.211.2.5 CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE** #define CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE CFE\_MISSION\_TIME\_CFGVAL(\_FAKE\_TONE)

**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 69 of file cfe\_time\_interface\_cfg.h.

**12.211.2.6 CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS CFE\_MISSION\_TIME\_CFGVAL(\_DELAY\_SECS)

Definition at line 165 of file cfe\_time\_interface\_cfg.h.

**12.211.2.7 CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS CFE\_MISSION\_TIME\_CFGVAL(\_DELAY\_SUBS)

Definition at line 168 of file cfe\_time\_interface\_cfg.h.

**12.211.2.8 CFE\_MISSION\_TIME\_DEF\_LEAPS** #define CFE\_MISSION\_TIME\_DEF\_LEAPS CFE\_MISSION\_TIME\_CFGVAL(DEF←\_LEAPS)

Definition at line 162 of file cfe\_time\_interface\_cfg.h.

**12.211.2.9 CFE\_MISSION\_TIME\_DEF\_MET\_SECS** #define CFE\_MISSION\_TIME\_DEF\_MET\_SECS CFE\_MISSION\_TIME\_CFGVAL(DEF←\_MET\_SECS)

**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 150 of file cfe\_time\_interface\_cfg.h.

**12.211.2.10 CFE\_MISSION\_TIME\_DEF\_MET\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_MET\_SUBS CFE\_MISSION\_TIME\_CFGVAL(DEF\_M<sub>ET</sub>\_SUBS)

Definition at line 153 of file cfe\_time\_interface\_cfg.h.

**12.211.2.11 CFE\_MISSION\_TIME\_DEF\_STCF\_SECS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SECS CFE\_MISSION\_TIME\_CFGVAL(DE<sub>STCF</sub>\_SECS)

Definition at line 156 of file cfe\_time\_interface\_cfg.h.

**12.211.2.12 CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS** #define CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS CFE\_MISSION\_TIME\_CFGVAL(DE<sub>STCF</sub>\_SUBS)

Definition at line 159 of file cfe\_time\_interface\_cfg.h.

**12.211.2.13 CFE\_MISSION\_TIME\_EPOCH\_DAY** #define CFE\_MISSION\_TIME\_EPOCH\_DAY CFE\_MISSION\_TIME\_CFGVAL(EPOCH<sub>DAY</sub>)

Definition at line 189 of file cfe\_time\_interface\_cfg.h.

**12.211.2.14 CFE\_MISSION\_TIME\_EPOCH\_HOUR** #define CFE\_MISSION\_TIME\_EPOCH\_HOUR CFE\_MISSION\_TIME\_CFGVAL(EPOCH<sub>HOUR</sub>)

Definition at line 192 of file cfe\_time\_interface\_cfg.h.

**12.211.2.15 CFE\_MISSION\_TIME\_EPOCH\_MICROS** #define CFE\_MISSION\_TIME\_EPOCH\_MICROS CFE\_MISSION\_TIME\_CFGVAL(EPO<sub>MICROS</sub>)

Definition at line 201 of file cfe\_time\_interface\_cfg.h.

**12.211.2.16 CFE\_MISSION\_TIME\_EPOCH\_MINUTE** #define CFE\_MISSION\_TIME\_EPOCH\_MINUTE CFE\_MISSION\_TIME\_CFGVAL(EPO<sub>MINUTE</sub>)

Definition at line 195 of file cfe\_time\_interface\_cfg.h.

**12.211.2.17 CFE\_MISSION\_TIME\_EPOCH\_SECOND** #define CFE\_MISSION\_TIME\_EPOCH\_SECOND CFE\_MISSION\_TIME\_CFGVAL (EPOCH\_SECOND)

Definition at line 198 of file cfe\_time\_interface\_cfg.h.

**12.211.2.18 CFE\_MISSION\_TIME\_EPOCH\_SECONDS** #define CFE\_MISSION\_TIME\_EPOCH\_SECONDS CFE\_MISSION\_TIME\_CFGVAL (EPOCH\_SECONDS)

Definition at line 205 of file cfe\_time\_interface\_cfg.h.

**12.211.2.19 CFE\_MISSION\_TIME\_EPOCH\_YEAR** #define CFE\_MISSION\_TIME\_EPOCH\_YEAR CFE\_MISSION\_TIME\_CFGVAL (EPOCH\_YEAR)

**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 186 of file cfe\_time\_interface\_cfg.h.

**12.211.2.20 CFE\_MISSION\_TIME\_FS\_FACTOR** #define CFE\_MISSION\_TIME\_FS\_FACTOR CFE\_MISSION\_TIME\_CFGVAL (FS\_FACTOR)

**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 244 of file cfe\_time\_interface\_cfg.h.

**12.211.2.21 CFE\_MISSION\_TIME\_MAX\_ELAPSED** #define CFE\_MISSION\_TIME\_MAX\_ELAPSED CFE\_MISSION\_TIME\_CFGVAL (MAX\_ELAPSED)

Definition at line 124 of file cfe\_time\_interface\_cfg.h.

**12.211.2.22 CFE\_MISSION\_TIME\_MIN\_ELAPSED** #define CFE\_MISSION\_TIME\_MIN\_ELAPSED CFE\_MISSION\_TIME\_CFGVAL(MIN←\_ELAPSED)

**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 121 of file cfe\_time\_interface\_cfg.h.

**12.211.2.23 DEFAULT\_CFE\_MISSION\_TIME\_AT\_TONE\_WAS** #define DEFAULT\_CFE\_MISSION\_TIME\_AT\_TONE←\_WAS true

Definition at line 94 of file cfe\_time\_interface\_cfg.h.

**12.211.2.24 DEFAULT\_CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE** #define DEFAULT\_CFE\_MISSION\_TIME\_AT←\_TONE\_WILL\_BE false

Definition at line 97 of file cfe\_time\_interface\_cfg.h.

**12.211.2.25 DEFAULT\_CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI** #define DEFAULT\_CFE\_MISSION\_TIME\_CFG←\_DEFAULT\_TAI true

Definition at line 54 of file cfe\_time\_interface\_cfg.h.

**12.211.2.26 DEFAULT\_CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC** #define DEFAULT\_CFE\_MISSION\_TIME\_CFG←\_DEFAULT\_UTC false

Definition at line 57 of file cfe\_time\_interface\_cfg.h.

**12.211.2.27 DEFAULT\_CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE** #define DEFAULT\_CFE\_MISSION\_TIME\_CFG←\_FAKE\_TONE true

Definition at line 70 of file cfe\_time\_interface\_cfg.h.

**12.211.2.28 DEFAULT\_CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS** #define DEFAULT\_CFE\_MISSION\_TIME\_DEF←\_DELAY\_SECS 0

Definition at line 166 of file cfe\_time\_interface\_cfg.h.

**12.211.2.29 DEFAULT\_CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS** #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_↪  
DELAY\_SUBS 1000

Definition at line 169 of file cfe\_time\_interface\_cfg.h.

**12.211.2.30 DEFAULT\_CFE\_MISSION\_TIME\_DEF\_LEAPS** #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_LEAPS 37  
Definition at line 163 of file cfe\_time\_interface\_cfg.h.

**12.211.2.31 DEFAULT\_CFE\_MISSION\_TIME\_DEF\_MET\_SECS** #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_↪  
MET\_SECS 1000

Definition at line 151 of file cfe\_time\_interface\_cfg.h.

**12.211.2.32 DEFAULT\_CFE\_MISSION\_TIME\_DEF\_MET\_SUBS** #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_↪  
MET\_SUBS 0

Definition at line 154 of file cfe\_time\_interface\_cfg.h.

**12.211.2.33 DEFAULT\_CFE\_MISSION\_TIME\_DEF\_STCF\_SECS** #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_↪  
STCF\_SECS 1000000

Definition at line 157 of file cfe\_time\_interface\_cfg.h.

**12.211.2.34 DEFAULT\_CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS** #define DEFAULT\_CFE\_MISSION\_TIME\_DEF\_↪  
STCF\_SUBS 0

Definition at line 160 of file cfe\_time\_interface\_cfg.h.

**12.211.2.35 DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_DAY** #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_DAY 1  
Definition at line 190 of file cfe\_time\_interface\_cfg.h.

**12.211.2.36 DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_HOUR** #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_↪  
HOUR 0

Definition at line 193 of file cfe\_time\_interface\_cfg.h.

**12.211.2.37 DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_MICROS** #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_↪  
MICROS 0

Definition at line 202 of file cfe\_time\_interface\_cfg.h.

**12.211.2.38 DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_MINUTE** #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_↪  
MINUTE 0

Definition at line 196 of file cfe\_time\_interface\_cfg.h.

**12.211.2.39 DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_SECOND** #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_↪  
\_SECOND 0

Definition at line 199 of file cfe\_time\_interface\_cfg.h.

**12.211.2.40 DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_SECONDS** #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_SECONDS 315532800  
Definition at line 206 of file cfe\_time\_interface\_cfg.h.

**12.211.2.41 DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_YEAR** #define DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_YEAR 1980  
Definition at line 187 of file cfe\_time\_interface\_cfg.h.

**12.211.2.42 DEFAULT\_CFE\_MISSION\_TIME\_FS\_FACTOR** #define DEFAULT\_CFE\_MISSION\_TIME\_FS\_FACTOR 789004800  
Definition at line 245 of file cfe\_time\_interface\_cfg.h.

**12.211.2.43 DEFAULT\_CFE\_MISSION\_TIME\_MAX\_ELAPSED** #define DEFAULT\_CFE\_MISSION\_TIME\_MAX\_ELAPSED 200000  
Definition at line 125 of file cfe\_time\_interface\_cfg.h.

**12.211.2.44 DEFAULT\_CFE\_MISSION\_TIME\_MIN\_ELAPSED** #define DEFAULT\_CFE\_MISSION\_TIME\_MIN\_ELAPSED 0  
Definition at line 122 of file cfe\_time\_interface\_cfg.h.

## 12.212 cfe/modules/time/fsw/inc/cfe\_time\_internal\_cfg.h File Reference

```
#include "cfe_time_internal_cfg_values.h"
```

### Macros

- #define CFE\_PLATFORM\_TIME\_CFG\_SERVER CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_SERVER)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SERVER true
- #define CFE\_PLATFORM\_TIME\_CFG\_CLIENT CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_CLIENT)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_CLIENT false
- #define CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_VIRTUAL)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL true
- #define CFE\_PLATFORM\_TIME\_CFG\_SIGNAL CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_SIGNAL)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SIGNAL false
- #define CFE\_PLATFORM\_TIME\_CFG\_SOURCE CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_SOURCE)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SOURCE false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_SRC\_MET)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_SRC\_GPS)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS false
- #define CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_SRC\_TIME)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME false
- #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS CFE\_PLATFORM\_TIME\_CFGVAL(MAX\_DELTA\_SECS)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS 0
- #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS CFE\_PLATFORM\_TIME\_CFGVAL(MAX\_DELTA\_SUBS)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS 500000
- #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS CFE\_PLATFORM\_TIME\_CFGVAL(MAX\_LOCAL\_SECS)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS 27

- #define CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS CFE\_PLATFORM\_TIME\_CFGVAL(MAX\_LOCAL\_SUBS)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS 0
- #define CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_TONE\_LIMIT)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT 20000
- #define CFE\_PLATFORM\_TIME\_CFG\_START\_FLY CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_START\_FLY)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_START\_FLY 2
- #define CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY CFE\_PLATFORM\_TIME\_CFGVAL(CFG\_LATCH\_FLY)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY 8
- #define CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY CFE\_PLATFORM\_TIME\_CFGVAL(START\_TASK\_PRIORITY)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY 60
- #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY CFE\_PLATFORM\_TIME\_CFGVAL(TONE\_TASK\_PRIORITY)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY 25
- #define CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY CFE\_PLATFORM\_TIME\_CFGVAL(ONEHZ\_TASK\_PRIORITY)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY 25
- #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_TIME\_CFGVAL(START\_TASK\_STACK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE
- #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE CFE\_PLATFORM\_TIME\_CFGVAL(TONE\_TASK\_STACK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096
- #define CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE CFE\_PLATFORM\_TIME\_CFGVAL(ONEHZ\_TASK\_STACK\_SIZE)
- #define DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE 8192

### 12.212.1 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.

### 12.212.2 Macro Definition Documentation

#### 12.212.2.1 CFE\_PLATFORM\_TIME\_CFG\_CLIENT `#define CFE_PLATFORM_TIME_CFG_CLIENT CFE_PLATFORM_TIME_CFGVAL(CFG_CLIENT)`

Definition at line 55 of file cfe\_time\_internal\_cfg.h.

#### 12.212.2.2 CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY `#define CFE_PLATFORM_TIME_CFG_LATCH_FLY CFE_PLATFORM_TIME_CFGVAL(CFG_LATCH_FLY)`

**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 217 of file cfe\_time\_internal\_cfg.h.

**12.212.2.3 CFE\_PLATFORM\_TIME\_CFG\_SERVER** #define CFE\_PLATFORM\_TIME\_CFG\_SERVER CFE\_PLATFORM\_TIME\_CFGVAL(CFG←\_SERVER)

**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 52 of file cfe\_time\_internal\_cfg.h.

**12.212.2.4 CFE\_PLATFORM\_TIME\_CFG\_SIGNAL** #define CFE\_PLATFORM\_TIME\_CFG\_SIGNAL CFE\_PLATFORM\_TIME\_CFGVAL(CFG←\_SIGNAL)

**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 92 of file cfe\_time\_internal\_cfg.h.

**12.212.2.5 CFE\_PLATFORM\_TIME\_CFG\_SOURCE** #define CFE\_PLATFORM\_TIME\_CFG\_SOURCE CFE\_PLATFORM\_TIME\_CFGVAL(CFG←\_SOURCE)

**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 113 of file cfe\_time\_internal\_cfg.h.

**12.212.2.6 CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS**

```
#define CFE_PLATFORM_TIME_CFG_SRC_GPS CFE_PLATFORM_TIME_CFGVAL(CF
```

```
_SRC_GPS)
```

Definition at line 133 of file cfe\_time\_internal\_cfg.h.

**12.212.2.7 CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET**

```
#define CFE_PLATFORM_TIME_CFG_SRC_MET CFE_PLATFORM_TIME_CFGVAL(CF
```

```
_SRC_MET)
```

**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 130 of file cfe\_time\_internal\_cfg.h.

**12.212.2.8 CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME**

```
#define CFE_PLATFORM_TIME_CFG_SRC_TIME CFE_PLATFORM_TIME_CFGVAL(CF
```

```
_SRC_TIME)
```

Definition at line 136 of file cfe\_time\_internal\_cfg.h.

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

```
#define CFE_PLATFORM_TIME_CFG_START_FLY CFE_PLATFORM_TIME_CFGVAL(CF
```

```
_START_FLY)
```

**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 203 of file cfe\_time\_internal\_cfg.h.

**12.212.2.10 CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT** #define CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT

**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 190 of file cfe\_time\_internal\_cfg.h.

**12.212.2.11 CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL** #define CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL CFE\_PLATFORM\_TIME\_CFGVAL(CF...

**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 76 of file cfe\_time\_internal\_cfg.h.

**12.212.2.12 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS** #define CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS CFE\_PLATFORM\_TIME\_C...

**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 156 of file cfe\_time\_internal\_cfg.h.

**12.212.2.13 CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS** `#define CFE_PLATFORM_TIME_MAX_DELTA_SUBS CFE_PLATFORM_TIME_CFE_PLATFORM_TIME_MAX_DELTA_SUBS`

Definition at line 159 of file cfe\_time\_internal\_cfg.h.

**12.212.2.14 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS** `#define CFE_PLATFORM_TIME_MAX_LOCAL_SECS CFE_PLATFORM_TIME_CFE_PLATFORM_TIME_MAX_LOCAL_SECS`

**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 172 of file cfe\_time\_internal\_cfg.h.

**12.212.2.15 CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS** `#define CFE_PLATFORM_TIME_MAX_LOCAL_SUBS CFE_PLATFORM_TIME_CFE_PLATFORM_TIME_MAX_LOCAL_SUBS`

Definition at line 175 of file cfe\_time\_internal\_cfg.h.

**12.212.2.16 CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY** `#define CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY CFE_PLATFORM_TIME_CFE_PLATFORM_TIME_ONEHZ_TASK_PRIORITY`

Definition at line 239 of file cfe\_time\_internal\_cfg.h.

**12.212.2.17 CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE** `#define CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE CFE_PLATFORM_TIME_CFE_PLATFORM_TIME_ONEHZ_TASK_STACK_SIZE`

Definition at line 263 of file cfe\_time\_internal\_cfg.h.

**12.212.2.18 CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY CFE\_PLATFORM\_TIME\_CFGVAL(START\_TASK\_PRIORITY)

**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 233 of file cfe\_time\_internal\_cfg.h.

**12.212.2.19 CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_TIME\_CFGVAL(START\_TASK\_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 257 of file cfe\_time\_internal\_cfg.h.

**12.212.2.20 CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY CFE\_PLATFORM\_TIME\_CFGVAL(TONE\_TASK\_PRIORITY)

Definition at line 236 of file cfe\_time\_internal\_cfg.h.

**12.212.2.21 CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE** #define CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE CFE\_PLATFORM\_TIME\_CFGVAL(TONE\_TASK\_STACK\_SIZE)

Definition at line 260 of file cfe\_time\_internal\_cfg.h.

**12.212.2.22 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_CLIENT** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_CLIENT false

Definition at line 56 of file cfe\_time\_internal\_cfg.h.

**12.212.2.23 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY 8

Definition at line 218 of file cfe\_time\_internal\_cfg.h.

**12.212.2.24 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SERVER** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SERVER true

Definition at line 53 of file cfe\_time\_internal\_cfg.h.

**12.212.2.25 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SIGNAL** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SIGNAL false

Definition at line 93 of file cfe\_time\_internal\_cfg.h.

**12.212.2.26 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SOURCE** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SOURCE false

Definition at line 114 of file cfe\_time\_internal\_cfg.h.

**12.212.2.27 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS false

Definition at line 134 of file cfe\_time\_internal\_cfg.h.

**12.212.2.28 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET false

Definition at line 131 of file cfe\_time\_internal\_cfg.h.

**12.212.2.29 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME false

Definition at line 137 of file cfe\_time\_internal\_cfg.h.

**12.212.2.30 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_START\_FLY** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_START\_FLY 2

Definition at line 204 of file cfe\_time\_internal\_cfg.h.

**12.212.2.31 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT 20000

Definition at line 191 of file cfe\_time\_internal\_cfg.h.

**12.212.2.32 DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL** #define DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL true

Definition at line 77 of file cfe\_time\_internal\_cfg.h.

**12.212.2.33 DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS** #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS 0

Definition at line 157 of file cfe\_time\_internal\_cfg.h.

**12.212.2.34 DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS** #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS 500000

Definition at line 160 of file cfe\_time\_internal\_cfg.h.

**12.212.2.35 DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS** #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS 27

Definition at line 173 of file cfe\_time\_internal\_cfg.h.

**12.212.2.36 DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS** #define DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS 0

Definition at line 176 of file cfe\_time\_internal\_cfg.h.

**12.212.2.37 DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY 25

Definition at line 240 of file cfe\_time\_internal\_cfg.h.

**12.212.2.38 DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE 8192

Definition at line 264 of file cfe\_time\_internal\_cfg.h.

**12.212.2.39 DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY 60

Definition at line 234 of file cfe\_time\_internal\_cfg.h.

**12.212.2.40 DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE

Definition at line 258 of file cfe\_time\_internal\_cfg.h.

**12.212.2.41 DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY** #define DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY 25

Definition at line 237 of file cfe\_time\_internal\_cfg.h.

**12.212.2.42 DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE** #define DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE 4096

Definition at line 261 of file cfe\_time\_internal\_cfg.h.

## 12.213 cfe/modules/time/fsw/inc/cfe\_time\_topicids.h File Reference

```
#include "cfe_time_topicid_values.h"
```

### Macros

- #define CFE\_MISSION\_TIME\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(CMD)
- #define DEFAULT\_CFE\_MISSION\_TIME\_CMD\_TOPICID 5
- #define CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(SEND\_HK)
- #define DEFAULT\_CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID 13
- #define CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(TONE\_CMD)
- #define DEFAULT\_CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID 16
- #define CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(ONEHZ\_CMD)
- #define DEFAULT\_CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID 17
- #define CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(DATA\_CMD)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID 0
- #define CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(SEND\_CMD)
- #define DEFAULT\_CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID 2
- #define CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(HK\_TLM)
- #define DEFAULT\_CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID 5
- #define CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(DIAG\_TLM)
- #define DEFAULT\_CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID 6

### 12.213.1 Detailed Description

CFE Time Services (CFE\_TIME) Application Topic IDs

### 12.213.2 Macro Definition Documentation

#### 12.213.2.1 CFE\_MISSION\_TIME\_CMD\_TOPICID #define CFE\_MISSION\_TIME\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(CMD)

**Purpose** cFE Portable Message Numbers for Commands

**Description:**

Portable message numbers for the cFE command messages

**Limits**

Not Applicable

Definition at line 37 of file cfe\_time\_topicids.h.

#### 12.213.2.2 CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID #define CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(\_CMD)

**Purpose** cFE Portable Message Numbers for Global Messages

**Description:**

Portable message numbers for the cFE global messages

**Limits**

Not Applicable

Definition at line 55 of file cfe\_time\_topicids.h.

**12.213.2.3 CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID** #define CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(\_TLM)

Definition at line 71 of file cfe\_time\_topicids.h.

**12.213.2.4 CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID** #define CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(\_TLM)

**Purpose** cFE Portable Message Numbers for Telemetry

**Description:**

Portable message numbers for the cFE telemetry messages

**Limits**

Not Applicable

Definition at line 69 of file cfe\_time\_topicids.h.

**12.213.2.5 CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID** #define CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(\_CMD)

Definition at line 43 of file cfe\_time\_topicids.h.

**12.213.2.6 CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID** #define CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(\_CMD)

Definition at line 57 of file cfe\_time\_topicids.h.

**12.213.2.7 CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID** #define CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(\_HK)

Definition at line 39 of file cfe\_time\_topicids.h.

**12.213.2.8 CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID** #define CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID CFE\_MISSION\_TIME\_TIDVAL(\_CMD)

Definition at line 41 of file cfe\_time\_topicids.h.

**12.213.2.9 DEFAULT\_CFE\_MISSION\_TIME\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_CMD\_TOPICID 5

Definition at line 38 of file cfe\_time\_topicids.h.

**12.213.2.10 DEFAULT\_CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_↪  
DATA\_CMD\_TOPICID 0  
Definition at line 56 of file cfe\_time\_topicids.h.

**12.213.2.11 DEFAULT\_CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_↪  
DIAG\_TLM\_TOPICID 6  
Definition at line 72 of file cfe\_time\_topicids.h.

**12.213.2.12 DEFAULT\_CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_HK\_↪  
TLM\_TOPICID 5  
Definition at line 70 of file cfe\_time\_topicids.h.

**12.213.2.13 DEFAULT\_CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_↪  
ONEHZ\_CMD\_TOPICID 17  
Definition at line 44 of file cfe\_time\_topicids.h.

**12.213.2.14 DEFAULT\_CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_↪  
SEND\_CMD\_TOPICID 2  
Definition at line 58 of file cfe\_time\_topicids.h.

**12.213.2.15 DEFAULT\_CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_↪  
SEND\_HK\_TOPICID 13  
Definition at line 40 of file cfe\_time\_topicids.h.

**12.213.2.16 DEFAULT\_CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID** #define DEFAULT\_CFE\_MISSION\_TIME\_↪  
TONE\_CMD\_TOPICID 16  
Definition at line 42 of file cfe\_time\_topicids.h.

## 12.214 osal/docs/src/osal\_frontpage.dox File Reference

## 12.215 osal/docs/src/osal\_fs.dox File Reference

## 12.216 osal/docs/src/osal\_timer.dox File Reference

## 12.217 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 long osal\_offset\_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)

### 12.217.1 Detailed Description

Purpose: Unit specification for common types.

Design Notes: Assumes make file has defined processor family

### 12.217.2 Macro Definition Documentation

#### 12.217.2.1 \_\_EXTENSION\_\_ #define \_\_EXTENSION\_\_

Definition at line 64 of file common\_types.h.

**12.217.2.2 CompileTimeAssert** `#define CompileTimeAssert(`  
    `Condition,`  
    `Message )` `typedef char Message[(Condition) ? 1 : -1]`  
Definition at line 47 of file common\_types.h.

**12.217.2.3 OS\_PRINTF** `#define OS_PRINTF(`  
    `n,`  
    `m )`

Definition at line 66 of file common\_types.h.

**12.217.2.4 OS\_USED** `#define OS_USED`  
Definition at line 65 of file common\_types.h.

**12.217.2.5 OSAL\_BLOCKCOUNT\_C** `#define OSAL_BLOCKCOUNT_C(`  
    `X )` `((osal_blockcount_t)(X))`

Definition at line 178 of file common\_types.h.

**12.217.2.6 OSAL\_INDEX\_C** `#define OSAL_INDEX_C(`  
    `X )` `((osal_index_t)(X))`

Definition at line 179 of file common\_types.h.

**12.217.2.7 OSAL\_OBJTYPE\_C** `#define OSAL_OBJTYPE_C(`  
    `X )` `((osal_objtype_t)(X))`

Definition at line 180 of file common\_types.h.

**12.217.2.8 OSAL\_SIZE\_C** `#define OSAL_SIZE_C(`  
    `X )` `((size_t)(X))`

Definition at line 177 of file common\_types.h.

**12.217.2.9 OSAL\_STATUS\_C** `#define OSAL_STATUS_C(`  
    `X )` `((osal_status_t)(X))`

Definition at line 181 of file common\_types.h.

### 12.217.3 Typedef Documentation

**12.217.3.1 cpuaddr** `typedef uintptr_t cpuaddr`  
Definition at line 87 of file common\_types.h.

**12.217.3.2 cpudiff** `typedef ptrdiff_t cpudiff`  
Definition at line 89 of file common\_types.h.

**12.217.3.3 cpusize** `typedef size_t cpusize`  
Definition at line 88 of file common\_types.h.

**12.217.3.4 int16** `typedef int16_t int16`  
Definition at line 79 of file common\_types.h.

**12.217.3.5 int32** `typedef int32_t int32`  
Definition at line 80 of file common\_types.h.

**12.217.3.6 int64** `typedef int64_t int64`  
Definition at line 81 of file common\_types.h.

**12.217.3.7 int8** `typedef int8_t int8`  
Definition at line 78 of file common\_types.h.

**12.217.3.8 intptr** `typedef intptr_t intptr`  
Definition at line 86 of file common\_types.h.

**12.217.3.9 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 149 of file common\_types.h.

**12.217.3.10 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 115 of file common\_types.h.

**12.217.3.11 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 107 of file common\_types.h.

**12.217.3.12 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 132 of file common\_types.h.

**12.217.3.13 `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 137 of file common\_types.h.

**12.217.3.14 `osal_offset_t`** `typedef long osal_offset_t`  
A type used to represent an offset into a file  
This is used with file system implementation.  
Definition at line 122 of file common\_types.h.

**12.217.3.15 `osal_status_t`** `typedef int32 osal_status_t`  
The preferred type to represent OSAL status codes defined in [osapi-error.h](#)  
Definition at line 142 of file common\_types.h.

**12.217.3.16 `uint16`** `typedef uint16_t uint16`  
Definition at line 83 of file common\_types.h.

**12.217.3.17 `uint32`** `typedef uint32_t uint32`  
Definition at line 84 of file common\_types.h.

**12.217.3.18 `uint64`** `typedef uint64_t uint64`  
Definition at line 85 of file common\_types.h.

**12.217.3.19 `uint8`** `typedef uint8_t uint8`  
Definition at line 82 of file common\_types.h.

## 12.217.4 Function Documentation

**12.217.4.1 `CompileTimeAssert()` [1/9]** `CompileTimeAssert (`  
 `sizeof(cpuaddr) >=sizeof(void *) ,`  
 `TypePtrWrongSize )`

**12.217.4.2 `CompileTimeAssert()` [2/9]** `CompileTimeAssert (`  
 `sizeof(int16) = =2,`  
 `Typeint16WrongSize )`

**12.217.4.3 `CompileTimeAssert()` [3/9]** `CompileTimeAssert (`  
 `sizeof(int32) = =4,`  
 `Typeint32WrongSize )`

**12.217.4.4 CompileTimeAssert() [4/9]** `CompileTimeAssert (`  
 `sizeof(int64) = =8,`  
 `Typeint64WrongSize )`

**12.217.4.5 CompileTimeAssert() [5/9]** `CompileTimeAssert (`  
 `sizeof(int8) = =1,`  
 `Typeint8WrongSize )`

**12.217.4.6 CompileTimeAssert() [6/9]** `CompileTimeAssert (`  
 `sizeof(uint16) = =2,`  
 `TypeUint16WrongSize )`

**12.217.4.7 CompileTimeAssert() [7/9]** `CompileTimeAssert (`  
 `sizeof(uint32) = =4,`  
 `TypeUint32WrongSize )`

**12.217.4.8 CompileTimeAssert() [8/9]** `CompileTimeAssert (`  
 `sizeof(uint64) = =8,`  
 `TypeUint64WrongSize )`

**12.217.4.9 CompileTimeAssert() [9/9]** `CompileTimeAssert (`  
 `sizeof(uint8) = =1,`  
 `TypeUint8WrongSize )`

## 12.218 osal/src/os/inc/osapi-binsem.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Data Structures

- struct `OS_bin_sem_prop_t`  
*OSAL binary semaphore properties.*

### Macros

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

### Functions

- int32 `OS_BinSemCreate (osal_id_t *sem_id, const char *sem_name, uint32 sem_initial_value, uint32 options)`  
*Creates a binary semaphore.*
- int32 `OS_BinSemFlush (osal_id_t sem_id)`

*Unblock all tasks pending on the specified semaphore.*

- `int32 OS_BinSemGive (osal_id_t sem_id)`  
*Increment the semaphore value.*
- `int32 OS_BinSemTake (osal_id_t sem_id)`  
*Decrement the semaphore value.*
- `int32 OS_BinSemTimedWait (osal_id_t sem_id, uint32 msecs)`  
*Decrement the semaphore value with a timeout.*
- `int32 OS_BinSemDelete (osal_id_t sem_id)`  
*Deletes the specified Binary Semaphore.*
- `int32 OS_BinSemGetIdByName (osal_id_t *sem_id, const char *sem_name)`  
*Find an existing semaphore ID by name.*
- `int32 OS_BinSemGetInfo (osal_id_t sem_id, OS_bin_sem_prop_t *bin_prop)`  
*Fill a property object buffer with details regarding the resource.*

#### 12.218.1 Detailed Description

Declarations and prototypes for binary semaphores

### 12.219 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)`

#### 12.219.1 Detailed Description

Declarations and prototypes for OSAL BSP

### 12.220 osal/src/os/inc/osapi-clock.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

#### Data Structures

- `struct OS_time_t`  
*OSAL time interval structure.*

## Macros

- `#define OS_TIME_MAX ((OS_time_t){INT64_MAX})`  
*The maximum value for `OS_time_t`.*
- `#define OS_TIME_ZERO ((OS_time_t){0})`  
*The zero value for `OS_time_t`.*
- `#define OS_TIME_MIN ((OS_time_t){INT64_MIN})`  
*The minimum value for `OS_time_t`.*

## Enumerations

- `enum { OS_TIME_TICK_RESOLUTION_NS = 100, OS_TIME_TICKS_PER_SECOND = 1000000000 / OS_TIME_TICK_RESOLUTION_NS, OS_TIME_TICKS_PER_MSEC = 1000000 / OS_TIME_TICK_RESOLUTION_NS, OS_TIME_TICKS_PER_USEC = 1000 / OS_TIME_TICK_RESOLUTION_NS }`  
*Multippliers/divisors to convert ticks into standardized units.*

## Functions

- `int32 OS_GetMonotonicTime (OS_time_t *time_struct)`  
*Get the monotonic time.*
- `int32 OS_GetLocalTime (OS_time_t *time_struct)`  
*Get the local time.*
- `int32 OS_SetLocalTime (const OS_time_t *time_struct)`  
*Set the local time.*
- `OS_time_t OS_TimeFromRelativeMilliseconds (int32 relative_msec)`  
*Gets an absolute time value relative to the current time.*
- `int32 OS_TimeToRelativeMilliseconds (OS_time_t time)`  
*Gets a relative time value from an absolute time.*
- `static int64 OS_TimeGetTotalSeconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to whole number of seconds.*
- `static OS_time_t OS_TimeFromTotalSeconds (int64 tm)`  
*Get an `OS_time_t` interval object from an integer number of seconds.*
- `static int64 OS_TimeGetTotalMilliseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to millisecond units.*
- `static OS_time_t OS_TimeFromTotalMilliseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of milliseconds.*
- `static int64 OS_TimeGetTotalMicroseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to microsecond units.*
- `static OS_time_t OS_TimeFromTotalMicroseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of microseconds.*
- `static int64 OS_TimeGetTotalNanoseconds (OS_time_t tm)`  
*Get interval from an `OS_time_t` object normalized to nanosecond units.*
- `static OS_time_t OS_TimeFromTotalNanoseconds (int64 tm)`  
*Get an `OS_time_t` interval object from a integer number of nanoseconds.*
- `static int64 OS_TimeGetFractionalPart (OS_time_t tm)`  
*Get subseconds portion (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetSubsecondsPart (OS_time_t tm)`  
*Get 32-bit normalized subseconds (fractional part only) from an `OS_time_t` object.*
- `static uint32 OS_TimeGetMillisecondsPart (OS_time_t tm)`

*Get milliseconds portion (fractional part only) from an `OS_time_t` object.*

- static `uint32 OS_TimeGetMicrosecondsPart (OS_time_t tm)`

*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.*

- static `bool OS_TimeEqual (OS_time_t time1, OS_time_t time2)`

*Checks if two time values are equal.*

- static `int8_t OS_TimeGetSign (OS_time_t time)`

*Checks the sign of the time value.*

- static `int8_t OS_TimeCompare (OS_time_t time1, OS_time_t time2)`

*Compares two time values.*

## 12.220.1 Detailed Description

Declarations and prototypes for osapi-clock module

## 12.220.2 Macro Definition Documentation

### 12.220.2.1 OS\_TIME\_MAX `#define OS_TIME_MAX ((OS_time_t) {INT64_MAX})`

The maximum value for `OS_time_t`.

This is the largest positive (future) time that is representable in an `OS_time_t` value.

Definition at line 56 of file osapi-clock.h.

### 12.220.2.2 OS\_TIME\_MIN `#define OS_TIME_MIN ((OS_time_t) {INT64_MIN})`

The minimum value for `OS_time_t`.

This is the largest negative (past) time that is representable in an `OS_time_t` value.

Definition at line 71 of file osapi-clock.h.

### 12.220.2.3 OS\_TIME\_ZERO `#define OS_TIME_ZERO ((OS_time_t) {0})`

The zero value for `OS_time_t`.

This is a reasonable initializer/placeholder value for an `OS_time_t`

Definition at line 63 of file osapi-clock.h.

### 12.220.3 Enumeration Type Documentation

#### 12.220.3.1 anonymous enum anonymous enum

Multipliers/divisors to convert ticks into standardized units.

Various fixed conversion factor constants used by the conversion routines

A 100ns tick time allows max intervals of about +/- 14000 years in a 64-bit signed integer value.

##### Note

Applications should not directly use these values, but rather use conversion routines below to obtain standardized units (seconds/microseconds/etc).

##### Enumerator

OS_TIME_TICK_RESOLUTION_NS	
OS_TIME_TICKS_PER_SECOND	
OS_TIME_TICKS_PER_MSEC	
OS_TIME_TICKS_PER_USEC	

Definition at line 84 of file osapi-clock.h.

## 12.221 osal/src/os/inc/osapi-common.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Typedefs

- **typedef int32(\* OS\_EventHandler\_t)** (OS\_Event\_t event, **osal\_id\_t** object\_id, void \*data)  
*A callback routine for event handling.*

### Enumerations

- **enum OS\_Event\_t {**  
**OS\_EVENT\_RESERVED** = 0 , **OS\_EVENT\_RESOURCE\_ALLOCATED** , **OS\_EVENT\_RESOURCE\_CREATED**  
, **OS\_EVENT\_RESOURCE\_DELETED** ,  
**OS\_EVENT\_TASK\_STARTUP** , **OS\_EVENT\_MAX** **}**

*A set of events that can be used with BSP event callback routines.*

### Functions

- **void OS\_Application\_Startup (void)**  
*Application startup.*
- **void OS\_Application\_Run (void)**  
*Application run.*
- **int32 OS\_API\_Init (void)**  
*Initialization of API.*
- **void OS\_API\_Teardown (void)**  
*Teardown/de-initialization of OSAL API.*

- void [OS\\_IdleLoop](#) (void)  
*Background thread implementation - waits forever for events to occur.*
- void [OS\\_DeleteAllObjects](#) (void)  
*delete all resources created in OSAL.*
- void [OS\\_ApplicationShutdown](#) (uint8 flag)  
*Initiate orderly shutdown.*
- void [OS\\_ApplicationExit](#) (int32 Status)  
*Exit/Abort the application.*
- int32 [OS\\_RegisterEventHandler](#) (OS\_EventHandler\_t handler)  
*Callback routine registration.*
- size\_t [OS\\_strlen](#) (const char \*s, size\_t maxlen)  
*get string length*

### 12.221.1 Detailed Description

Declarations and prototypes for general OSAL functions that are not part of a subsystem

### 12.221.2 Typedef Documentation

**12.221.2.1 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.

### 12.221.3 Enumeration Type Documentation

**12.221.3.1 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.

## 12.222 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.*

### 12.222.1 Detailed Description

Declarations and prototypes for condition variables

## 12.223 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.*

### 12.223.1 Detailed Description

General constants for OSAL that are shared across subsystems

### 12.223.2 Macro Definition Documentation

#### 12.223.2.1 OS\_CHECK `#define OS_CHECK (0)`

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

#### 12.223.2.2 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.

#### 12.223.2.3 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.

**12.223.2.4 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.

**12.223.2.5 OS\_PEND** #define OS\_PEND (-1)  
 Definition at line 34 of file osapi-constants.h.

## 12.224 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.*

### 12.224.1 Detailed Description

Declarations and prototypes for counting semaphores

## 12.225 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.*

### 12.225.1 Detailed Description

Declarations and prototypes for directories

### 12.225.2 Macro Definition Documentation

#### 12.225.2.1 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.

## 12.226 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\\_ERR\\_TRY AGAIN](#) (-43)  
*Failure is temporary in nature, call may be repeated.*
- #define [OS\\_ERR\\_EMPTY\\_SET](#) (-44)  
*Address or name lookup returned no results.*
- #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.*

### 12.226.1 Detailed Description

OSAL error code definitions

### 12.226.2 Macro Definition Documentation

#### 12.226.2.1 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.

#### 12.226.2.2 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.

### 12.226.3 Typedef Documentation

#### 12.226.3.1 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.

#### 12.226.3.2 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.

## 12.227 osal/src/os/inc/osapi-file.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

### Data Structures

- struct [OS\\_file\\_prop\\_t](#)

*OSAL file properties.*

- struct [os\\_fstat\\_t](#)

*File system status.*

## Macros

- #define OS\_READ\_ONLY 0
- #define OS\_WRITE\_ONLY 1
- #define OS\_READ\_WRITE 2
- #define OS\_SEEK\_SET 0
- #define OS\_SEEK\_CUR 1
- #define OS\_SEEK\_END 2
- #define OS\_FILESTAT\_MODE(x) ((x). FileModeBits)  
*Access file stat mode bits.*
- #define OS\_FILESTAT\_ISDIR(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_DIR)  
*File stat is directory logical.*
- #define OS\_FILESTAT\_EXEC(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_EXEC)  
*File stat is executable logical.*
- #define OS\_FILESTAT\_WRITE(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_WRITE)  
*File stat is write enabled logical.*
- #define OS\_FILESTAT\_READ(x) ((x). FileModeBits & OS\_FILESTAT\_MODE\_READ)  
*File stat is read enabled logical.*
- #define OS\_FILESTAT\_SIZE(x) ((x). FileSize)  
*Access file stat size field.*
- #define OS\_FILESTAT\_TIME(x) (OS\_TimeGetTotalSeconds((x). FileTime))  
*Access file stat time field as a whole number of seconds.*

## Enumerations

- enum { OS\_FILESTAT\_MODE\_EXEC = 0x00001 , OS\_FILESTAT\_MODE\_WRITE = 0x00002 , OS\_FILESTAT\_MODE\_READ = 0x00004 , OS\_FILESTAT\_MODE\_DIR = 0x10000 }  
*File stat mode bits.*
- enum OS\_file\_flag\_t { OS\_FILE\_FLAG\_NONE = 0x00 , OS\_FILE\_FLAG\_CREATE = 0x01 , OS\_FILE\_FLAG\_TRUNCATE = 0x02 }  
*Flags that can be used with opening of a file (bitmask)*

## Functions

- int32 OS\_OpenCreate (osal\_id\_t \*filedes, const char \*path, int32 flags, int32 access\_mode)  
*Open or create a file.*
- int32 OS\_close (osal\_id\_t filedes)  
*Closes an open file handle.*
- int32 OS\_read (osal\_id\_t filedes, void \*buffer, size\_t nbytes)  
*Read from a file handle.*
- int32 OS\_write (osal\_id\_t filedes, const void \*buffer, size\_t nbytes)  
*Write to a file handle.*
- int32 OS\_TimedReadAbs (osal\_id\_t filedes, void \*buffer, size\_t nbytes, OS\_time\_t abstime)  
*File/Stream input read with a timeout.*
- int32 OS\_TimedRead (osal\_id\_t filedes, void \*buffer, size\_t nbytes, int32 timeout)  
*File/Stream input read with a timeout.*
- int32 OS\_TimedWriteAbs (osal\_id\_t filedes, const void \*buffer, size\_t nbytes, OS\_time\_t abstime)  
*File/Stream output write with a timeout.*
- int32 OS\_TimedWrite (osal\_id\_t filedes, const void \*buffer, size\_t nbytes, int32 timeout)

- File/Stream output write with a timeout.*
- `int32 OS_FileAllocate (osal_id_t filedes, osal_offset_t offset, osal_offset_t len)`  
*Pre-allocates space at the given file location.*
  - `int32 OS_FileTruncate (osal_id_t filedes, osal_offset_t len)`  
*Changes the size of the file.*
  - `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, osal_offset_t 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.*

### 12.227.1 Detailed Description

Declarations and prototypes for file objects

### 12.227.2 Macro Definition Documentation

**12.227.2.1 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.

**12.227.2.2 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.

**12.227.2.3 OS\_FILESTAT\_MODE** #define OS\_FILESTAT\_MODE ( x ) ((x). FileModeBits)

Access file stat mode bits.

Definition at line 88 of file osapi-file.h.

**12.227.2.4 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.

**12.227.2.5 OS\_FILESTAT\_SIZE** #define OS\_FILESTAT\_SIZE ( x ) ((x).FileSize)

Access file stat size field.

Definition at line 98 of file osapi-file.h.

**12.227.2.6 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.

**12.227.2.7 OS\_FILESTAT\_WRITE** #define OS\_FILESTAT\_WRITE ( x ) ((x). FileModeBits & OS\_FILESTAT\_MODE\_WRITE)

File stat is write enabled logical.

Definition at line 94 of file osapi-file.h.

## 12.227.3 Enumeration Type Documentation

**12.227.3.1 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.

**12.227.3.2 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.

## 12.228 osal/src/os/inc/osapi-filesystem.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Data Structures

- struct [os\\_fsinfo\\_t](#)  
*OSAL file system info.*
- struct [OS\\_statvfs\\_t](#)

### Macros

- #define [OS\\_CHK\\_ONLY](#) 0
- #define [OS\\_REPAIR](#) 1

### Functions

- [int32 OS\\_FileSysAddFixedMap \(osal\\_id\\_t \\*filesystem\\_id, const char \\*phys\\_path, const char \\*virt\\_path\)](#)  
*Create a fixed mapping between an existing directory and a virtual OSAL mount point.*
- [int32 OS\\_mkfs \(char \\*address, const char \\*devname, const char \\*volname, size\\_t blocksize, osal\\_blockcount\\_t numblocks\)](#)  
*Makes a file system on the target.*
- [int32 OS\\_mount \(const char \\*devname, const char \\*mountpoint\)](#)  
*Mounts a file system.*
- [int32 OS\\_initfs \(char \\*address, const char \\*devname, const char \\*volname, size\\_t blocksize, osal\\_blockcount\\_t numblocks\)](#)  
*Initializes an existing file system.*
- [int32 OS\\_rmfs \(const char \\*devname\)](#)  
*Removes a file system.*
- [int32 OS\\_unmount \(const char \\*mountpoint\)](#)  
*Unmounts a mounted file system.*
- [int32 OS\\_FileSysStatVolume \(const char \\*name, OS\\_statvfs\\_t \\*statbuf\)](#)  
*Obtains information about size and free space in a volume.*
- [int32 OS\\_chkfs \(const char \\*name, bool repair\)](#)  
*Checks the health of a file system and repairs it if necessary.*
- [int32 OS\\_FS\\_GetPhysDriveName \(char \\*PhysDriveName, const char \\*MountPoint\)](#)  
*Obtains the physical drive name associated with a mount point.*
- [int32 OS\\_TranslatePath \(const char \\*VirtualPath, char \\*LocalPath\)](#)  
*Translates an OSAL Virtual file system path to a host Local path.*
- [int32 OS\\_GetFsInfo \(os\\_fsinfo\\_t \\*filesystem\\_info\)](#)  
*Returns information about the file system.*

### 12.228.1 Detailed Description

Declarations and prototypes for file systems

### 12.228.2 Macro Definition Documentation

#### 12.228.2.1 OS\_CHK\_ONLY #define OS\_CHK\_ONLY 0

Unused, API takes bool

Definition at line 31 of file osapi-filesystems.h.

#### 12.228.2.2 OS\_REPAIR #define OS\_REPAIR 1

Unused, API takes bool

Definition at line 32 of file osapi-filesystems.h.

## 12.229 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.*

### 12.229.1 Detailed Description

Declarations and prototypes for heap functions

## 12.230 osal/src/os/inc/osapi-idmap.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Macros

- [#define OS\\_OBJECT\\_INDEX\\_MASK 0xFFFF](#)  
*Object index mask.*
- [#define OS\\_OBJECT\\_TYPE\\_SHIFT 16](#)  
*Object type shift.*
- [#define OS\\_OBJECT\\_TYPE\\_UNDEFINED 0x00](#)  
*Object type undefined.*
- [#define OS\\_OBJECT\\_TYPE\\_OS\\_TASK 0x01](#)  
*Object task type.*

- #define OS\_OBJECT\_TYPE\_OS\_QUEUE 0x02  
*Object queue type.*
- #define OS\_OBJECT\_TYPE\_OS\_COUNTSEM 0x03  
*Object counting semaphore type.*
- #define OS\_OBJECT\_TYPE\_OS\_BINSEM 0x04  
*Object binary semaphore type.*
- #define OS\_OBJECT\_TYPE\_OS\_MUTEX 0x05  
*Object mutex type.*
- #define OS\_OBJECT\_TYPE\_OS\_STREAM 0x06  
*Object stream type.*
- #define OS\_OBJECT\_TYPE\_OS\_DIR 0x07  
*Object directory type.*
- #define OS\_OBJECT\_TYPE\_OS\_TIMEBASE 0x08  
*Object timebase type.*
- #define OS\_OBJECT\_TYPE\_OS\_TIMECB 0x09  
*Object timer callback type.*
- #define OS\_OBJECT\_TYPE\_OS\_MODULE 0x0A  
*Object module type.*
- #define OS\_OBJECT\_TYPE\_OS\_FILESYS 0x0B  
*Object file system type.*
- #define OS\_OBJECT\_TYPE\_OS\_CONSOLE 0x0C  
*Object console type.*
- #define OS\_OBJECT\_TYPE\_OS\_CONDVAR 0x0D  
*Object condition variable type.*
- #define OS\_OBJECT\_TYPE\_OS\_RWLOCK 0x0E  
*Object readers-writer lock 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\_EachObjectOfType** (**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*

### 12.230.1 Detailed Description

Declarations and prototypes for object IDs

### 12.230.2 Macro Definition Documentation

#### 12.230.2.1 OS\_OBJECT\_INDEX\_MASK #define OS\_OBJECT\_INDEX\_MASK 0xFFFF

Object index mask.

Definition at line 32 of file osapi-idmap.h.

#### 12.230.2.2 OS\_OBJECT\_TYPE\_SHIFT #define OS\_OBJECT\_TYPE\_SHIFT 16

Object type shift.

Definition at line 33 of file osapi-idmap.h.

## 12.231 osal/src/os/inc/osapi-macros.h File Reference

```
#include <stdio.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.*

### 12.231.1 Detailed Description

Macro definitions that are used across all OSAL subsystems

### 12.231.2 Macro Definition Documentation

```
12.231.2.1 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 130 of file osapi-macros.h.

```
12.231.2.2 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 104 of file osapi-macros.h.

```
12.231.2.3 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 154 of file osapi-macros.h.

```
12.231.2.4 BUGREPORT #define BUGREPORT(
    ... ) OS_printf(__VA_ARGS__)
```

Definition at line 87 of file osapi-macros.h.

```
12.231.2.5 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 145 of file osapi-macros.h.

## 12.232 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.*

### 12.232.1 Detailed Description

Declarations and prototypes for module subsystem

### 12.232.2 Macro Definition Documentation

#### 12.232.2.1 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.

#### 12.232.2.2 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.

## 12.233 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.*

#### 12.233.1 Detailed Description

Declarations and prototypes for mutexes

### 12.234 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.*

#### 12.234.1 Detailed Description

Declarations and prototypes for network subsystem

### 12.235 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.*

#### 12.235.1 Detailed Description

Declarations and prototypes for printf/console output

## 12.236 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.*

### 12.236.1 Detailed Description

Declarations and prototypes for queue subsystem

## 12.237 osal/src/os/inc/osapi-rwlock.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
```

### Data Structures

- struct **OS\_rwlock\_prop\_t**

*OSAL rwlock properties.*

### Functions

- **int32 OS\_RwLockCreate (osal\_id\_t \*rw\_id, const char \*rw\_name, uint32 options)**  
*Creates an readers-writer lock (rwlock)*
- **int32 OS\_RwLockReadGive (osal\_id\_t rw\_id)**  
*Releases the reader lock on the rwlock object referenced by rw\_id.*
- **int32 OS\_RwLockWriteGive (osal\_id\_t rw\_id)**  
*Releases the rwlock object referenced by rw\_id.*
- **int32 OS\_RwLockReadTake (osal\_id\_t rw\_id)**  
*Acquire the rwlock object as a read lock as referenced by rw\_id.*

- `int32 OS_RwLockWriteTake (osal_id_t rw_id)`  
*Acquire the rwlock object as a write lock as referenced by rw\_id.*
- `int32 OS_RwLockDelete (osal_id_t rw_id)`  
*Deletes the specified RwLock.*
- `int32 OS_RwLockGetIdByName (osal_id_t *rw_id, const char *rw_name)`  
*Find an existing rwlock ID by name.*
- `int32 OS_RwLockGetInfo (osal_id_t rw_id, OS_rwlock_prop_t *rw_prop)`  
*Fill a property object buffer with details regarding the resource.*

### 12.237.1 Detailed Description

Declarations and prototypes for rwlocks

## 12.238 osal/src/os/inc/osapi-select.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

### Data Structures

- struct `OS_FdSet`  
*An abstract structure capable of holding several OSAL IDs.*

### Enumerations

- enum `OS_StreamState_t`{  
  `OS_STREAM_STATE_BOUND` = 0x01 , `OS_STREAM_STATE_CONNECTED` = 0x02 , `OS_STREAM_STATE_READABLE` = 0x04 , `OS_STREAM_STATE_WRITABLE` = 0x08 ,  
  `OS_STREAM_STATE_LISTENING` = 0x10 }

*For the `OS_SelectSingle()` function's in/out `StateFlags` parameter, the state(s) of the stream and the result of the select is a combination of one or more of these states.*

### Functions

- `int32 OS_SelectMultipleAbs (OS_FdSet *ReadSet, OS_FdSet *WriteSet, OS_time_t abs_timeout)`  
*Wait for events across multiple file handles.*
- `int32 OS_SelectMultiple (OS_FdSet *ReadSet, OS_FdSet *WriteSet, int32 msecs)`  
*Wait for events across multiple file handles.*
- `int32 OS_SelectSingleAbs (osal_id_t objid, uint32 *StateFlags, OS_time_t abs_timeout)`  
*Wait for events on a single file handle.*
- `int32 OS_SelectSingle (osal_id_t objid, uint32 *StateFlags, int32 msecs)`  
*Wait for events on a single file handle.*
- `int32 OS_SelectFdZero (OS_FdSet *Set)`  
*Clear a FdSet structure.*
- `int32 OS_SelectFdAdd (OS_FdSet *Set, osal_id_t objid)`  
*Add an ID to an FdSet structure.*
- `int32 OS_SelectFdClear (OS_FdSet *Set, osal_id_t objid)`  
*Clear an ID from an FdSet structure.*
- `bool OS_SelectFdIsSet (const OS_FdSet *Set, osal_id_t objid)`  
*Check if an FdSet structure contains a given ID.*

### 12.238.1 Detailed Description

Declarations and prototypes for select abstraction

### 12.238.2 Enumeration Type Documentation

#### 12.238.2.1 OS\_StreamState\_t enum `OS_StreamState_t`

For the `OS_SelectSingle()` function's in/out StateFlags parameter, the state(s) of the stream and the result of the select is a combination of one or more of these states.

See also

[OS\\_SelectSingle\(\)](#)

Enumerator

<code>OS_STREAM_STATE_BOUND</code>	whether the stream is bound
<code>OS_STREAM_STATE_CONNECTED</code>	whether the stream is connected
<code>OS_STREAM_STATE_READABLE</code>	whether the stream is readable
<code>OS_STREAM_STATE_WRITABLE</code>	whether the stream is writable
<code>OS_STREAM_STATE_LISTENING</code>	whether the stream is listening

Definition at line 56 of file osapi-select.h.

## 12.239 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.*

### 12.239.1 Detailed Description

Declarations and prototypes for shell abstraction

## 12.240 osal/src/os/inc/osapi-sockets.h File Reference

```
#include "osconfig.h"
#include "common_types.h"
#include "osapi-clock.h"
```

### Data Structures

- union `OS_SockAddrData_t`

- **Storage buffer for generic network address.**
- struct **OS\_SockAddr\_t**
  - Encapsulates a generic network address.*
- struct **OS\_socket\_prop\_t**
  - Encapsulates socket properties.*
- union **OS\_socket\_optval**
  - Socket option value.*

## Macros

- #define **OS\_SOCKET\_MAX\_LEN** 28

## Typedefs

- typedef enum **OS\_socket\_option** **OS\_socket\_option\_t**
  - Socket option identifier.*
- typedef union **OS\_socket\_optval** **OS\_socket\_optval\_t**
  - Socket option value.*

## 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.*
- enum **OS\_socket\_option** { **OS\_socket\_option\_UNDEFINED** , **OS\_socket\_option\_IP\_DSCP** , **OS\_socket\_option\_MAX** }
  - Socket option identifier.*

## Functions

- int32 **OS\_SocketAddrInit** (**OS\_SockAddr\_t** \*Addr, **OS\_SocketDomain\_t** Domain)
  - Initialize a socket address structure to hold an address of the given family.*
- int32 **OS\_SocketAddrToString** (char \*buffer, size\_t buflen, const **OS\_SockAddr\_t** \*Addr)
  - Get a string representation of a network host address.*
- int32 **OS\_SocketAddrFromString** (**OS\_SockAddr\_t** \*Addr, const char \*string)
  - Set a network host address from a string representation.*
- int32 **OS\_SocketAddrGetPort** (uint16 \*PortNum, const **OS\_SockAddr\_t** \*Addr)
  - Get the port number of a network address.*
- int32 **OS\_SocketAddrSetPort** (**OS\_SockAddr\_t** \*Addr, uint16 PortNum)
  - Set the port number of a network address.*
- int32 **OS\_SocketOpen** (**osal\_id\_t** \*sock\_id, **OS\_SocketDomain\_t** Domain, **OS\_SocketType\_t** Type)
  - Opens a socket.*
- int32 **OS\_SocketBind** (**osal\_id\_t** sock\_id, const **OS\_SockAddr\_t** \*Addr)
  - Binds a socket to a given local address and enter listening (server) mode.*

- `int32 OS_SocketListen (osal_id_t sock_id)`  
*Places the specified socket into a listening state.*
- `int32 OS_SocketBindAddress (osal_id_t sock_id, const OS_SockAddr_t *Addr)`  
*Binds a socket to a given local address.*
- `int32 OS_SocketConnectAbs (osal_id_t sock_id, const OS_SockAddr_t *Addr, OS_time_t abs_timeout)`  
*Connects a socket to a given remote address.*
- `int32 OS_SocketConnect (osal_id_t sock_id, const OS_SockAddr_t *Addr, int32 timeout)`  
*Connects a socket to a given remote address.*
- `int32 OS_SocketShutdown (osal_id_t sock_id, OS_SocketShutdownMode_t Mode)`  
*Implement graceful shutdown of a stream socket.*
- `int32 OS_SocketAcceptAbs (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, OS_time_t abs_timeout)`  
*Waits for and accept the next incoming connection on the given socket.*
- `int32 OS_SocketAccept (osal_id_t sock_id, osal_id_t *connsock_id, OS_SockAddr_t *Addr, int32 timeout)`  
*Waits for and accept the next incoming connection on the given socket.*
- `int32 OS_SocketRecvFromAbs (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, OS_time_t abs_timeout)`  
*Reads data from a message-oriented (datagram) socket.*
- `int32 OS_SocketRecvFrom (osal_id_t sock_id, void *buffer, size_t buflen, OS_SockAddr_t *RemoteAddr, int32 timeout)`  
*Reads data from a message-oriented (datagram) socket.*
- `int32 OS_SocketSendTo (osal_id_t sock_id, const void *buffer, size_t buflen, const OS_SockAddr_t *RemoteAddr)`  
*Sends data to a message-oriented (datagram) socket.*
- `int32 OS_SocketGetIdByName (osal_id_t *sock_id, const char *sock_name)`  
*Gets an OSAL ID from a given name.*
- `int32 OS_SocketGetInfo (osal_id_t sock_id, OS_socket_prop_t *sock_prop)`  
*Gets information about an OSAL Socket ID.*
- `int32 OS_SocketgetOption (osal_id_t sock_id, OS_socket_option_t opt_id, OS_socket_optval_t *optval)`  
*Gets the value of a socket option.*
- `int32 OS_SocketSetOption (osal_id_t sock_id, OS_socket_option_t opt_id, const OS_socket_optval_t *optval)`  
*Sets the value of a socket option.*

### 12.240.1 Detailed Description

Declarations and prototypes for sockets abstraction

### 12.240.2 Macro Definition Documentation

#### 12.240.2.1 OS SOCKADDR\_MAX\_LEN #define OS SOCKADDR\_MAX\_LEN 28

Definition at line 46 of file osapi-sockets.h.

### 12.240.3 Typedef Documentation

#### 12.240.3.1 OS socket option t typedef enum OS socket option OS socket option t

Socket option identifier.

This is used with `OS_SocketGetOption()` and `OS_SocketSetOption()` to specify which option to get or set, respectively.

**12.240.3.2 OS\_socket\_optval\_t** `typedef union OS_socket_optval OS_socket_optval_t`  
Socket option value.

This is used with [OS\\_SocketGetOption\(\)](#) and [OS\\_SocketSetOption\(\)](#) to store the option value that is get or set, respectively. Currently only integers values are relevant but defining as a union will allow other types to be transparently added in the future if needed.

#### 12.240.4 Enumeration Type Documentation

**12.240.4.1 OS\_socket\_option** `enum OS_socket_option`

Socket option identifier.

This is used with [OS\\_SocketGetOption\(\)](#) and [OS\\_SocketSetOption\(\)](#) to specify which option to get or set, respectively.

Enumerator

<code>OS_socket_option_UNDEFINED</code>	Placeholder, no-op if set, always reads 0.
<code>OS_socket_option_IP_DSCP</code>	Get/Set the value for the IP DSCP/Differentiated Services field
<code>OS_socket_option_MAX</code>	Placeholder, marks 1+ the highest valid value

Definition at line 135 of file osapi-sockets.h.

**12.240.4.2 OS\_SocketDomain\_t** `enum OS_SocketDomain_t`

Socket domain.

Enumerator

<code>OS_SocketDomain_INVALID</code>	Invalid.
<code>OS_SocketDomain_INET</code>	IPv4 address family, most commonly used)
<code>OS_SocketDomain_INET6</code>	IPv6 address family, depends on OS/network stack support.
<code>OS_SocketDomain_MAX</code>	Maximum.

Definition at line 61 of file osapi-sockets.h.

**12.240.4.3 OS\_SocketShutdownMode\_t** `enum OS_SocketShutdownMode_t`

Shutdown Mode.

Enumerator

<code>OS_SocketShutdownMode_NONE</code>	Reserved value, no effect.
<code>OS_SocketShutdownMode_SHUT_READ</code>	Disable future reading.
<code>OS_SocketShutdownMode_SHUT_WRITE</code>	Disable future writing.
<code>OS_SocketShutdownMode_SHUT_READWRITE</code>	Disable future reading or writing.

Definition at line 80 of file osapi-sockets.h.

**12.240.4.4 OS\_SocketType\_t** `enum OS_SocketType_t`

Socket type.

**Enumerator**

<code>OS_SocketType_INVALID</code>	Invalid.
<code>OS_SocketType_DATAGRAM</code>	A connectionless, message-oriented socket.
<code>OS_SocketType_STREAM</code>	A stream-oriented socket with the concept of a connection.
<code>OS_SocketType_MAX</code>	Maximum.

Definition at line 70 of file osapi-sockets.h.

## 12.241 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.*

### 12.241.1 Detailed Description

Declarations and prototypes for task abstraction

### 12.241.2 Macro Definition Documentation

#### 12.241.2.1 OS\_FP\_ENABLED `#define OS_FP_ENABLED 1`

Floating point enabled state for a task.

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

#### 12.241.2.2 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.

#### 12.241.2.3 OSAL\_PRIORITY\_C `#define OSAL_PRIORITY_C(`

`X ) ((osal_priority_t) {X})`

Definition at line 46 of file osapi-task.h.

#### 12.241.2.4 OSAL\_STACKPTR\_C `#define OSAL_STACKPTR_C(`

`X ) ((osal_stackptr_t) {X})`

Definition at line 53 of file osapi-task.h.

#### 12.241.2.5 OSAL\_TASK\_STACK\_ALLOCATE `#define OSAL_TASK_STACK_ALLOCATE OSAL_STACKPTR_C(NULL)`

Definition at line 54 of file osapi-task.h.

### 12.241.3 Typedef Documentation

**12.241.3.1 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.

**12.241.3.2 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.

**12.241.3.3 osal\_task** `typedef void osal_task`

For task entry point.

Definition at line 68 of file osapi-task.h.

**12.241.4 Function Documentation****12.241.4.1 osal\_task()** `typedef osal_task (`  
    `(*) (void) osal_task_entry )`

For task entry point.

**12.242 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.*

### 12.242.1 Detailed Description

Declarations and prototypes for timebase abstraction

### 12.242.2 Typedef Documentation

#### 12.242.2.1 `OS_TimerSync_t` `typedef uint32 (* OS_TimerSync_t) (osal_id_t timer_id)`

Timer sync.

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

## 12.243 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.*

### 12.243.1 Detailed Description

Declarations and prototypes for timer abstraction (app callbacks)

### 12.243.2 Typedef Documentation

**12.243.2.1 OS\_TimerCallback\_t** `typedef void(* OS_TimerCallback_t) (osal_id_t timer_id)`  
Timer callback.

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

## 12.244 osal/src/os/inc/osapi-version.h File Reference

```
#include "common_types.h"
```

### Macros

- `#define OS_BUILD_NUMBER 0`  
*Development: Release name for current development cycle.*
- `#define OS_BUILD_BASELINE "v7.0.0"`  
*: Development: Code name for the current build*
- `#define OS_MAJOR_VERSION 7`  
*Major version number.*
- `#define OS_MINOR_VERSION 0`  
*Minor version number.*
- `#define OS_REVISION 0`  
*Revision version number. Value of 99 indicates a development version.*
- `#define OS_LAST_OFFICIAL "v7.0.0"`  
*Last official release.*
- `#define OS_MISSION_REV 0x0`  
*Mission revision.*
- `#define OS_STR_HELPER(x) #x`  
*Helper function to concatenate strings from integer.*
- `#define OS_STR(x) OS_STR_HELPER(x)`  
*Helper function to concatenate strings from integer.*
- `#define OS_VERSION OS_BUILD_BASELINE "+dev" OS_STR(OS_BUILD_NUMBER)`  
*Development Build Version Number.*
- `#define OSAL_API_VERSION ((OS_MAJOR_VERSION * 10000) + (OS_MINOR_VERSION * 100) + OS_REVISION)`  
*Combines the revision components into a single value.*
- `#define OS_CFG_MAX_VERSION_STR_LEN 256`  
*Max Version String length.*

### Functions

- `const char * OS_GetVersionString (void)`
- `const char * OS_GetVersionCodeName (void)`
- `void OS_GetVersionNumber (uint8 VersionNumbers[4])`  
*Obtain the OSAL numeric version number.*
- `uint32 OS_GetBuildNumber (void)`  
*Obtain the OSAL library numeric build number.*

### 12.244.1 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.

### 12.244.2 Macro Definition Documentation

#### 12.244.2.1 OS\_BUILD\_BASELINE `#define OS_BUILD_BASELINE "v7.0.0"`

Definition at line 38 of file osapi-version.h.

#### 12.244.2.2 OS\_BUILD\_CODENAME `#define OS_BUILD_CODENAME "Draco"`

: Development: Code name for the current build

Definition at line 40 of file osapi-version.h.

#### 12.244.2.3 OS\_BUILD\_DEV\_CYCLE `#define OS_BUILD_DEV_CYCLE "v7.0.0"`

Development: Release name for current development cycle.

Definition at line 39 of file osapi-version.h.

#### 12.244.2.4 OS\_BUILD\_NUMBER `#define OS_BUILD_NUMBER 0`

Definition at line 37 of file osapi-version.h.

#### 12.244.2.5 OS\_CFG\_MAX\_VERSION\_STR\_LEN `#define OS_CFG_MAX_VERSION_STR_LEN 256`

Max Version String length.

Maximum length that an OSAL version string can be.

Definition at line 154 of file osapi-version.h.

#### 12.244.2.6 OS\_LAST\_OFFICIAL `#define OS_LAST_OFFICIAL "v7.0.0"`

Last official release.

Definition at line 52 of file osapi-version.h.

#### 12.244.2.7 OS\_MAJOR\_VERSION `#define OS_MAJOR_VERSION 7`

Major version number.

Definition at line 45 of file osapi-version.h.

#### 12.244.2.8 OS\_MINOR\_VERSION `#define OS_MINOR_VERSION 0`

Minor version number.

Definition at line 46 of file osapi-version.h.

**12.244.2.9 OS\_MISSION\_REV** #define OS\_MISSION\_REV 0x0

Mission revision.

Reserved for mission use to denote patches/customizations as needed. Values 1-254 are reserved for mission use to denote patches/customizations as needed. NOTE: Reserving 0 and 0xFF for cFS open-source development use (pending resolution of nasa/cFS#440)

Definition at line 61 of file osapi-version.h.

**12.244.2.10 OS\_REVISION** #define OS\_REVISION 0

Revision version number. Value of 99 indicates a development version.

Definition at line 47 of file osapi-version.h.

**12.244.2.11 OS\_STR** #define OS\_STR(

x ) OS\_STR\_HELPER(x)

Helper function to concatenate strings from integer.

Definition at line 67 of file osapi-version.h.

**12.244.2.12 OS\_STR\_HELPER** #define OS\_STR\_HELPER(

x ) #x

Helper function to concatenate strings from integer.

Definition at line 66 of file osapi-version.h.

**12.244.2.13 OS\_VERSION** #define OS\_VERSION OS\_BUILD\_BASELINE "+dev" OS\_STR(OS\_BUILD\_NUMBER)

Development Build Version Number.

Baseline git tag + Number of commits since baseline.

Definition at line 72 of file osapi-version.h.

**12.244.2.14 OSAL\_API\_VERSION** #define OSAL\_API\_VERSION ((OS\_MAJOR\_VERSION \* 10000) + (OS\_MINOR\_VERSION \* 100) + OS\_REVISION)

Combines the revision components into a single value.

Applications can check against this number

e.g. "#if OSAL\_API\_VERSION >= 40100" would check if some feature added in OSAL 4.1 is present.

Definition at line 79 of file osapi-version.h.

## 12.244.3 Function Documentation

**12.244.3.1 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

**12.244.3.2 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.

**12.244.3.3 OS\_GetVersionNumber()** `void OS_GetVersionNumber ( uint8 VersionNumbers[4] )`

Obtain the OSAL numeric version number.

This retrieves the numeric OSAL version identifier as an array of 4 uint8 values.

The array of numeric values is in order of precedence: [0] = Major Number [1] = Minor Number [2] = Revision Number [3] = Mission Revision

The "Mission Revision" (last output) also indicates whether this is an official release, a patched release, or a development version. 0 indicates an official release 1-254 local patch level (reserved for mission use) 255 indicates a development build

**Parameters**

<code>out</code>	<code>VersionNumbers</code>	A fixed-size array to be filled with the version numbers
------------------	-----------------------------	--

**12.244.3.4 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.

## 12.245 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-rwlock.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"
```

#### 12.245.1 Detailed Description

Purpose: Contains functions prototype definitions and variables declarations for the OS Abstraction Layer, Core OS module

### 12.246 psp/fsw/inc/cfe\_psp.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "pspconfig.h"
#include "cfe_psp_cache_api.h"
#include "cfe_psp_cds_api.h"
#include "cfe_psp_eepromaccess_api.h"
#include "cfe_psp_error.h"
#include "cfe_psp_exception_api.h"
#include "cfe_psp_id_api.h"
#include "cfe_psp_memaccess_api.h"
#include "cfe_psp_memrange_api.h"
#include "cfe_psp_port_api.h"
#include "cfe_psp_ssr_api.h"
#include "cfe_psp_timetick_api.h"
#include "cfe_psp_version_api.h"
#include "cfe_psp_watchdog_api.h"
```

#### Macros

- #define PSP\_DEBUGLEV(l, ...)
- #define PSP\_DEBUG(...)

## Functions

- void **CFE\_PSP\_Main** (void)  
*PSP Entry Point to initialize the OSAL and start up the cFE.*

### 12.246.1 Macro Definition Documentation

**12.246.1.1 PSP\_DEBUG** #define PSP\_DEBUG(  
    ...  
)

Definition at line 78 of file cfe\_psp.h.

**12.246.1.2 PSP\_DEBUGLEV** #define PSP\_DEBUGLEV(  
    1,  
    ...  
)

Definition at line 77 of file cfe\_psp.h.

### 12.246.2 Function Documentation

**12.246.2.1 CFE\_PSP\_Main()** void CFE\_PSP\_Main (  
    void  
)

PSP Entry Point to initialize the OSAL and start up the cFE.

This is the entry point that the real-time OS calls to start our software. This routine will do any BSP/OS-specific setup, then call the entry point of the flight software (i.e. the cFE main entry point).

#### Note

The flight software (i.e. cFE) should not call this routine.

## 12.247 psp/fsw/inc/cfe\_psp\_cache\_api.h File Reference

```
#include "common_types.h"  
#include "osapi.h"  
#include "cfe_psp_error.h"
```

## Functions

- void **CFE\_PSPFlushCaches** (uint32 type, void \*address, uint32 size)  
*This is a BSP-specific cache flush routine.*

### 12.247.1 Function Documentation

**12.247.1.1 CFE\_PSPFlushCaches()** void CFE\_PSPFlushCaches (  
    uint32 type,  
    void \* address,  
    uint32 size )

This is a BSP-specific cache flush routine.

Provides a common interface to flush the processor caches. This routine is in the BSP because it is sometimes implemented in hardware and sometimes taken care of by the RTOS.

**Parameters**

in	<i>type</i>	
in	<i>address</i>	
in	<i>size</i>	

**12.248 psp/fsw/inc/cfe\_psp\_cds\_api.h File Reference**

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

**Functions**

- `int32 CFE_PSP_GetCDSSize (uint32 *SizeOfCDS)`  
*Fetches the size of the OS Critical Data Store area.*
- `int32 CFE_PSP_WriteToCDS (const void *PtrToDataToWrite, uint32 CDSOffset, uint32 NumBytes)`  
*Writes to the CDS Block.*
- `int32 CFE_PSP_ReadFromCDS (void *PtrToDataFromRead, uint32 CDSOffset, uint32 NumBytes)`  
*Reads from the CDS Block.*

**12.248.1 Function Documentation****12.248.1.1 CFE\_PSP\_GetCDSSize()** `int32 CFE_PSP_GetCDSSize (`  
`uint32 * SizeOfCDS )`

Fetches the size of the OS Critical Data Store area.

**Parameters**

out	<i>SizeOfCDS</i>	Pointer to the variable that will store the size of the CDS
-----	------------------	---

**Returns**

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error

**12.248.1.2 CFE\_PSP\_ReadFromCDS()** `int32 CFE_PSP_ReadFromCDS (`  
`void * PtrToDataFromRead,`  
`uint32 CDSOffset,`  
`uint32 NumBytes )`

Reads from the CDS Block.

**Parameters**

out	<i>PtrToDataFromRead</i>	Pointer to the location that will store the data to be read from the CDS
in	<i>CDSOffset</i>	CDS offset
in	<i>NumBytes</i>	Number of bytes to read

**Returns**

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error

**12.248.1.3 CFE\_PSP\_WriteToCDS()** `int32 CFE_PSP_WriteToCDS (`

```
    const void * PtrToDataToWrite,
    uint32 CDSOffset,
    uint32 NumBytes )
```

Writes to the CDS Block.

**Parameters**

in	<i>PtrToDataToWrite</i>	Pointer to the data that will be written to the CDS
in	<i>CDSOffset</i>	CDS offset
in	<i>NumBytes</i>	Number of bytes to write

**Returns**

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error

**12.249 psp/fsw/inc/cfe\_psp\_eepromaccess\_api.h File Reference**

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

**Functions**

- `int32 CFE_PSP_EepromWrite8 (cpuaddr MemoryAddress, uint8 ByteValue)`  
*Write one byte (ByteValue) to EEPROM address MemoryAddress.*
- `int32 CFE_PSP_EepromWrite16 (cpuaddr MemoryAddress, uint16 uint16Value)`  
*Write two bytes (uint16Value) to EEPROM address MemoryAddress.*
- `int32 CFE_PSP_EepromWrite32 (cpuaddr MemoryAddress, uint32 uint32Value)`  
*Write four bytes (uint32Value) to EEPROM address MemoryAddress.*
- `int32 CFE_PSP_EepromWriteEnable (uint32 Bank)`  
*Enable the EEPROM for write operation.*
- `int32 CFE_PSP_EepromWriteDisable (uint32 Bank)`  
*Disable the EEPROM from write operation.*
- `int32 CFE_PSP_EepromPowerUp (uint32 Bank)`  
*Power up the EEPROM.*
- `int32 CFE_PSP_EepromPowerDown (uint32 Bank)`  
*Power down the EEPROM.*

**12.249.1 Function Documentation****12.249.1.1 CFE\_PSP\_EepromPowerDown()** `int32 CFE_PSP_EepromPowerDown (`

```
    uint32 Bank )
```

Power down the EEPROM.

**Parameters**

in	<i>Bank</i>	The bank of EEPROM to power down
----	-------------	----------------------------------

**Return values**

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.249.1.2 CFE\_PSP\_EepromPowerUp()** `int32 CFE_PSP_EepromPowerUp ( uint32 Bank )`

Power up the EEPROM.

**Parameters**

in	<i>Bank</i>	The bank of EEPROM to power up
----	-------------	--------------------------------

**Return values**

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.249.1.3 CFE\_PSP\_EepromWrite16()** `int32 CFE_PSP_EepromWrite16 ( cpuaddr MemoryAddress, uint16 uint16Value )`

Write two bytes (`uint16Value`) to EEPROM address `MemoryAddress`.

**Parameters**

out	<i>MemoryAddress</i>	Memory address to write to
in	<i>uint16Value</i>	Value to write to memory

**Return values**

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_TIMEOUT</i>	write operation did not go through after a specific timeout.
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.249.1.4 CFE\_PSP\_EepromWrite32()** `int32 CFE_PSP_EepromWrite32 ( cpuaddr MemoryAddress, uint32 uint32Value )`

Write four bytes (uint32Value) to EEPROM address MemoryAddress.

#### Parameters

out	<i>MemoryAddress</i>	Memory address to write to
in	<i>uint32Value</i>	Value to write to memory

#### Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_TIMEOUT</i>	write operation did not go through after a specific timeout.
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

#### 12.249.1.5 CFE\_PSP\_EepromWrite8()

```
int32 CFE_PSP_EepromWrite8 (
    cpuaddr MemoryAddress,
    uint8 ByteValue )
```

Write one byte (ByteValue) to EEPROM address MemoryAddress.

#### Parameters

out	<i>MemoryAddress</i>	Memory address to write to
in	<i>ByteValue</i>	Value to write to memory

#### Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_TIMEOUT</i>	write operation did not go through after a specific timeout.
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

#### 12.249.1.6 CFE\_PSP\_EepromWriteDisable()

```
int32 CFE_PSP_EepromWriteDisable (
    uint32 Bank )
```

Disable the EEPROM from write operation.

#### Parameters

in	<i>Bank</i>	The bank of EEPROM to disable
----	-------------	-------------------------------

#### Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.249.1.7 CFE\_PSP\_EepromWriteEnable()** `int32 CFE_PSP_EepromWriteEnable (`  
`uint32 Bank )`

Enable the EEPROM for write operation.

#### Parameters

in	<i>Bank</i>	The bank of EEPROM to enable
----	-------------	------------------------------

#### Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

## 12.250 psp/fsw/inc/cfe\_psp\_endian.h File Reference

```
#include "common_types.h"
```

#### Functions

- `uint16 CFE_PSP_HtoBE16 (uint16 host_16bits)`
- `uint16 CFE_PSP_HtoLE16 (uint16 host_16bits)`
- `uint16 CFE_PSP_BE16toH (uint16 big_endian_16bits)`
- `uint16 CFE_PSP_LE16toH (uint16 little_endian_16bits)`
- `uint32 CFE_PSP_HtoBE32 (uint32 host_32bits)`
- `uint32 CFE_PSP_HtoLE32 (uint32 host_32bits)`
- `uint32 CFE_PSP_BE32toH (uint32 big_endian_32bits)`
- `uint32 CFE_PSP_LE32toH (uint32 little_endian_32bits)`
- `uint64 CFE_PSP_HtoBE64 (uint64 host_64bits)`
- `uint64 CFE_PSP_HtoLE64 (uint64 host_64bits)`
- `uint64 CFE_PSP_BE64toH (uint64 big_endian_64bits)`
- `uint64 CFE_PSP_LE64toH (uint64 little_endian_64bits)`
- `bool CFE_PSP_IsBigEndian (void)`
- `bool CFE_PSP_IsLittleEndian (void)`

### 12.250.1 Detailed Description

Map the PSP endian conversion routines to the system-provided endian.h file. This is done using static inline functions, such that the result should be optimized as much as possible.

### 12.250.2 Function Documentation

**12.250.2.1 CFE\_PSP\_BE16toH()** `uint16 CFE_PSP_BE16toH (`  
`uint16 big_endian_16bits )`

**12.250.2.2 CFE\_PSP\_BE32toH()** `uint32 CFE_PSP_BE32toH (`  
`uint32 big_endian_32bits )`

**12.250.2.3 CFE\_PSP\_BE64toH()** `uint64 CFE_PSP_BE64toH (`  
`uint64 big_endian_64bits )`

**12.250.2.4 CFE\_PSP\_HtoBE16()** `uint16 CFE_PSP_HtoBE16 (`  
`uint16 host_16bits )`

**12.250.2.5 CFE\_PSP\_HtoBE32()** `uint32 CFE_PSP_HtoBE32 (`  
`uint32 host_32bits )`

**12.250.2.6 CFE\_PSP\_HtoBE64()** `uint64 CFE_PSP_HtoBE64 (`  
`uint64 host_64bits )`

**12.250.2.7 CFE\_PSP\_HtoLE16()** `uint16 CFE_PSP_HtoLE16 (`  
`uint16 host_16bits )`

**12.250.2.8 CFE\_PSP\_HtoLE32()** `uint32 CFE_PSP_HtoLE32 (`  
`uint32 host_32bits )`

**12.250.2.9 CFE\_PSP\_HtoLE64()** `uint64 CFE_PSP_HtoLE64 (`  
`uint64 host_64bits )`

**12.250.2.10 CFE\_PSP\_IsBigEndian()** `bool CFE_PSP_IsBigEndian (`  
`void )`

**12.250.2.11 CFE\_PSP\_IsLittleEndian()** `bool CFE_PSP_IsLittleEndian (`  
`void )`

**12.250.2.12 CFE\_PSP\_LE16toH()** `uint16 CFE_PSP_LE16toH (`  
`uint16 little_endian_16bits )`

**12.250.2.13 CFE\_PSP\_LE32toH()** `uint32 CFE_PSP_LE32toH (`  
`uint32 little_endian_32bits )`

**12.250.2.14 CFE\_PSP\_LE64toH()** `uint64 CFE_PSP_LE64toH (`  
`uint64 little_endian_64bits )`

## 12.251 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.*

#### 12.251.1 Detailed Description

cFE PSP Error header

#### 12.251.2 Macro Definition Documentation

**12.251.2.1 CFE\_PSP\_ERROR** `#define CFE_PSP_ERROR (CFE_PSP_STATUS_C(-1))`  
Definition at line 67 of file cfe\_psp\_error.h.

**12.251.2.2 CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED** #define CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED (CFE\_PSP\_STATUS\_C (-3))  
Definition at line 69 of file cfe\_psp\_error.h.

**12.251.2.3 CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED** #define CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED (CFE\_PSP\_STATUS\_C (-27))  
Definition at line 78 of file cfe\_psp\_error.h.

**12.251.2.4 CFE\_PSP\_ERROR\_TIMEOUT** #define CFE\_PSP\_ERROR\_TIMEOUT (CFE\_PSP\_STATUS\_C (-4))  
Definition at line 70 of file cfe\_psp\_error.h.

**12.251.2.5 CFE\_PSP\_INVALID\_INT\_NUM** #define CFE\_PSP\_INVALID\_INT\_NUM (CFE\_PSP\_STATUS\_C (-5))  
Definition at line 71 of file cfe\_psp\_error.h.

**12.251.2.6 CFE\_PSP\_INVALID\_MEM\_ADDR** #define CFE\_PSP\_INVALID\_MEM\_ADDR (CFE\_PSP\_STATUS\_C (-21))  
Definition at line 72 of file cfe\_psp\_error.h.

**12.251.2.7 CFE\_PSP\_INVALID\_MEM\_ATTR** #define CFE\_PSP\_INVALID\_MEM\_ATTR (CFE\_PSP\_STATUS\_C (-26))  
Definition at line 77 of file cfe\_psp\_error.h.

**12.251.2.8 CFE\_PSP\_INVALID\_MEM\_RANGE** #define CFE\_PSP\_INVALID\_MEM\_RANGE (CFE\_PSP\_STATUS\_C (-23))  
Definition at line 74 of file cfe\_psp\_error.h.

**12.251.2.9 CFE\_PSP\_INVALID\_MEM\_SIZE** #define CFE\_PSP\_INVALID\_MEM\_SIZE (CFE\_PSP\_STATUS\_C (-25))  
Definition at line 76 of file cfe\_psp\_error.h.

**12.251.2.10 CFE\_PSP\_INVALID\_MEM\_TYPE** #define CFE\_PSP\_INVALID\_MEM\_TYPE (CFE\_PSP\_STATUS\_C (-22))  
Definition at line 73 of file cfe\_psp\_error.h.

**12.251.2.11 CFE\_PSP\_INVALID\_MEM\_WORDSIZE** #define CFE\_PSP\_INVALID\_MEM\_WORDSIZE (CFE\_PSP\_STATUS\_C (-24))  
Definition at line 75 of file cfe\_psp\_error.h.

**12.251.2.12 CFE\_PSP\_INVALID\_MODULE\_ID** #define CFE\_PSP\_INVALID\_MODULE\_ID (CFE\_PSP\_STATUS\_C (-29))  
Definition at line 80 of file cfe\_psp\_error.h.

**12.251.2.13 CFE\_PSP\_INVALID\_MODULE\_NAME** #define CFE\_PSP\_INVALID\_MODULE\_NAME (CFE\_PSP\_STATUS\_C (-28))  
Definition at line 79 of file cfe\_psp\_error.h.

**12.251.2.14 CFE\_PSP\_INVALID\_POINTER** #define CFE\_PSP\_INVALID\_POINTER (CFE\_PSP\_STATUS\_C (-2))  
Definition at line 68 of file cfe\_psp\_error.h.

**12.251.2.15 CFE\_PSP\_NO\_EXCEPTION\_DATA** #define CFE\_PSP\_NO\_EXCEPTION\_DATA (CFE\_PSP\_STATUS\_C(-30))  
Definition at line 81 of file cfe\_psp\_error.h.

**12.251.2.16 CFE\_PSP\_STATUS\_C** #define CFE\_PSP\_STATUS\_C(  
X ) ((CFE\_PSP\_Status\_t)(X))

PSP Status macro for literal.

Definition at line 37 of file cfe\_psp\_error.h.

**12.251.2.17 CFE\_PSP\_STATUS\_STRING\_LENGTH** #define CFE\_PSP\_STATUS\_STRING\_LENGTH 12

PSP Status converted to string length limit.

Used for sizing CFE\_PSP\_StatusString\_t intended for use in printing CFE\_PSP\_Status\_t values Sized for Id (LONG←\_MIN) including NULL

Definition at line 45 of file cfe\_psp\_error.h.

**12.251.2.18 CFE\_PSP\_SUCCESS** #define CFE\_PSP\_SUCCESS (CFE\_PSP\_STATUS\_C(0))

Definition at line 66 of file cfe\_psp\_error.h.

### 12.251.3 Typedef Documentation

**12.251.3.1 CFE\_PSP\_Status\_t** typedef int32 CFE\_PSP\_Status\_t

PSP Status type for readability and potentially type safety.

Definition at line 32 of file cfe\_psp\_error.h.

**12.251.3.2 CFE\_PSP\_StatusString\_t** typedef char CFE\_PSP\_StatusString\_t[CFE\_PSP\_STATUS\_STRING\_LENGTH]

For the CFE\_PSP\_StatusToString() function, to ensure everyone is making an array of the same length.

Definition at line 51 of file cfe\_psp\_error.h.

### 12.251.4 Function Documentation

**12.251.4.1 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

**12.252 psp/fsw/inc/cfe\_psp\_exception\_api.h File Reference**

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

**Functions**

- void [CFE\\_PSP\\_AttachExceptions](#) (void)  
*Sets up the exception environment for the chosen platform.*
- void [CFE\\_PSP\\_SetDefaultExceptionEnvironment](#) (void)  
*Defines the CPU and FPU exceptions that are enabled for each cFE Task/App.*
- uint32 [CFE\\_PSP\\_Exception\\_GetCount](#) (void)  
*Returns the unread exception count.*
- int32 [CFE\\_PSP\\_Exception\\_GetSummary](#) (uint32 \*ContextLogId, [osal\\_id\\_t](#) \*TaskId, char \*ReasonBuf, uint32 ReasonSize)  
*Retrieves a summary of an exception log entry.*
- int32 [CFE\\_PSP\\_Exception\\_CopyContext](#) (uint32 ContextLogId, void \*ContextBuf, uint32 ContextSize)  
*Retrieves exception log entry context information.*

**12.252.1 Function Documentation****12.252.1.1 CFE\_PSP\_AttachExceptions()** `void CFE_PSP_AttachExceptions (`  
 `void )`

Sets up the exception environment for the chosen platform.

On a board, this can be configured to look at a debug flag or switch in order to keep the standard OS exception handlers, rather than restarting the system.

**12.252.1.2 CFE\_PSP\_Exception\_CopyContext()** `int32 CFE_PSP_Exception_CopyContext (`  
 `uint32 ContextLogId,`  
 `void * ContextBuf,`  
 `uint32 ContextSize )`

Retrieves exception log entry context information.

**Parameters**

in	<i>ContextLogId</i>	ID of the exception log entry to copy
out	<i>ContextBuf</i>	Pointer to the buffer where the context information is to be copied to
in	<i>ContextSize</i>	Maximum size of context information data to copy

**Returns**

Size of the copied data

**Return values**

<code>CFE_PSP_NO_EXCEPTION_DATA</code>	if data has expired from the memory log
--	---

**12.252.1.3 CFE\_PSP\_Exception\_GetCount()** `uint32 CFE_PSP_Exception_GetCount ( void )`

Returns the unread exception count.

**Returns**

The unread exception count

**12.252.1.4 CFE\_PSP\_Exception\_GetSummary()** `int32 CFE_PSP_Exception_GetSummary (`

```
    uint32 * ContextLogId,
    osal_id_t * TaskId,
    char * ReasonBuf,
    uint32 ReasonSize )
```

Retrieves a summary of an exception log entry.

**Note**

This function returns CFE\_PSP\_SUCCESS to indicate that an entry was popped from the queue. This doesn't necessarily mean that the output fields have valid data, but it does mean they are initialized to something.

**Parameters**

<code>in</code>	<code>ContextLogId</code>	ID of the exception log entry to get a summary for
<code>in, out</code>	<code>TaskId</code>	Pointer to the TaskID buffer
<code>out</code>	<code>ReasonBuf</code>	Pointer to the buffer that will store the exception summary string
<code>in</code>	<code>ReasonSize</code>	Maximum size of the summary string to retrieve

**Return values**

<code>CFE_PSP_SUCCESS</code>	on success (see note above)
<code>CFE_PSP_NO_EXCEPTION_DATA</code>	if no context available for reading

**12.252.1.5 CFE\_PSP\_SetDefaultExceptionEnvironment()** `void CFE_PSP_SetDefaultExceptionEnvironment ( void )`

Defines the CPU and FPU exceptions that are enabled for each cFE Task/App.

This function sets a default exception environment that can be used

**Note**

The exception environment is local to each task. Therefore, this must be Called for each task that wants to do floating point and catch exceptions.

## 12.253 psp/fsw/inc/cfe\_psp\_id\_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

### Functions

- `uint32 CFE_PSP_GetProcessorId (void)`  
*Returns the CPU ID as defined by the specific board and BSP.*
- `uint32 CFE_PSP_GetSpacecraftId (void)`  
*Returns the Spacecraft ID (if any)*
- `const char * CFE_PSP_GetProcessorName (void)`  
*Returns the processor name.*

### 12.253.1 Function Documentation

#### 12.253.1.1 CFE\_PSP\_GetProcessorId() `uint32 CFE_PSP_GetProcessorId (`     `void )`

Returns the CPU ID as defined by the specific board and BSP.

##### Returns

The processor ID

#### 12.253.1.2 CFE\_PSP\_GetProcessorName() `const char* CFE_PSP_GetProcessorName (`     `void )`

Returns the processor name.

##### Returns

The processor name

#### 12.253.1.3 CFE\_PSP\_GetSpacecraftId() `uint32 CFE_PSP_GetSpacecraftId (`     `void )`

Returns the Spacecraft ID (if any)

##### Returns

The Spacecraft ID

## 12.254 psp/fsw/inc/cfe\_psp\_memaccess\_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

## Functions

- `int32 CFE_PSP_MemRead8 (cpuaddr MemoryAddress, uint8 *ByteValue)`  
*Read one byte of memory.*
- `int32 CFE_PSP_MemWrite8 (cpuaddr MemoryAddress, uint8 ByteValue)`  
*Write one byte of memory.*
- `int32 CFE_PSP_MemRead16 (cpuaddr MemoryAddress, uint16 *uint16Value)`  
*Read 2 bytes of memory.*
- `int32 CFE_PSP_MemWrite16 (cpuaddr MemoryAddress, uint16 uint16Value)`  
*Write 2 bytes of memory.*
- `int32 CFE_PSP_MemRead32 (cpuaddr MemoryAddress, uint32 *uint32Value)`  
*Read 4 bytes of memory.*
- `int32 CFE_PSP_MemWrite32 (cpuaddr MemoryAddress, uint32 uint32Value)`  
*Write 4 bytes of memory.*
- `int32 CFE_PSP_MemCpy (void *dest, const void *src, uint32 n)`  
*Copy 'n' bytes from 'src' to 'dest'.*
- `int32 CFE_PSP_MemSet (void *dest, uint8 value, uint32 n)`  
*Copy 'n' bytes of value 'value' to 'dest'.*

### 12.254.1 Function Documentation

**12.254.1.1 CFE\_PSP\_MemCpy()** `int32 CFE_PSP_MemCpy (`  
    `void * dest,`  
    `const void * src,`  
    `uint32 n )`

Copy 'n' bytes from 'src' to 'dest'.

Copies 'n' bytes from memory address pointed to by 'src' to memory address pointed to by 'dest'.

#### Note

For now we are using the standard C library call 'memcpy' but if we find we need to make it more efficient then we'll implement it in assembly.

#### Parameters

<code>out</code>	<code>dest</code>	Pointer to the destination address to copy to
<code>in</code>	<code>src</code>	Pointer to the address to copy from
<code>in</code>	<code>n</code>	Number of bytes to copy

#### Returns

Always returns CFE\_PSP\_SUCCESS

**12.254.1.2 CFE\_PSP\_MemRead16()** `int32 CFE_PSP_MemRead16 (`  
    `cpuaddr MemoryAddress,`  
    `uint16 * uint16Value )`

Read 2 bytes of memory.

**Parameters**

in	<i>MemoryAddress</i>	Address to be read
out	<i>uint16Value</i>	The address content will be copied to the location pointed to by this argument

**Return values**

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.254.1.3 CFE\_PSP\_MemRead32()** `int32 CFE_PSP_MemRead32 (`

```
    cpuaddr MemoryAddress,
    uint32 * uint32Value )
```

Read 4 bytes of memory.

**Parameters**

in	<i>MemoryAddress</i>	Address to be read
out	<i>uint32Value</i>	The address content will be copied to the location pointed to by this argument

**Return values**

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.254.1.4 CFE\_PSP\_MemRead8()** `int32 CFE_PSP_MemRead8 (`

```
    cpuaddr MemoryAddress,
    uint8 * ByteValue )
```

Read one byte of memory.

**Parameters**

in	<i>MemoryAddress</i>	Address to be read
out	<i>ByteValue</i>	The address content will be copied to the location pointed to by this argument

**Returns**

Always returns CFE\_PSP\_SUCCESS (if implemented)

**Return values**

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

```
12.254.1.5 CFE_PSP_MemSet() int32 CFE_PSP_MemSet (
    void * dest,
    uint8 value,
    uint32 n )
```

Copy 'n' bytes of value 'value' to 'dest'.

Copies 'n' number of bytes of value 'value' to memory address pointed to by 'dest'.

#### Note

For now we are using the standard C library call 'memset' but if we find we need to make it more efficient then we'll implement it in assembly.

#### Parameters

out	<i>dest</i>	Pointer to the destination address to copy to
in	<i>value</i>	Value to set
in	<i>n</i>	Number of bytes to copy

#### Returns

Always returns CFE\_PSP\_SUCCESS

```
12.254.1.6 CFE_PSP_MemWrite16() int32 CFE_PSP_MemWrite16 (
    cpuaddr MemoryAddress,
    uint16 uint16Value )
```

Write 2 bytes of memory.

#### Parameters

out	<i>MemoryAddress</i>	Address to be written to
in	<i>uint16Value</i>	The content pointed to by this argument will be copied to the address

#### Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

```
12.254.1.7 CFE_PSP_MemWrite32() int32 CFE_PSP_MemWrite32 (
    cpuaddr MemoryAddress,
    uint32 uint32Value )
```

Write 4 bytes of memory.

#### Parameters

out	<i>MemoryAddress</i>	Address to be written to
-----	----------------------	--------------------------

**Parameters**

in	<i>uint32Value</i>	The content pointed to by this argument will be copied to the address
----	--------------------	---

**Return values**

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.254.1.8 CFE\_PSP\_MemWrite8()** `int32 CFE_PSP_MemWrite8 (`  
`cpuaddr MemoryAddress,`  
`uint8 ByteValue )`

Write one byte of memory.

**Parameters**

out	<i>MemoryAddress</i>	Address to be written to
in	<i>ByteValue</i>	The content pointed to by this argument will be copied to the address

**Returns**

Always returns CFE\_PSP\_SUCCESS (if implemented)

**Return values**

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

**12.255 psp/fsw/inc/cfe\_psp\_memrange\_api.h File Reference**

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

**Macros**

- #define CFE\_PSP\_MEM\_RAM 1
- #define CFE\_PSP\_MEM\_EEPROM 2
- #define CFE\_PSP\_MEM\_ANY 3
- #define CFE\_PSP\_MEM\_INVALID 4
- #define CFE\_PSP\_MEM\_ATTR\_WRITE 0x01
- #define CFE\_PSP\_MEM\_ATTR\_READ 0x02
- #define CFE\_PSP\_MEM\_ATTR\_READWRITE 0x03
- #define CFE\_PSP\_MEM\_SIZE\_BYTE 0x01
- #define CFE\_PSP\_MEM\_SIZE\_WORD 0x02
- #define CFE\_PSP\_MEM\_SIZE\_DWORD 0x04

## Functions

- `int32 CFE_PSP_GetResetArea (cpuaddr *PtrToResetArea, uint32 *SizeOfResetArea)`  
*Returns the location and size of the ES Reset information area.*
- `int32 CFE_PSP.GetUserReservedArea (cpuaddr *PtrToUserArea, uint32 *SizeOfUserArea)`  
*Returns the location and size of the memory used for the cFE user-reserved area.*
- `int32 CFE_PSP_GetVolatileDiskMem (cpuaddr *PtrToVolDisk, uint32 *SizeOfVolDisk)`  
*Returns the location and size of the memory used for the cFE volatile disk.*
- `int32 CFE_PSP_GetKernelTextSegmentInfo (cpuaddr *PtrToKernelSegment, uint32 *SizeOfKernelSegment)`  
*Returns the location and size of the kernel memory.*
- `int32 CFE_PSP_GetCFETextSegmentInfo (cpuaddr *PtrToCFESegment, uint32 *SizeOfCFESegment)`  
*Returns the location and size of the kernel memory.*
- `int32 CFE_PSP_MemValidateRange (cpuaddr Address, size_t Size, uint32 MemoryType)`  
*Validates the memory range and type using the global CFE\_PSP\_MemoryTable.*
- `uint32 CFE_PSP_MemRanges (void)`  
*Returns the number of memory ranges in the CFE\_PSP\_MemoryTable.*
- `int32 CFE_PSP_MemRangeSet (uint32 RangeNum, uint32 MemoryType, cpuaddr StartAddr, size_t Size, size_t WordSize, uint32 Attributes)`  
*Populates one of the records in the CFE\_PSP\_MemoryTable.*
- `int32 CFE_PSP_MemRangeGet (uint32 RangeNum, uint32 *MemoryType, cpuaddr *StartAddr, size_t *Size, size_t *WordSize, uint32 *Attributes)`  
*Retrieves one of the records in the CFE\_PSP\_MemoryTable.*

### 12.255.1 Macro Definition Documentation

**12.255.1.1 CFE\_PSP\_MEM\_ANY** #define CFE\_PSP\_MEM\_ANY 3  
Definition at line 53 of file cfe\_psp\_memrange\_api.h.

**12.255.1.2 CFE\_PSP\_MEM\_ATTR\_READ** #define CFE\_PSP\_MEM\_ATTR\_READ 0x02  
Definition at line 60 of file cfe\_psp\_memrange\_api.h.

**12.255.1.3 CFE\_PSP\_MEM\_ATTR\_READWRITE** #define CFE\_PSP\_MEM\_ATTR\_READWRITE 0x03  
Definition at line 61 of file cfe\_psp\_memrange\_api.h.

**12.255.1.4 CFE\_PSP\_MEM\_ATTR\_WRITE** #define CFE\_PSP\_MEM\_ATTR\_WRITE 0x01  
Definition at line 59 of file cfe\_psp\_memrange\_api.h.

**12.255.1.5 CFE\_PSP\_MEM\_EEPROM** #define CFE\_PSP\_MEM\_EEPROM 2  
Definition at line 52 of file cfe\_psp\_memrange\_api.h.

**12.255.1.6 CFE\_PSP\_MEM\_INVALID** #define CFE\_PSP\_MEM\_INVALID 4  
Definition at line 54 of file cfe\_psp\_memrange\_api.h.

**12.255.1.7 CFE\_PSP\_MEM\_RAM** #define CFE\_PSP\_MEM\_RAM 1  
 Definition at line 51 of file cfe\_psp\_memrange\_api.h.

**12.255.1.8 CFE\_PSP\_MEM\_SIZE\_BYTE** #define CFE\_PSP\_MEM\_SIZE\_BYTE 0x01  
 Definition at line 66 of file cfe\_psp\_memrange\_api.h.

**12.255.1.9 CFE\_PSP\_MEM\_SIZE\_DWORD** #define CFE\_PSP\_MEM\_SIZE\_DWORD 0x04  
 Definition at line 68 of file cfe\_psp\_memrange\_api.h.

**12.255.1.10 CFE\_PSP\_MEM\_SIZE\_WORD** #define CFE\_PSP\_MEM\_SIZE\_WORD 0x02  
 Definition at line 67 of file cfe\_psp\_memrange\_api.h.

## 12.255.2 Function Documentation

**12.255.2.1 CFE\_PSP\_GetCFETextSegmentInfo()** int32 CFE\_PSP\_GetCFETextSegmentInfo ( cpuaddr \* PtrToCFEsegment,  
                           uint32 \* SizeOfCFEsegment )

Returns the location and size of the kernel memory.

This function returns the start and end address of the CFE text segment. It may not be implemented on all architectures.

### Parameters

out	<i>PtrToCFEsegment</i>	Pointer to the variable that will store the location of the cFE text segment
out	<i>SizeOfCFEsegment</i>	Pointer to the variable that will store the size of the cFE text segment

### Returns

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error

**12.255.2.2 CFE\_PSP\_GetKernelTextSegmentInfo()** int32 CFE\_PSP\_GetKernelTextSegmentInfo ( cpuaddr \* PtrToKernelSegment,  
                           uint32 \* SizeOfKernelSegment )

Returns the location and size of the kernel memory.

This function returns the start and end address of the kernel text segment. It may not be implemented on all architectures.

### Parameters

out	<i>PtrToKernelSegment</i>	Pointer to the variable that will store the location of the kernel text segment
out	<i>SizeOfKernelSegment</i>	Pointer to the variable that will store the size of the kernel text segment

**Returns**

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error or CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED if not implemented

**12.255.2.3 CFE\_PSP\_GetResetArea()** `int32 CFE_PSP_GetResetArea (`  
 `cpuaddr * PtrToResetArea,`  
 `uint32 * SizeOfResetArea )`

Returns the location and size of the ES Reset information area.

This area is preserved during a processor reset and is used to store the ER Log, System Log and reset-related variables

**Parameters**

<code>out</code>	<code>PtrToResetArea</code>	Pointer to the variable that will store the location of the reset area
<code>out</code>	<code>SizeOfResetArea</code>	Pointer to the variable that will store the reset area size

**Returns**

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error

**12.255.2.4 CFE\_PSP.GetUserReservedArea()** `int32 CFE_PSP.GetUserReservedArea (`  
 `cpuaddr * PtrToUserArea,`  
 `uint32 * SizeOfUserArea )`

Returns the location and size of the memory used for the cFE user-reserved area.

**Parameters**

<code>out</code>	<code>PtrToUserArea</code>	Pointer to the variable that will store the location of the user-reserved area
<code>out</code>	<code>SizeOfUserArea</code>	Pointer to the variable that will store the size of the user-reserved area

**Returns**

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error

**12.255.2.5 CFE\_PSP\_GetVolatileDiskMem()** `int32 CFE_PSP_GetVolatileDiskMem (`  
 `cpuaddr * PtrToVolDisk,`  
 `uint32 * SizeOfVolDisk )`

Returns the location and size of the memory used for the cFE volatile disk.

**Parameters**

<code>out</code>	<code>PtrToVolDisk</code>	Pointer to the variable that will store the location of the cFE volatile disk
<code>out</code>	<code>SizeOfVolDisk</code>	Pointer to the variable that will store the size of the cFE volatile disk

**Returns**

0 (OS\_SUCCESS or CFE\_PSP\_SUCCESS) on success, -1 (OS\_ERROR or CFE\_PSP\_ERROR) on error

```
12.255.2.6 CFE_PSP_MemRangeGet() int32 CFE_PSP_MemRangeGet (
    uint32 RangeNum,
    uint32 * MemoryType,
    cpuaddr * StartAddr,
    size_t * Size,
    size_t * WordSize,
    uint32 * Attributes )
```

Retrieves one of the records in the CFE\_PSP\_MemoryTable.

#### Note

Because the table is fixed size, the entries are accessed by using the integer index.

#### Parameters

in	<i>RangeNum</i>	A 32-bit integer (starting with 0) specifying the MemoryTable entry.
out	<i>MemoryType</i>	A pointer to the 32-bit integer where the Memory Type is stored. Any defined CFE_PSP_MEM_* enumeration can be specified
out	<i>StartAddr</i>	Pointer to the 32-bit integer where the 32-bit starting address of the memory range is stored.
out	<i>Size</i>	A pointer to the 32-bit integer where the 32-bit size of the memory range is stored.
out	<i>WordSize</i>	A pointer to the 32-bit integer where the minimum addressable size of the range: (CFE_PSP_MEM_SIZE_BYTE, CFE_PSP_MEM_SIZE_WORD, CFE_PSP_MEM_SIZE_DWORD)
out	<i>Attributes</i>	The attributes of the Memory Range: (CFE_PSP_MEM_ATTR_WRITE, CFE_PSP_MEM_ATTR_READ, CFE_PSP_MEM_ATTR_READWRITE)

#### Return values

<i>CFE_PSP_SUCCESS</i>	Memory range returned successfully.
<i>CFE_PSP_INVALID_POINTER</i>	Parameter error
<i>CFE_PSP_INVALID_MEM_RANGE</i>	The index into the table is invalid

```
12.255.2.7 CFE_PSP_MemRanges() uint32 CFE_PSP_MemRanges (
    void )
```

Returns the number of memory ranges in the CFE\_PSP\_MemoryTable.

#### Returns

Positive integer number of entries in the memory range table

```
12.255.2.8 CFE_PSP_MemRangeSet() int32 CFE_PSP_MemRangeSet (
    uint32 RangeNum,
    uint32 MemoryType,
    cpuaddr StartAddr,
    size_t Size,
    size_t WordSize,
    uint32 Attributes )
```

Populates one of the records in the CFE\_PSP\_MemoryTable.

**Note**

Because the table is fixed size, the entries are set by using the integer index. No validation is done with the address or size.

**Parameters**

in	<i>RangeNum</i>	A 32-bit integer (starting with 0) specifying the MemoryTable entry.
in	<i>MemoryType</i>	The memory type to validate, including but not limited to: CFE_PSP_MEM_RAM, CFE_PSP_MEM_EEPROM, or CFE_PSP_MEM_ANY Any defined CFE_PSP_MEM_* enumeration can be specified
in	<i>StartAddr</i>	A 32-bit starting address of the memory range
in	<i>Size</i>	A 32-bit size of the memory range (Address + Size = End Address)
in	<i>WordSize</i>	The minimum addressable size of the range: (CFE_PSP_MEM_SIZE_BYTE, CFE_PSP_MEM_SIZE_WORD, CFE_PSP_MEM_SIZE_DWORD)
in	<i>Attributes</i>	The attributes of the Memory Range: (CFE_PSP_MEM_ATTR_WRITE, CFE_PSP_MEM_ATTR_READ, CFE_PSP_MEM_ATTR_READWRITE)

**Return values**

<i>CFE_PSP_SUCCESS</i>	Memory range set successfully.
<i>CFE_PSP_INVALID_MEM_RANGE</i>	The index into the table is invalid
<i>CFE_PSP_INVALID_MEM_ADDR</i>	Starting address is not valid
<i>CFE_PSP_INVALID_MEM_TYPE</i>	Memory type associated with the range does not match the passed-in type.
<i>CFE_PSP_INVALID_MEM_WORDSIZE</i>	The WordSize parameter is not one of the predefined types.
<i>CFE_PSP_INVALID_MEM_ATTR</i>	The Attributes parameter is not one of the predefined types.
<i>OP_INVALID_MEM_SIZE</i>	The Memory range associated with the address is not large enough to contain Address + Size.

**12.255.2.9 CFE\_PSP\_MemValidateRange()**

```
int32 CFE_PSP_MemValidateRange (
    cpuaddr Address,
    size_t Size,
    uint32 MemoryType )
```

Validates the memory range and type using the global CFE\_PSP\_MemoryTable.

**Parameters**

in	<i>Address</i>	A 32-bit starting address of the memory range
in	<i>Size</i>	A 32-bit size of the memory range (Address + Size = End Address)
in	<i>MemoryType</i>	The memory type to validate, including but not limited to: CFE_PSP_MEM_RAM, CFE_PSP_MEM_EEPROM, or CFE_PSP_MEM_ANY Any defined CFE_PSP_MEM_* enumeration can be specified

**Return values**

<i>CFE_PSP_SUCCESS</i>	Memory range and type information is valid and can be used.
------------------------	---

#### Return values

<code>CFE_PSP_INVALID_MEM_ADDR</code>	Starting address is not valid
<code>CFE_PSP_INVALID_MEM_TYPE</code>	Memory type associated with the range does not match the passed-in type.
<code>CFE_PSP_INVALID_MEM_RANGE</code>	The Memory range associated with the address is not large enough to contain Address + Size.

## 12.256 psp/fsw/inc/cfe\_psp\_port\_api.h File Reference

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

#### Functions

- `int32 CFE_PSP_PortRead8 (cpuaddr PortAddress, uint8 *ByteValue)`  
*Read one byte of memory.*
- `int32 CFE_PSP_PortWrite8 (cpuaddr PortAddress, uint8 ByteValue)`  
*Write one byte of memory.*
- `int32 CFE_PSP_PortRead16 (cpuaddr PortAddress, uint16 *uint16Value)`  
*Read 2 bytes of memory.*
- `int32 CFE_PSP_PortWrite16 (cpuaddr PortAddress, uint16 uint16Value)`  
*Write 2 bytes of memory.*
- `int32 CFE_PSP_PortRead32 (cpuaddr PortAddress, uint32 *uint32Value)`  
*Read 4 bytes of memory.*
- `int32 CFE_PSP_PortWrite32 (cpuaddr PortAddress, uint32 uint32Value)`  
*Write 4 bytes of memory.*

### 12.256.1 Function Documentation

**12.256.1.1 CFE\_PSP\_PortRead16()** `int32 CFE_PSP_PortRead16 (`  
`cpuaddr PortAddress,`  
`uint16 * uint16Value )`

Read 2 bytes of memory.

#### Parameters

<code>in</code>	<code>PortAddress</code>	Address to be read
<code>out</code>	<code>uint16Value</code>	The address content will be copied to the location pointed to by this argument

#### Return values

<code>CFE_PSP_SUCCESS</code>	on success
<code>CFE_PSP_ERROR_ADDRESS_MISALIGNED</code>	if the address is not aligned to a 16-bit addressing scheme.
<code>CFE_PSP_ERROR_NOT_IMPLEMENTED</code>	if not implemented

```
12.256.1.2 CFE_PSP_PortRead32() int32 CFE_PSP_PortRead32 (
    cpuaddr PortAddress,
    uint32 * uint32Value )
```

Read 4 bytes of memory.

#### Parameters

in	<i>PortAddress</i>	Address to be read
out	<i>uint32Value</i>	The address content will be copied to the location pointed to by this argument

#### Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

```
12.256.1.3 CFE_PSP_PortRead8() int32 CFE_PSP_PortRead8 (
    cpuaddr PortAddress,
    uint8 * ByteValue )
```

Read one byte of memory.

#### Parameters

in	<i>PortAddress</i>	Address to be read
out	<i>ByteValue</i>	The address content will be copied to the location pointed to by this argument

#### Returns

Always returns *CFE\_PSP\_SUCCESS* (if implemented)

#### Return values

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

```
12.256.1.4 CFE_PSP_PortWrite16() int32 CFE_PSP_PortWrite16 (
    cpuaddr PortAddress,
    uint16 uint16Value )
```

Write 2 bytes of memory.

#### Parameters

out	<i>PortAddress</i>	Address to be written to
in	<i>uint16Value</i>	the content pointed to by this argument will be copied to the address

## Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.256.1.5 CFE\_PSP\_PortWrite32()** `int32 CFE_PSP_PortWrite32 (`  
 `cpuaddr PortAddress,`  
 `uint32 uint32Value )`

Write 4 bytes of memory.

## Parameters

<i>out</i>	<i>PortAddress</i>	Address to be written to
<i>in</i>	<i>uint32Value</i>	The content pointed to by this argument will be copied to the address

## Return values

<i>CFE_PSP_SUCCESS</i>	on success
<i>CFE_PSP_ERROR_ADDRESS_MISALIGNED</i>	if the address is not aligned to a 16-bit addressing scheme.
<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented

**12.256.1.6 CFE\_PSP\_PortWrite8()** `int32 CFE_PSP_PortWrite8 (`  
 `cpuaddr PortAddress,`  
 `uint8 ByteValue )`

Write one byte of memory.

## Parameters

<i>out</i>	<i>PortAddress</i>	Address to be written to
<i>in</i>	<i>ByteValue</i>	The content pointed to by this argument will be copied to the address

## Returns

Always returns CFE\_PSP\_SUCCESS (if implemented)

## Return values

<i>CFE_PSP_ERROR_NOT_IMPLEMENTED</i>	if not implemented
--------------------------------------	--------------------

**12.257 psp/fsw/inc/cfe\_psp\_ssr\_api.h File Reference**

```
#include "common_types.h"
#include "osapi.h"
```

```
#include "cfe_psp_error.h"
```

## Functions

- `int32 CFE_PSP_InitSSR (uint32 bus, uint32 device, char *DeviceName)`

*Initializes the Solid State recorder memory for a particular platform.*

### 12.257.1 Function Documentation

#### 12.257.1.1 CFE\_PSP\_InitSSR() `int32 CFE_PSP_InitSSR (`

```
    uint32 bus,  
    uint32 device,  
    char * DeviceName )
```

Initializes the Solid State recorder memory for a particular platform.

#### Note

For the MCP750, this simply initializes the Hard Disk device.

#### Parameters

in	<i>bus</i>	
in	<i>device</i>	
in	<i>DeviceName</i>	

#### Return values

<code>CFE_PSP_SUCCESS</code>	on success
<code>CFE_PSP_ERROR</code>	on error

## 12.258 psp/fsw/inc/cfe\_psp\_timertick\_api.h File Reference

```
#include "common_types.h"  
#include "osapi.h"  
#include "cfe_psp_error.h"
```

#### Macros

- `#define CFE_PSP_SOFT_TIMEBASE_NAME "cFS-Master"`

*The name of the software/RTOS timebase for general system timers.*

## Functions

- `void CFE_PSP_GetTime (OS_time_t *LocalTime)`

*Sample/Read a monotonic platform clock with normalization.*

- `uint32 CFE_PSP_GetTimerTicksPerSecond (void)`

- `uint32 CFE_PSP_GetTimerLow32Rollover (void)`

- void [CFE\\_PSP\\_Get\\_Timebase](#) (uint32 \*Tbu, uint32 \*Tbl)  
*Sample/Read a monotonic platform clock without normalization.*

### 12.258.1 Macro Definition Documentation

#### 12.258.1.1 CFE\_PSP\_SOFT\_TIMEBASE\_NAME #define CFE\_PSP\_SOFT\_TIMEBASE\_NAME "cFS-Master"

The name of the software/RTOS timebase for general system timers.

This name may be referred to by CFE TIME and/or SCH when setting up its own timers.

Definition at line 53 of file cfe\_psp\_timertick\_api.h.

### 12.258.2 Function Documentation

#### 12.258.2.1 CFE\_PSP\_Get\_Timebase() void CFE\_PSP\_Get\_Timebase (

```
    uint32 * Tbu,
    uint32 * Tbl )
```

Sample/Read a monotonic platform clock without normalization.

Provides a common interface to system timebase. This routine is in the BSP because it is sometimes implemented in hardware and sometimes taken care of by the RTOS.

This is defined as a free-running, monotonically-increasing tick counter. The epoch is not defined, but typically is the system boot time, and the value increases indefinitely as the system runs. The tick period/rate is also not defined.

Rollover events - where the range of representable values is exceeded - are theoretically possible, but would take many years of continuous uptime to occur (typically hundreds of years, if not thousands). System designers should ensure that the actual tick rate and resulting timebase range is sufficiently large to ensure that rollover is not a concern.

#### Note

This is a "raw" value from the underlying platform with minimal/no conversions or normalization applied. Neither the epoch nor the resolution of this tick counter is specified, and it may vary from platform to platform. Use the [CFE\\_PSP\\_GetTime\(\)](#) function to sample the timebase and also convert the units into a normalized/more consistent form.

#### See also

[CFE\\_PSP\\_GetTime\(\)](#)

#### Parameters

out	Tbu	Buffer to hold the upper 32 bits of a 64-bit tick counter
out	Tbl	Buffer to hold the lower 32 bits of a 64-bit tick counter

#### 12.258.2.2 CFE\_PSP\_GetTime() void CFE\_PSP\_GetTime (

```
    OS_time_t * LocalTime )
```

Sample/Read a monotonic platform clock with normalization.

Outputs an [OS\\_time\\_t](#) value indicating the time elapsed since an epoch. The epoch is not defined, but typically represents the system boot time. The value increases continuously over time and cannot be reset by software.

This is similar to the [CFE\\_PSP\\_Get\\_Timebase\(\)](#), but additionally it normalizes the output value to an [OS\\_time\\_t](#), thereby providing consistent units to the calling application. Any OSAL-provided routine that accepts [OS\\_time\\_t](#) inputs may be

used to convert this value into other standardized time units.

#### Note

This should refer to the same time domain as [CFE\\_PSP\\_Get\\_Timebase\(\)](#), the primary difference being the format and units of the output value.

#### See also

[CFE\\_PSP\\_Get\\_Timebase\(\)](#)

#### Parameters

<code>out</code>	<code>LocalTime</code>	Value of PSP tick counter as <a href="#">OS_time_t</a>
------------------	------------------------	--

### **12.258.2.3 CFE\_PSP\_GetTimerLow32Rollover()** `uint32 CFE_PSP_GetTimerLow32Rollover ( void )`

Provides the number that the least significant 32 bits of the 64-bit time stamp returned by CFE\_PSP\_Get\_Timebase rolls over. If the lower 32 bits rolls at 1 second, then the CFE\_PSP\_TIMER\_LOW32\_ROLLOVER will be 1000000. If the lower 32 bits rolls at its maximum value ( $2^{32}$ ) then CFE\_PSP\_TIMER\_LOW32\_ROLLOVER will be 0.

#### Returns

The number that the least significant 32 bits of the 64-bit time stamp returned by CFE\_PSP\_Get\_Timebase rolls over.

### **12.258.2.4 CFE\_PSP\_GetTimerTicksPerSecond()** `uint32 CFE_PSP_GetTimerTicksPerSecond ( void )`

Provides the resolution of the least significant 32 bits of the 64-bit time stamp returned by CFE\_PSP\_Get\_Timebase in timer ticks per second. The timer resolution for accuracy should not be any slower than 1000000 ticks per second or 1 us (microsecond) per tick

#### Returns

The number of timer ticks per second of the time stamp returned by CFE\_PSP\_Get\_Timebase

## **12.259 psp/fsw/inc/cfe\_psp\_version\_api.h File Reference**

```
#include "common_types.h"
#include "osapi.h"
#include "cfe_psp_error.h"
```

### Functions

- const char \* [CFE\\_PSP\\_GetVersionString](#) (void)  
*Obtain the PSP version/baseline identifier string.*
- const char \* [CFE\\_PSP\\_GetVersionCodeName](#) (void)  
*Obtain the version code name.*
- void [CFE\\_PSP\\_GetVersionNumber](#) (`uint8 VersionNumbers[4]`)  
*Retrieves the numeric PSP version identifier as an array of uint8 values.*
- `uint32 CFE_PSP_GetBuildNumber` (void)  
*Obtain the PSP library numeric build number.*

## 12.259.1 Function Documentation

**12.259.1.1 CFE\_PSP\_GetBuildNumber()** `uint32 CFE_PSP_GetBuildNumber ( void )`

Obtain the PSP library numeric build number.

The build number is a monotonically increasing number that (coarsely) reflects the number of commits/changes that have been merged since the epoch release. During development cycles this number should increase after each subsequent merge/modification.

Like other version information, this is a fixed number assigned at compile time.

### Returns

The OSAL library build number

**12.259.1.2 CFE\_PSP\_GetVersionCodeName()** `const char* CFE_PSP_GetVersionCodeName ( void )`

Obtain the version code name.

This retrieves the PSP code name. This is a compatibility indicator for the overall NASA cFS ecosystem. All modular components which are intended to interoperate should report the same code name.

### Returns

Code name. This is a fixed string and cannot be NULL.

**12.259.1.3 CFE\_PSP\_GetVersionNumber()** `void CFE_PSP_GetVersionNumber ( uint8 VersionNumbers[4] )`

Retrieves the numeric PSP version identifier as an array of uint8 values.

The array of numeric values is in order of precedence: [0] = Major Number [1] = Minor Number [2] = Revision Number [3] = Mission Revision

The "Mission Revision" (last output) also indicates whether this is an official release, a patched release, or a development version. 0 indicates an official release 1-254 local patch level (reserved for mission use) 255 indicates a development build

### Parameters

<code>out</code>	<code>VersionNumbers</code>	A fixed-size array to be filled with the version numbers
------------------	-----------------------------	--

**12.259.1.4 CFE\_PSP\_GetVersionString()** `const char* CFE_PSP_GetVersionString ( void )`

Obtain the PSP version/baseline identifier string.

This retrieves the PSP version identifier string without extra info

### Returns

Version string. This is a fixed string and cannot be NULL.

## 12.260 psp/fsw/inc/cfe\_psp\_watchdog\_api.h File Reference

```
#include "common_types.h"
```

```
#include "osapi.h"
#include "cfe_psp_error.h"
```

## Macros

### Definitions for PSP PANIC types

- #define CFE\_PSP\_PANIC\_STARTUP 1
- #define CFE\_PSP\_PANIC\_VOLATILE\_DISK 2
- #define CFE\_PSP\_PANIC\_MEMORY\_ALLOC 3
- #define CFE\_PSP\_PANIC\_NONVOL\_DISK 4
- #define CFE\_PSP\_PANIC\_STARTUP\_SEM 5
- #define CFE\_PSP\_PANIC\_CORE\_APP 6
- #define CFE\_PSP\_PANIC\_GENERAL\_FAILURE 7

### Reset Types

- #define CFE\_PSP\_RST\_TYPE\_PROCESSOR 1
- #define CFE\_PSP\_RST\_TYPE\_POWERON 2
- #define CFE\_PSP\_RST\_TYPE\_MAX 3

### Reset Sub-Types

- #define CFE\_PSP\_RST\_SUBTYPE\_POWER\_CYCLE 1  
*Reset caused by power having been removed and restored.*
- #define CFE\_PSP\_RST\_SUBTYPE\_PUSH\_BUTTON 2  
*Reset caused by reset button on the board.*
- #define CFE\_PSP\_RST\_SUBTYPE\_HW\_SPECIAL\_COMMAND 3  
*Reset was caused by a reset line having been stimulated by a hardware special command.*
- #define CFE\_PSP\_RST\_SUBTYPE\_HW\_WATCHDOG 4  
*Reset was caused by a watchdog timer expiring.*
- #define CFE\_PSP\_RST\_SUBTYPE\_RESET\_COMMAND 5  
*Reset was caused by cFE ES processing a [Reset Command](#).*
- #define CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION 6  
*Reset was caused by a Processor Exception.*
- #define CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET 7  
*Reset was caused in an unknown manner.*
- #define CFE\_PSP\_RST\_SUBTYPE\_HWDEBUG\_RESET 8  
*Reset was caused by a JTAG or BDM connection.*
- #define CFE\_PSP\_RST\_SUBTYPE\_BANKSWITCH\_RESET 9  
*Reset reverted to a cFE POWERON due to a boot bank switch.*
- #define CFE\_PSP\_RST\_SUBTYPE\_MAX 10  
*Placeholder to indicate 1+ the maximum value that the PSP will ever use.*

## Functions

- void CFE\_PSP\_Restart (uint32 resetType)  
*Entry point back to the BSP to restart the processor.*
- uint32 CFE\_PSP\_GetRestartType (uint32 \*restartSubType)  
*Returns the last reset type.*
- void CFE\_PSP\_WatchdogInit (void)  
*Configures the watchdog timer.*
- void CFE\_PSP\_WatchdogEnable (void)  
*Enables the watchdog timer.*

- void **CFE\_PSP\_WatchdogDisable** (void)  
*Disables the watchdog timer.*
- void **CFE\_PSP\_WatchdogService** (void)  
*Services the watchdog timer according to the value set in WatchDogSet.*
- uint32 **CFE\_PSP\_WatchdogGet** (void)  
*Gets the watchdog time in milliseconds.*
- void **CFE\_PSP\_WatchdogSet** (uint32 WatchdogValue)  
*Sets the watchdog time in milliseconds.*
- void **CFE\_PSP\_Panic** (int32 ErrorCode)  
*Aborts the cFE startup.*

### 12.260.1 Macro Definition Documentation

#### 12.260.1.1 CFE\_PSP\_PANIC\_CORE\_APP #define CFE\_PSP\_PANIC\_CORE\_APP 6

Definition at line 59 of file cfe\_psp\_watchdog\_api.h.

#### 12.260.1.2 CFE\_PSP\_PANIC\_GENERAL\_FAILURE #define CFE\_PSP\_PANIC\_GENERAL\_FAILURE 7

Definition at line 60 of file cfe\_psp\_watchdog\_api.h.

#### 12.260.1.3 CFE\_PSP\_PANIC\_MEMORY\_ALLOC #define CFE\_PSP\_PANIC\_MEMORY\_ALLOC 3

Definition at line 56 of file cfe\_psp\_watchdog\_api.h.

#### 12.260.1.4 CFE\_PSP\_PANIC\_NONVOL\_DISK #define CFE\_PSP\_PANIC\_NONVOL\_DISK 4

Definition at line 57 of file cfe\_psp\_watchdog\_api.h.

#### 12.260.1.5 CFE\_PSP\_PANIC\_STARTUP #define CFE\_PSP\_PANIC\_STARTUP 1

Definition at line 54 of file cfe\_psp\_watchdog\_api.h.

#### 12.260.1.6 CFE\_PSP\_PANIC\_STARTUP\_SEM #define CFE\_PSP\_PANIC\_STARTUP\_SEM 5

Definition at line 58 of file cfe\_psp\_watchdog\_api.h.

#### 12.260.1.7 CFE\_PSP\_PANIC\_VOLATILE\_DISK #define CFE\_PSP\_PANIC\_VOLATILE\_DISK 2

Definition at line 55 of file cfe\_psp\_watchdog\_api.h.

#### 12.260.1.8 CFE\_PSP\_RST\_SUBTYPE\_BANKSWITCH\_RESET #define CFE\_PSP\_RST\_SUBTYPE\_BANKSWITCH\_RESET 9

Reset reverted to a cFE POWERON due to a boot bank switch.

Definition at line 96 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.9 CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION** #define CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION 6

Reset was caused by a Processor Exception.

Definition at line 90 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.10 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 84 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.11 CFE\_PSP\_RST\_SUBTYPE\_HW\_WATCHDOG** #define CFE\_PSP\_RST\_SUBTYPE\_HW\_WATCHDOG 4

Reset was caused by a watchdog timer expiring.

Definition at line 86 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.12 CFE\_PSP\_RST\_SUBTYPE\_HWDEBUG\_RESET** #define CFE\_PSP\_RST\_SUBTYPE\_HWDEBUG\_RESET 8

Reset was caused by a JTAG or BDM connection.

Definition at line 94 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.13 CFE\_PSP\_RST\_SUBTYPE\_MAX** #define CFE\_PSP\_RST\_SUBTYPE\_MAX 10

Placeholder to indicate 1+ the maximum value that the PSP will ever use.

Definition at line 98 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.14 CFE\_PSP\_RST\_SUBTYPE\_POWER\_CYCLE** #define CFE\_PSP\_RST\_SUBTYPE\_POWER\_CYCLE 1

Reset caused by power having been removed and restored.

Definition at line 80 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.15 CFE\_PSP\_RST\_SUBTYPE\_PUSH\_BUTTON** #define CFE\_PSP\_RST\_SUBTYPE\_PUSH\_BUTTON 2

Reset caused by reset button on the board.

Definition at line 82 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.16 CFE\_PSP\_RST\_SUBTYPE\_RESET\_COMMAND** #define CFE\_PSP\_RST\_SUBTYPE\_RESET\_COMMAND 5

Reset was caused by cFE ES processing a [Reset Command](#).

Definition at line 88 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.17 CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET** #define CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET 7

Reset was caused in an unknown manner.

Definition at line 92 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.18 CFE\_PSP\_RST\_TYPE\_MAX** #define CFE\_PSP\_RST\_TYPE\_MAX 3

Placeholder to indicate 1+ the maximum value that the PSP will ever use.

Definition at line 70 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.19 CFE\_PSP\_RST\_TYPE\_POWERON** #define CFE\_PSP\_RST\_TYPE\_POWERON 2  
 All memory has been cleared  
 Definition at line 69 of file cfe\_psp\_watchdog\_api.h.

**12.260.1.20 CFE\_PSP\_RST\_TYPE\_PROCESSOR** #define CFE\_PSP\_RST\_TYPE\_PROCESSOR 1  
 Volatile disk, CDS and User Reserved memory may be valid  
 Definition at line 68 of file cfe\_psp\_watchdog\_api.h.

## 12.260.2 Function Documentation

**12.260.2.1 CFE\_PSP\_GetRestartType()** uint32 CFE\_PSP\_GetRestartType ( uint32 \* restartSubType )

Returns the last reset type.

### Note

If a pointer to a valid memory space is passed in, it returns the reset sub-type in that memory. Right now the reset types are application-specific. For the cFE they are defined in the [cfe\\_es.h](#) file.

### Parameters

restartSubType	
----------------	--

**12.260.2.2 CFE\_PSP\_Panic()** void CFE\_PSP\_Panic ( int32 ErrorCode )

Aborts the cFE startup.

Provides a common interface to abort the cFE startup process and return back to the OS.

### Note

This is called by the cFE Core startup code when it needs to abort the cFE startup. This should not be called by applications.

### Parameters

in	ErrorCode	Reason for exiting
----	-----------	--------------------

**12.260.2.3 CFE\_PSP\_Restart()** void CFE\_PSP\_Restart ( uint32 resetType )

Entry point back to the BSP to restart the processor.

The flight software calls this routine to restart the processor.

### Parameters

in	resetType	Type of reset
----	-----------	---------------

```
12.260.2.4 CFE_PSP_WatchdogDisable() void CFE_PSP_WatchdogDisable (
    void )
```

Disables the watchdog timer.

```
12.260.2.5 CFE_PSP_WatchdogEnable() void CFE_PSP_WatchdogEnable (
    void )
```

Enables the watchdog timer.

```
12.260.2.6 CFE_PSP_WatchdogGet() uint32 CFE_PSP_WatchdogGet (
    void )
```

Gets the watchdog time in milliseconds.

#### Returns

The current watchdog value

```
12.260.2.7 CFE_PSP_WatchdogInit() void CFE_PSP_WatchdogInit (
    void )
```

Configures the watchdog timer.

To set up the timer resolution and/or other settings custom to this platform.

```
12.260.2.8 CFE_PSP_WatchdogService() void CFE_PSP_WatchdogService (
    void )
```

Services the watchdog timer according to the value set in WatchDogSet.

Load the watchdog timer with a count that corresponds to the millisecond time given in the parameter.

#### Note

Currently an ExpireTime value of zero will result in the minimum reset time of 4.5 seconds. All other ExpireTime values will result in a reset time of 5.5 seconds.

```
12.260.2.9 CFE_PSP_WatchdogSet() void CFE_PSP_WatchdogSet (
    uint32 WatchdogValue )
```

Sets the watchdog time in milliseconds.

#### Parameters

in	<i>WatchdogValue</i>	New watchdog value to set
----	----------------------	---------------------------

## Index

\_EXTENSION\_  
    common\_types.h, [1747](#)

accuracy  
    OS\_timebase\_prop\_t, [797](#)  
    OS\_timer\_prop\_t, [798](#)

ack  
    CF\_Logical\_IntHeader, [581](#)

ack\_directive\_code  
    CF\_Logical\_PduAck, [583](#)

ack\_limit  
    CF\_ChannelConfig, [544](#)  
    CF\_HkFault, [574](#)

ack\_subtype\_code  
    CF\_Logical\_PduAck, [583](#)

ack\_timer  
    CF\_Transaction, [614](#)

ack\_timer\_armed  
    CF\_Flags\_Common, [562](#)

ack\_timer\_s  
    CF\_ChannelConfig, [544](#)

acknak\_count  
    CF\_StateData, [609](#)

action  
    CF\_ChanAction\_SuspResArg, [541](#)

ActiveBuffer  
    CFE\_TBL\_HousekeepingTlm\_Payload, [741](#)

ActiveBufferAddr  
    CFE\_TBL\_TblRegPacket\_Payload, [752](#)

ActiveTableFlag  
    CFE\_TBL\_DumpCmd\_Payload, [735](#)  
    CFE\_TBL\_ValidateCmd\_Payload, [755](#)

ActualLength  
    OS\_SockAddr\_t, [792](#)

ActualType  
    CFE\_Config\_ValueEntry, [629](#)

addr  
    OS\_module\_prop\_t, [790](#)

AddrData  
    OS\_SockAddr\_t, [792](#)

Address  
    OS\_static\_symbol\_record\_t, [795](#)

AddressesAreValid  
    CFE\_ES\_AppInfo, [631](#)

AdjustmentFactor  
    CFE\_TIME\_HousekeepingTlm\_Payload, [767](#)

AlignPtr  
    OS\_SockAddrData\_t, [793](#)

AlignU32  
    OS\_SockAddrData\_t, [793](#)

AppData  
    CFE\_EVS\_HousekeepingTlm\_Payload, [688](#)

AppDataFilename  
    CFE\_EVS\_AppDataCmd\_Payload, [675](#)

AppEnableStatus  
    CFE\_EVS\_AppTlmData, [679](#)

AppEntryPoint  
    CFE\_ES\_StartAppCmd\_Payload, [668](#)

AppFileName  
    CFE\_ES\_AppReloadCmd\_Payload, [635](#)  
    CFE\_ES\_StartAppCmd\_Payload, [668](#)

AppID  
    CFE\_EVS\_AppTlmData, [679](#)

AppId  
    CFE\_ES\_TaskInfo, [672](#)  
    CFE\_SB\_PipeInfoEntry, [716](#)

AppInfo  
    CFE\_ES\_OneAppTlm\_Payload, [654](#)

Application  
    CFE\_ES\_AppNameCmd\_Payload, [634](#)  
    CFE\_ES\_AppReloadCmd\_Payload, [635](#)  
    CFE\_ES\_SendMemPoolStatsCmd\_Payload, [663](#)  
    CFE\_ES\_StartAppCmd\_Payload, [668](#)

ApplicationID  
    CFE\_FS\_Header, [703](#)

AppMessageSentCounter  
    CFE\_EVS\_AppTlmData, [679](#)

AppMessageSquelchedCounter  
    CFE\_EVS\_AppTlmData, [679](#)

AppName  
    CFE\_ES\_TaskInfo, [672](#)  
    CFE\_EVS\_AppNameBitMaskCmd\_Payload, [676](#)  
    CFE\_EVS\_AppNameCmd\_Payload, [677](#)  
    CFE\_EVS\_AppNameEventIDCmd\_Payload, [677](#)  
    CFE\_EVS\_AppNameEventIDMaskCmd\_Payload, [678](#)  
    CFE\_EVS\_PacketID, [693](#)  
    CFE\_SB\_PipeInfoEntry, [716](#)  
    CFE\_SB\_RoutingFileEntry, [719](#)

apps/cf/config/default\_cf\_extern\_typedefs.h, [799](#)  
apps/cf/config/default\_cf\_fcncode\_values.h, [801](#)  
apps/cf/config/default\_cf\_interface\_cfg\_values.h, [802](#)  
apps/cf/config/default\_cf\_internal\_cfg\_values.h, [803](#)  
apps/cf/config/default\_cf\_mission\_cfg.h, [803](#)  
apps/cf/config/default\_cf\_msg.h, [804](#)  
apps/cf/config/default\_cf\_msgdefs.h, [804](#)  
apps/cf/config/default\_cf\_msqid\_values.h, [806](#)  
apps/cf/config/default\_cf\_msgids.h, [807](#)  
apps/cf/config/default\_cf\_msgstruct.h, [807](#)  
apps/cf/config/default\_cf\_platform\_cfg.h, [810](#)  
apps/cf/config/default\_cf\_tbl.h, [811](#)  
apps/cf/config/default\_cf\_tbldefs.h, [812](#)

apps/cf/config/default\_cf\_tblstruct.h, 812  
 apps/cf/config/default\_cf\_topicid\_values.h, 813  
 apps/cf/config/eds\_cf\_extern\_typedefs.h, 813  
 apps/cf/config/eds\_cf\_fncode\_values.h, 815  
 apps/cf/config/eds\_cf\_interface\_cfg\_values.h, 815  
 apps/cf/config/eds\_cf\_msgdefs.h, 816  
 apps/cf/config/eds\_cf\_msgstruct.h, 816  
 apps/cf/config/eds\_cf\_tbldefs.h, 816  
 apps/cf/config/eds\_cf\_tblstruct.h, 816  
 apps/cf/config/eds\_cf\_topicid\_values.h, 816  
 apps/cf/docs/dox\_src/cfs\_cf.dox, 817  
 apps/cf/fsw/inc/cf\_eventids.h, 817  
 apps/cf/fsw/inc/cf\_fcncodes.h, 823  
 apps/cf/fsw/inc/cf\_interface\_cfg.h, 824  
 apps/cf/fsw/inc/cf\_internal\_cfg.h, 825  
 apps/cf/fsw/inc/cf\_perfids.h, 826  
 apps/cf/fsw/inc/cf\_topicids.h, 826  
 apps/cf/fsw/src/cf\_app.c, 829  
 apps/cf/fsw/src/cf\_app.h, 835  
 apps/cf/fsw/src/cf\_assert.h, 843  
 apps/cf/fsw/src/cf\_cfdp.c, 844  
 apps/cf/fsw/src/cf\_cfdp.h, 897  
 apps/cf/fsw/src/cf\_cfdp\_dispatch.c, 947  
 apps/cf/fsw/src/cf\_cfdp\_dispatch.h, 949  
 apps/cf/fsw/src/cf\_cfdp\_pdu.h, 953  
 apps/cf/fsw/src/cf\_cfdp\_r.c, 960  
 apps/cf/fsw/src/cf\_cfdp\_r.h, 980  
 apps/cf/fsw/src/cf\_cfdp\_s.c, 997  
 apps/cf/fsw/src/cf\_cfdp\_s.h, 1013  
 apps/cf/fsw/src/cf\_cfdp\_sbintf.c, 1027  
 apps/cf/fsw/src/cf\_cfdp\_sbintf.h, 1031  
 apps/cf/fsw/src/cf\_cfdp\_types.h, 1035  
 apps/cf/fsw/src/cf\_chunk.c, 1042  
 apps/cf/fsw/src/cf\_chunk.h, 1052  
 apps/cf/fsw/src/cf\_clist.c, 1063  
 apps/cf/fsw/src/cf\_clist.h, 1068  
 apps/cf/fsw/src/cf\_cmd.c, 1074  
 apps/cf/fsw/src/cf\_cmd.h, 1105  
 apps/cf/fsw/src/cf\_codec.c, 1137  
 apps/cf/fsw/src/cf\_codec.h, 1168  
 apps/cf/fsw/src/cf\_crc.c, 1198  
 apps/cf/fsw/src/cf\_crc.h, 1199  
 apps/cf/fsw/src/cf\_dispatch.c, 1201  
 apps/cf/fsw/src/cf\_dispatch.h, 1204  
 apps/cf/fsw/src/cf\_eds\_dispatch.c, 1206  
 apps/cf/fsw/src/cf\_logical\_pdu.h, 1209  
 apps/cf/fsw/src/cf\_timer.c, 1213  
 apps/cf/fsw/src/cf\_timer.h, 1215  
 apps/cf/fsw/src/cf\_utils.c, 1218  
 apps/cf/fsw/src/cf\_utils.h, 1237  
 apps/cf/fsw/src/cf\_verify.h, 1260  
 apps/cf/fsw/src/cf\_version.h, 1260  
 apps/cf/fsw/tables/cf\_def\_config.c, 1261  
 ARGCHECK  
 osapi-macros.h, 1770  
 AsInteger  
     CFE\_Config\_ValueBuffer, 629  
 AsPointer  
     CFE\_Config\_ValueBuffer, 629  
 AtToneDelay  
     CFE\_TIME\_DiagnosticTlm\_Payload, 759  
 AtToneLatch  
     CFE\_TIME\_DiagnosticTlm\_Payload, 760  
 AtToneLeapSeconds  
     CFE\_TIME\_DiagnosticTlm\_Payload, 760  
     CFE\_TIME\_ToneDataCmd\_Payload, 782  
 AtToneMET  
     CFE\_TIME\_DiagnosticTlm\_Payload, 760  
     CFE\_TIME\_ToneDataCmd\_Payload, 782  
 AtToneState  
     CFE\_TIME\_ToneDataCmd\_Payload, 782  
 AtToneSTCF  
     CFE\_TIME\_DiagnosticTlm\_Payload, 760  
     CFE\_TIME\_ToneDataCmd\_Payload, 782  
 barg  
     CF\_ChanAction\_BoolArg, 540  
     CF\_ChanAction\_BoolMsgArg, 540  
 base  
     CF\_DecoderState, 553  
     CF\_EncoderState, 557  
 BitMask  
     CFE\_EVS\_AppNameBitMaskCmd\_Payload, 676  
     CFE\_EVS\_BitMaskCmd\_Payload, 680  
 bits  
     CFE\_ES\_MemAddress\_t, 650  
     CFE\_ES\_MemOffset\_t, 651  
 block\_size  
     OS\_statvfs\_t, 795  
 blocks\_free  
     OS\_statvfs\_t, 795  
 BlockSize  
     CFE\_ES\_BlockStats, 635  
 BlockStats  
     CFE\_ES\_MemPoolStats, 652  
 BootSource  
     CFE\_ES\_HousekeepingTlm\_Payload, 644  
 bss\_address  
     OS\_module\_address\_t, 789  
 bss\_size  
     OS\_module\_address\_t, 789  
 BSSAddress  
     CFE\_ES\_AppInfo, 631  
 BSSSize  
     CFE\_ES\_AppInfo, 631  
 Buffer  
     OS\_SockAddrData\_t, 793  
 BUGCHECK

osapi-macros.h, 1771  
BUGCHECK\_VOID  
osapi-macros.h, 1771  
BUGREPORT  
osapi-macros.h, 1771  
buildosal\_public\_apiincosconfig.h, 1261  
busy  
CF\_Playback, 600  
byte  
CF\_UnionArgs\_Payload, 625  
ByteAlign4  
CFE\_TBL\_TblRegPacket\_Payload, 752  
ByteAlignPad1  
CFE\_TBL\_HousekeepingTlm\_Payload, 741  
ByteAlignSpare  
CFE\_ES\_CDSRegDumpRec, 636  
cached\_pos  
CF\_StateData, 609  
canceled  
CF\_Flags\_Common, 562  
cc  
CF\_CFDP\_PduEof, 529  
CF\_Logical\_PduAck, 583  
CF\_Logical\_PduEof, 586  
CF\_Logical\_PduFin, 588  
cc\_and\_transaction\_status  
CF\_CFDP\_PduAck, 528  
CCSDS\_ExtendedHeader, 523  
Subsystem, 523  
SystemId, 523  
CCSDS\_ExtendedHeader\_t  
ccsds\_hdr.h, 1564  
ccsds\_hdr.h  
CCSDS\_ExtendedHeader\_t, 1564  
CCSDS\_PrimaryHeader\_t, 1564  
CCSDS\_PrimaryHeader, 523  
Length, 524  
Sequence, 524  
StreamId, 524  
CCSDS\_PrimaryHeader\_t  
ccsds\_hdr.h, 1564  
CdsName  
CFE\_ES\_DeleteCDSCmd\_Payload, 639  
CF\_ABANDON\_CC  
CFS CFDP Command Codes, 194  
CF\_Abandon\_TxnCmd  
cf\_cmd.c, 1076  
cf\_cmd.h, 1109  
CF\_AbandonCmd, 524  
cf\_cmd.c, 1077  
cf\_cmd.h, 1109  
CommandHeader, 524  
Payload, 524  
CF\_AbandonCmd\_t  
CFS CFDP Command Structures, 224  
CF\_ALL\_CHANNELS  
default\_cf\_msg.h, 804  
CF\_ALL\_POLLDIRS  
default\_cf\_msg.h, 804  
cf\_app.c  
CF\_AppData, 835  
CF\_AppInit, 830  
CF\_AppMain, 831  
CF\_CheckTables, 832  
CF\_TableInit, 833  
CF\_ValidateConfigTable, 834  
cf\_app.h  
CF\_AppData, 843  
CF\_AppInit, 838  
CF\_AppMain, 839  
CF\_CHANNEL\_PIPE\_PREFIX, 837  
CF\_CheckTables, 840  
CF\_ERROR, 837  
CF\_FILENAME\_TRUNCATED, 837  
CF\_PDU\_METADATA\_ERROR, 837  
CF\_PIPE\_NAME, 837  
CF\_REC\_PDU\_BAD\_EOF\_ERROR, 837  
CF\_REC\_PDU\_FSIZE\_MISMATCH\_ERROR, 837  
CF\_SEND\_PDU\_ERROR, 837  
CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR, 837  
CF\_SHORT\_PDU\_ERROR, 837  
CF\_TableInit, 841  
CF\_ValidateConfigTable, 842  
CF\_APP\_MAX\_HEADER\_SIZE  
cf\_cfdp\_pdu.h, 956  
CF\_AppData  
cf\_app.c, 835  
cf\_app.h, 843  
CF\_AppData\_t, 525  
CmdPipe, 525  
config\_handle, 525  
config\_table, 525  
engine, 525  
hk, 526  
RunStatus, 526  
CF\_AppInit  
cf\_app.c, 830  
cf\_app.h, 838  
CF\_AppMain  
cf\_app.c, 831  
cf\_app.h, 839  
CF\_AppPipe  
cf\_dispatch.c, 1202  
cf\_dispatch.h, 1204  
cf\_eds\_dispatch.c, 1207  
CF\_Assert  
cf\_assert.h, 844

cf\_assert.h  
   CF\_ASSERT, 844  
   CF\_TRACE, 844

CF\_CANCEL\_CC  
   CFS CFDP Command Codes, 194

CF\_Cancel\_TxnCmd  
   cf\_cmd.c, 1078  
   cf\_cmd.h, 1110

CF\_CancelCmd, 526  
   cf\_cmd.c, 1078  
   cf\_cmd.h, 1111  
   CommandHeader, 526  
   Payload, 526

CF\_CancelCmd\_t  
   CFS CFDP Command Structures, 224

CF\_CC\_ERR\_EID  
   CFS CFDP Event IDs, 158

CF\_CCVAL  
   default\_cf\_fcncode\_values.h, 801  
   eds\_cf\_fcncode\_values.h, 815

cf\_cfdp.c  
   CF\_CFDP\_AllocChunkList, 847  
   CF\_CFDP\_AppendTlv, 848  
   CF\_CFDP\_ArmAckTimer, 848  
   CF\_CFDP\_ArmInactTimer, 849  
   CF\_CFDP\_CancelTransaction, 850  
   CF\_CFDP\_CheckAckNakCount, 850  
   CF\_CFDP\_CloseFiles, 851  
   CF\_CFDP\_CompleteTick, 852  
   CF\_CFDP\_ConstructPduHeader, 853  
   CF\_CFDP\_CopyStringFromLV, 854  
   CF\_CFDP\_CycleEngine, 855  
   CF\_CFDP\_DecodeStart, 856  
   CF\_CFDP\_DisableEngine, 857  
   CF\_CFDP\_DispatchRecv, 858  
   CF\_CFDP\_DoTick, 859  
   CF\_CFDP\_EncodeStart, 860  
   CF\_CFDP\_FindUnusedChunks, 860  
   CF\_CFDP\_FinishTransaction, 861  
   CF\_CFDP\_GetClass, 862  
   CF\_CFDP\_GetMoveTarget, 862  
   CF\_CFDP\_GetTempName, 863  
   CF\_CFDP\_GetTxnStatus, 863  
   CF\_CFDP\_InitEngine, 864  
   CF\_CFDP\_InitTxnTxFile, 865  
   CF\_CFDP\_IsSender, 866  
   CF\_CFDP\_PlaybackDir, 866  
   CF\_CFDP\_PlaybackDir\_Initiate, 867  
   CF\_CFDP\_ProcessPlaybackDirectories, 867  
   CF\_CFDP\_ProcessPlaybackDirectory, 868  
   CF\_CFDP\_ProcessPollingDirectories, 869  
   CF\_CFDP\_ReceivePdu, 870  
   CF\_CFDP\_RecvAck, 870  
   CF\_CFDP\_RecvDrop, 871

CF\_CFDP\_RecvEof, 872  
   CF\_CFDP\_RecvFd, 873  
   CF\_CFDP\_RecvFin, 874  
   CF\_CFDP\_RecvHold, 874  
   CF\_CFDP\_RecvInit, 875  
   CF\_CFDP\_RecvMd, 876  
   CF\_CFDP\_RecvNak, 877  
   CF\_CFDP\_RecvPh, 878  
   CF\_CFDP\_RecycleTransaction, 879  
   CF\_CFDP\_S\_Tick\_NewData, 880  
   CF\_CFDP\_SendAck, 881  
   CF\_CFDP\_SendEof, 882  
   CF\_CFDP\_SendEotPkt, 883  
   CF\_CFDP\_SendFd, 884  
   CF\_CFDP\_SendFin, 885  
   CF\_CFDP\_SendMd, 886  
   CF\_CFDP\_SendNak, 887  
   CF\_CFDP\_SetPduLength, 888  
   CF\_CFDP\_SetTxnStatus, 888  
   CF\_CFDP\_SetupRxTransaction, 889  
   CF\_CFDP\_SetupTxTransaction, 890  
   CF\_CFDP\_StartFirstPending, 891  
   CF\_CFDP\_StartRxTransaction, 892  
   CF\_CFDP\_TickTransactions, 893  
   CF\_CFDP\_TxFile, 894  
   CF\_CFDP\_TxFile\_Initiate, 895  
   CF\_CFDP\_TxnIsOK, 896  
   CF\_CFDP\_UpdatePollPbCounted, 897

cf\_cfdp.h  
   CF\_CFDP\_AllocChunkList, 900  
   CF\_CFDP\_AppendTlv, 901  
   CF\_CFDP\_ArmAckTimer, 901  
   CF\_CFDP\_ArmInactTimer, 902  
   CF\_CFDP\_CancelTransaction, 903  
   CF\_CFDP\_CheckAckNakCount, 903  
   CF\_CFDP\_CloseFiles, 904  
   CF\_CFDP\_CompleteTick, 905  
   CF\_CFDP\_ConstructPduHeader, 906  
   CF\_CFDP\_CopyStringFromLV, 907  
   CF\_CFDP\_CycleEngine, 908  
   CF\_CFDP\_DecodeStart, 909  
   CF\_CFDP\_DisableEngine, 910  
   CF\_CFDP\_DispatchRecv, 911  
   CF\_CFDP\_DoTick, 912  
   CF\_CFDP\_EncodeStart, 913  
   CF\_CFDP\_FinishTransaction, 913  
   CF\_CFDP\_GetMoveTarget, 914  
   CF\_CFDP\_GetPrintClass, 915  
   CF\_CFDP\_GetTempName, 916  
   CF\_CFDP\_GetTxnStatus, 916  
   CF\_CFDP\_InitEngine, 916  
   CF\_CFDP\_InitTxnTxFile, 918  
   CF\_CFDP\_PlaybackDir, 919  
   CF\_CFDP\_ProcessPlaybackDirectory, 920

CF\_CFDP\_ProcessPollingDirectories, 921  
CF\_CFDP\_ReceivePdu, 922  
CF\_CFDP\_RecvAck, 922  
CF\_CFDP\_RecvDrop, 923  
CF\_CFDP\_RecvEof, 924  
CF\_CFDP\_RecvFd, 925  
CF\_CFDP\_RecvFin, 926  
CF\_CFDP\_RecvHold, 926  
CF\_CFDP\_RecvInit, 927  
CF\_CFDP\_RecvMd, 928  
CF\_CFDP\_RecvNak, 929  
CF\_CFDP\_RecvPh, 930  
CF\_CFDP\_RecycleTransaction, 931  
CF\_CFDP\_S\_Tick\_NewData, 932  
CF\_CFDP\_SendAck, 933  
CF\_CFDP\_SendEof, 934  
CF\_CFDP\_SendEotPkt, 935  
CF\_CFDP\_SendFd, 936  
CF\_CFDP\_SendFin, 937  
CF\_CFDP\_SendMd, 938  
CF\_CFDP\_SendNak, 939  
CF\_CFDP\_SetTxnStatus, 940  
CF\_CFDP\_SetupRxTransaction, 940  
CF\_CFDP\_SetupTxTransaction, 941  
CF\_CFDP\_StartFirstPending, 942  
CF\_CFDP\_StartRxTransaction, 943  
CF\_CFDP\_Tick\_args\_t, 900  
CF\_CFDP\_TickTransactions, 944  
CF\_CFDP\_TxFile, 945  
CF\_CFDP\_TxnIsOK, 946  
CF\_CFDP\_AckTxnStatus\_ACTIVE  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_AckTxnStatus\_INVALID  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_AckTxnStatus\_t  
    cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_AckTxnStatus\_TERMINATED  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_AckTxnStatus\_UNDEFINED  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_AckTxnStatus\_UNRECOGNIZED  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_AllocChunkList  
    cf\_cfdp.c, 847  
    cf\_cfdp.h, 900  
CF\_CFDP\_AppendTlv  
    cf\_cfdp.c, 848  
    cf\_cfdp.h, 901  
CF\_CFDP\_ArmAckTimer  
    cf\_cfdp.c, 848  
    cf\_cfdp.h, 901  
CF\_CFDP\_ArmInactTimer  
    cf\_cfdp.c, 849  
    cf\_cfdp.h, 902  
CF\_CFDP\_CancelTransaction  
    cf\_cfdp.c, 850  
    cf\_cfdp.h, 903  
CF\_CFDP\_CheckAckNakCount  
    cf\_cfdp.c, 850  
    cf\_cfdp.h, 903  
CF\_CFDP\_CLASS\_1  
    default\_cf\_extern\_typedefs.h, 800  
CF\_CFDP\_CLASS\_2  
    default\_cf\_extern\_typedefs.h, 800  
CF\_CFDP\_Class\_t  
    default\_cf\_extern\_typedefs.h, 800  
    eds\_cf\_extern\_typedefs.h, 814  
CF\_CFDP\_CLOSE\_ERR\_EID  
    CFS CFDP Event IDs, 159  
CF\_CFDP\_CloseFiles  
    cf\_cfdp.c, 851  
    cf\_cfdp.h, 904  
CF\_CFDP\_CodecCheckSize  
    cf\_codec.c, 1140  
    cf\_codec.h, 1173  
CF\_CFDP\_CodecGetPosition  
    cf\_codec.h, 1174  
CF\_CFDP\_CodecGetRemain  
    cf\_codec.h, 1174  
CF\_CFDP\_CodecGetSize  
    cf\_codec.h, 1175  
CF\_CFDP\_CodeclsOK  
    cf\_codec.h, 1175  
CF\_CFDP\_CodecReset  
    cf\_codec.h, 1175  
CF\_CFDP\_CodecSetDone  
    cf\_codec.h, 1176  
CF\_CFDP\_CompleteTick  
    cf\_cfdp.c, 852  
    cf\_cfdp.h, 905  
CF\_CFDP\_ConditionCode\_CANCEL\_REQUEST\_RECEIVED  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_CHECK\_LIMIT\_REACHED  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_FILE\_CHECKSUM\_FAILURE  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_FILE\_SIZE\_ERROR  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_FILESTORE\_REJECTION  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_INACTIVITY\_DETECTED  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_INVALID\_FILE\_STRUCTURE  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_INVALID\_TRANSMISSION\_MODE  
    cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_ConditionCode\_KEEP\_ALIVE\_LIMIT\_REACHED  
    cf\_cfdp\_pdu.h, 958

CF\_CFDP\_ConditionCode\_NAK\_LIMIT\_REACHED  
    cf\_cfdp\_pdu.h, 958

CF\_CFDP\_ConditionCode\_NO\_ERROR  
    cf\_cfdp\_pdu.h, 958

CF\_CFDP\_ConditionCode\_POS\_ACK\_LIMIT\_REACHED  
    cf\_cfdp\_pdu.h, 958

CF\_CFDP\_ConditionCode\_SUSPEND\_REQUEST\_RECEIVED  
    cf\_cfdp\_pdu.h, 958

CF\_CFDP\_ConditionCode\_t  
    cf\_cfdp\_pdu.h, 958

CF\_CFDP\_ConditionCode\_UNSUPPORTED\_CHECKSUM\_TYPE  
    cf\_cfdp\_pdu.h, 958

CF\_CFDP\_ConstructPduHeader  
    cf\_cfdp.c, 853  
    cf\_cfdp.h, 906

CF\_CFDP\_CopyStringFromLV  
    cf\_cfdp.c, 854  
    cf\_cfdp.h, 907

CF\_CFDP\_CycleEngine  
    cf\_cfdp.c, 855  
    cf\_cfdp.h, 908

CF\_CFDP\_DecodeAck  
    cf\_codec.c, 1141  
    cf\_codec.h, 1176

CF\_CFDP\_DecodeAllSegments  
    cf\_codec.c, 1142  
    cf\_codec.h, 1176

CF\_CFDP\_DecodeAllTlv  
    cf\_codec.c, 1142  
    cf\_codec.h, 1177

CF\_CFDP\_DecodeCrc  
    cf\_codec.c, 1143  
    cf\_codec.h, 1177

CF\_CFDP\_DecodeEof  
    cf\_codec.c, 1143  
    cf\_codec.h, 1178

CF\_CFDP\_DecodeFileDataHeader  
    cf\_codec.c, 1144  
    cf\_codec.h, 1178

CF\_CFDP\_DecodeFileDirectiveHeader  
    cf\_codec.c, 1145  
    cf\_codec.h, 1179

CF\_CFDP\_DecodeFin  
    cf\_codec.c, 1145  
    cf\_codec.h, 1180

CF\_CFDP\_DecodeHeader  
    cf\_codec.c, 1146  
    cf\_codec.h, 1180

CF\_CFDP\_DecodeLv  
    cf\_codec.c, 1147  
    cf\_codec.h, 1181

CF\_CFDP\_DecodeMd  
    cf\_codec.c, 1147  
    cf\_codec.h, 1182

CF\_CFDP\_DecodeNak  
    cf\_codec.c, 1149  
    cf\_codec.h, 1182

CF\_CFDP\_DecodeSegmentRequest  
    cf\_codec.c, 1149  
    cf\_codec.h, 1183

CF\_CFDP\_DecodeStart  
    cf\_cfdp.c, 856  
    cf\_cfdp.h, 909

CF\_CFDP\_DecodeTlv  
    cf\_codec.c, 1150  
    cf\_codec.h, 1184

CF\_CFDP\_DIR\_SLOT\_ERR\_EID  
    CFS CFDP Event IDs, 159

CF\_CFDP\_DisableEngine  
    cf\_cfdp.c, 857  
    cf\_cfdp.h, 910

cf\_cfdp\_dispatch.c  
    CF\_CFDP\_R\_DispatchRecv, 948  
    CF\_CFDP\_RxStateDispatch, 948  
    CF\_CFDP\_S\_DispatchRecv, 949

cf\_cfdp\_dispatch.h  
    CF\_CFDP\_R\_DispatchRecv, 951  
    CF\_CFDP\_RxStateDispatch, 952  
    CF\_CFDP\_S\_DispatchRecv, 952  
    CF\_CFDP\_StateRecvFunc\_t, 950  
    CF\_CFDP\_StateSendFunc\_t, 951

CF\_CFDP\_DispatchRecv  
    cf\_cfdp.c, 858  
    cf\_cfdp.h, 911

CF\_CFDP\_DoDecodeChunk  
    cf\_codec.c, 1151  
    cf\_codec.h, 1184

CF\_CFDP\_DoEncodeChunk  
    cf\_codec.c, 1151  
    cf\_codec.h, 1185

CF\_CFDP\_DoTick  
    cf\_cfdp.c, 859  
    cf\_cfdp.h, 912

CF\_CFDP\_EncodeAck  
    cf\_codec.c, 1152  
    cf\_codec.h, 1186

CF\_CFDP\_EncodeAllSegments  
    cf\_codec.c, 1153  
    cf\_codec.h, 1186

CF\_CFDP\_EncodeAllTlv  
    cf\_codec.c, 1153  
    cf\_codec.h, 1187

CF\_CFDP\_EncodeCrc  
    cf\_codec.c, 1154  
    cf\_codec.h, 1187

CF\_CFDP\_EncodeEof  
    cf\_codec.c, 1154  
    cf\_codec.h, 1188

CF\_CFDP\_EncodeFileDataHeader  
  cf\_codec.c, 1155  
  cf\_codec.h, 1188  
CF\_CFDP\_EncodeFileDirectiveHeader  
  cf\_codec.c, 1156  
  cf\_codec.h, 1189  
CF\_CFDP\_EncodeFin  
  cf\_codec.c, 1156  
  cf\_codec.h, 1190  
CF\_CFDP\_EncodeHeaderFinalSize  
  cf\_codec.c, 1157  
  cf\_codec.h, 1190  
CF\_CFDP\_EncodeHeaderWithoutSize  
  cf\_codec.c, 1158  
  cf\_codec.h, 1191  
CF\_CFDP\_EncodeLV  
  cf\_codec.c, 1159  
  cf\_codec.h, 1192  
CF\_CFDP\_EncodeMd  
  cf\_codec.c, 1159  
  cf\_codec.h, 1192  
CF\_CFDP\_EncodeNak  
  cf\_codec.c, 1160  
  cf\_codec.h, 1193  
CF\_CFDP\_EncodeSegmentRequest  
  cf\_codec.c, 1161  
  cf\_codec.h, 1194  
CF\_CFDP\_EncodeStart  
  cf\_cfdp.c, 860  
  cf\_cfdp.h, 913  
CF\_CFDP\_EncodeTLV  
  cf\_codec.c, 1161  
  cf\_codec.h, 1194  
CF\_CFDP\_FD\_UNHANDLED\_ERR\_EID  
  CFS CFDP Event IDs, 159  
CF\_CFDP\_FileDirective\_ACK  
  cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_FileDirective\_EOF  
  cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_FileDirective\_FIN  
  cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_FileDirective\_INVALID\_MAX  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FileDirective\_INVALID\_MIN  
  cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_FileDirective\_KEEP\_ALIVE  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FileDirective\_METADATA  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FileDirective\_NAK  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FileDirective\_PROMPT  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FileDirective\_t  
  cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_FileDirective\_DISPATCH\_TABLE\_t  
  cf\_cfdp\_pdu.h, 958  
CF\_CFDP\_FinDeliveryCode\_COMPLETE  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinDeliveryCode\_INCOMPLETE  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinDeliveryCode\_INVALID  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinDeliveryCode\_t  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FindUnusedChunks  
  cf\_cfdp.c, 860  
CF\_CFDP\_FinFileStatus\_DISCARDED  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinFileStatus\_DISCARDED\_FILESTORE  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinFileStatus\_INVALID  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinFileStatus\_RETAINED  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinFileStatus\_t  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinFileStatus\_UNREPORTED  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_FinishTransaction  
  cf\_cfdp.c, 861  
  cf\_cfdp.h, 913  
CF\_CFDP\_GetAckTxnStatus  
  cf\_utils.c, 1219  
  cf\_utils.h, 1240  
CF\_CFDP\_GetClass  
  cf\_cfdp.c, 862  
CF\_CFDP\_GetMoveTarget  
  cf\_cfdp.c, 862  
  cf\_cfdp.h, 914  
CF\_CFDP\_GetPrintClass  
  cf\_cfdp.h, 915  
CF\_CFDP\_GetTempName  
  cf\_cfdp.c, 863  
  cf\_cfdp.h, 916  
CF\_CFDP\_GetTxnStatus  
  cf\_cfdp.c, 863  
  cf\_cfdp.h, 916  
CF\_CFDP\_GetValueEncodedSize  
  cf\_codec.c, 1162  
  cf\_codec.h, 1196  
CF\_CFDP\_IDLE\_MD\_ERR\_EID  
  CFS CFDP Event IDs, 159  
CF\_CFDP\_InitEngine  
  cf\_cfdp.c, 864  
  cf\_cfdp.h, 916  
CF\_CFDP\_InitTxnTxFile  
  cf\_cfdp.c, 865

cf\_cfdp.h, 918  
CF\_CFDP\_INVALID\_DST\_ERR\_EID  
    CFS CFDP Event IDs, 160  
CF\_CFDP\_IsSender  
    cf\_cfdp.c, 866  
CF\_CFDP\_Inv, 527  
    length, 528  
CF\_CFDP\_Inv\_t  
    cf\_cfdp\_pdu.h, 956  
CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID  
    CFS CFDP Event IDs, 160  
CF\_CFDP\_MAX\_HEADER\_SIZE  
    cf\_cfdp\_pdu.h, 956  
CF\_CFDP\_MIN\_HEADER\_SIZE  
    cf\_cfdp\_pdu.h, 956  
CF\_CFDP\_MsgOutGet  
    cf\_cfdp\_sbintf.c, 1028  
    cf\_cfdp\_sbintf.h, 1032  
CF\_CFDP\_NO\_CHUNKLIST\_AVAIL\_EID  
    CFS CFDP Event IDs, 160  
CF\_CFDP\_NO\_MSG\_ERR\_EID  
    CFS CFDP Event IDs, 160  
CF\_CFDP\_OPENDIR\_ERR\_EID  
    CFS CFDP Event IDs, 161  
cf\_cfdp\_pdu.h  
    CF\_APP\_MAX\_HEADER\_SIZE, 956  
    CF\_CFDP\_AckTxnStatus\_ACTIVE, 958  
    CF\_CFDP\_AckTxnStatus\_INVALID, 958  
    CF\_CFDP\_AckTxnStatus\_t, 957  
    CF\_CFDP\_AckTxnStatus\_TERMINATED, 958  
    CF\_CFDP\_AckTxnStatus\_UNDEFINED, 958  
    CF\_CFDP\_AckTxnStatus\_UNRECOGNIZED, 958  
    CF\_CFDP\_ConditionCode\_CANCEL\_REQUEST\_RECEIVED, 958  
    CF\_CFDP\_ConditionCode\_CHECK\_LIMIT\_REACHED, 958  
    CF\_CFDP\_ConditionCode\_FILE\_CHECKSUM\_FAILURE, 958  
    CF\_CFDP\_ConditionCode\_FILE\_SIZE\_ERROR, 958  
    CF\_CFDP\_ConditionCode\_FILESTORE\_REJECTION, 958  
    CF\_CFDP\_ConditionCode\_INACTIVITY\_DETECTED, 958  
    CF\_CFDP\_ConditionCode\_INVALID\_FILE\_STRUCTURE, 958  
    CF\_CFDP\_ConditionCode\_INVALID\_TRANSMISSION\_MODE, 958  
    CF\_CFDP\_ConditionCode\_KEEP\_ALIVE\_LIMIT\_REACHED, 958  
    CF\_CFDP\_ConditionCode\_NAK\_LIMIT\_REACHED, 958  
    CF\_CFDP\_ConditionCode\_NO\_ERROR, 958  
    CF\_CFDP\_ConditionCode\_POS\_ACK\_LIMIT\_REACHED, 958  
        cc\_and\_transaction\_status, 528  
958  
CF\_CFDP\_ConditionCode\_SUSPEND\_REQUEST\_RECEIVED, 958  
CF\_CFDP\_ConditionCode\_t, 958  
CF\_CFDP\_ConditionCode\_UNSUPPORTED\_CHECKSUM\_TYPE, 958  
CF\_CFDP\_FileDirective\_ACK, 958  
CF\_CFDP\_FileDirective\_EOF, 958  
CF\_CFDP\_FileDirective\_FIN, 958  
CF\_CFDP\_FileDirective\_INVALID\_MAX, 959  
CF\_CFDP\_FileDirective\_INVALID\_MIN, 958  
CF\_CFDP\_FileDirective\_KEEP\_ALIVE, 959  
CF\_CFDP\_FileDirective\_METADATA, 959  
CF\_CFDP\_FileDirective\_NAK, 959  
CF\_CFDP\_FileDirective\_PROMPT, 959  
CF\_CFDP\_FileDirective\_t, 958  
CF\_CFDP\_FinDeliveryCode\_COMPLETE, 959  
CF\_CFDP\_FinDeliveryCode\_INCOMPLETE, 959  
CF\_CFDP\_FinDeliveryCode\_INVALID, 959  
CF\_CFDP\_FinDeliveryCode\_t, 959  
CF\_CFDP\_FinFileStatus\_DISCARDED, 959  
CF\_CFDP\_FinFileStatus\_DISCARDED\_FILESTORE, 959  
CF\_CFDP\_FinFileStatus\_INVALID, 959  
CF\_CFDP\_FinFileStatus\_RETAINED, 959  
CF\_CFDP\_FinFileStatus\_t, 959  
CF\_CFDP\_FinFileStatus\_UNREPORTED, 959  
CF\_CFDP\_Inv\_t, 956  
CF\_CFDP\_MAX\_HEADER\_SIZE, 956  
CF\_CFDP\_MIN\_HEADER\_SIZE, 956  
CF\_CFDP\_PduAck\_t, 956  
CF\_CFDP\_PduEof\_t, 956  
CF\_CFDP\_PduFileContent\_t, 956  
CF\_CFDP\_PduFileHeader\_t, 957  
CF\_CFDP\_PduDirectiveHeader\_t, 957  
CF\_CFDP\_PduFin\_t, 957  
CF\_CFDP\_PduHeader\_t, 957  
CF\_CFDP\_PduMd\_t, 957  
CF\_CFDP\_PduNak\_t, 957  
CF\_CFDP\_SegmentRequest\_t, 957  
CF\_CFDP\_Tlv\_t, 957  
CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID, 960  
CF\_CFDP\_TLV\_TYPE\_FAULT\_HANDLER\_OVERRIDE, 959  
CF\_CFDP\_TLV\_TYPE\_FILESTORE\_REQUEST, 959  
CF\_CFDP\_TLV\_TYPE\_FILESTORE\_RESPONSE, 959  
CF\_CFDP\_TLV\_TYPE\_FLOW\_LABEL, 960  
CF\_CFDP\_TLV\_TYPE\_INVALID\_MAX, 960  
CF\_CFDP\_TLV\_TYPE\_MESSAGE\_TO\_USER, 959  
CF\_CFDP\_TlvType\_t, 959

directive\_and\_subtype\_code, 528  
CF\_CFDP\_PduAck\_CC  
  cf\_codec.c, 1165  
CF\_CFDP\_PduAck\_DIR\_CODE  
  cf\_codec.c, 1165  
CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE  
  cf\_codec.c, 1165  
CF\_CFDP\_PduAck\_t  
  cf\_cfdp\_pdu.h, 956  
CF\_CFDP\_PduAck\_TRANSACTION\_STATUS  
  cf\_codec.c, 1166  
CF\_CFDP\_PduEof, 528  
  cc, 529  
  crc, 529  
  size, 529  
CF\_CFDP\_PduEof\_FLAGS\_CC  
  cf\_codec.c, 1166  
CF\_CFDP\_PduEof\_t  
  cf\_cfdp\_pdu.h, 956  
CF\_CFDP\_PduFileData\_RECORD\_CONTINUATION\_STATE  
  cf\_codec.c, 1166  
CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH  
  cf\_codec.c, 1166  
CF\_CFDP\_PduFileDataContent, 529  
  data, 529  
CF\_CFDP\_PduFileDataContent\_t  
  cf\_cfdp\_pdu.h, 956  
CF\_CFDP\_PduFileDataHeader, 530  
  offset, 530  
CF\_CFDP\_PduFileDataHeader\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_PduFileDirectiveHeader, 530  
  directive\_code, 530  
CF\_CFDP\_PduFileDirectiveHeader\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_PduFin, 530  
  flags, 531  
CF\_CFDP\_PduFin\_FLAGS\_CC  
  cf\_codec.c, 1166  
CF\_CFDP\_PduFin\_FLAGS\_DELIVERY\_CODE  
  cf\_codec.c, 1166  
CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS  
  cf\_codec.c, 1166  
CF\_CFDP\_PduFin\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_PduHeader, 531  
  eid\_tsn\_lengths, 531  
  flags, 532  
  length, 532  
CF\_CFDP\_PduHeader\_FLAGS\_CRC  
  cf\_codec.c, 1166  
CF\_CFDP\_PduHeader\_FLAGS\_DIR  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_FLAGS\_LARGEFILE  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_FLAGS\_MODE  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_FLAGS\_TYPE  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_FLAGS\_VERSION  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_LENGTHS\_ENTITY  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_LENGTHS\_TRANSACTION\_SEQUENCE  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_SEGMENT\_METADATA  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_SEGMENTATION\_CONTROL  
  cf\_codec.c, 1167  
CF\_CFDP\_PduHeader\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_PduMd, 532  
  segmentation\_control, 532  
  size, 532  
CF\_CFDP\_PduMd\_CHECKSUM\_TYPE  
  cf\_codec.c, 1168  
CF\_CFDP\_PduMd\_CLOSURE\_REQUESTED  
  cf\_codec.c, 1168  
CF\_CFDP\_PduMd\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_PduNak, 533  
  scope\_end, 533  
  scope\_start, 533  
CF\_CFDP\_PduNak\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_PlaybackDir  
  cf\_cfdp.c, 866  
  cf\_cfdp.h, 919  
CF\_CFDP\_PlaybackDir\_Initiate  
  cf\_cfdp.c, 867  
CF\_CFDP\_ProcessPlaybackDirectories  
  cf\_cfdp.c, 867  
CF\_CFDP\_ProcessPlaybackDirectory  
  cf\_cfdp.c, 868  
  cf\_cfdp.h, 920  
CF\_CFDP\_ProcessPollingDirectories  
  cf\_cfdp.c, 869  
  cf\_cfdp.h, 921  
cf\_cfdp\_r\_c  
  CF\_CFDP\_R1\_Recv, 961  
  CF\_CFDP\_R2\_GapCompute, 962  
  CF\_CFDP\_R2\_Recv, 962  
  CF\_CFDP\_R2\_SubstateRecvFinAck, 963  
  CF\_CFDP\_R\_AckTimerTick, 964  
  CF\_CFDP\_R\_CalcCrcChunk, 965  
  CF\_CFDP\_R\_CalcCrcStart, 966  
  CF\_CFDP\_R\_CheckComplete, 967  
  CF\_CFDP\_R\_CheckCrc, 968

CF\_CFDP\_R\_CheckState, [968](#)  
 CF\_CFDP\_R\_CheckState\_DATA\_EOF, [969](#)  
 CF\_CFDP\_R\_CheckState\_DATA\_NORMAL, [970](#)  
 CF\_CFDP\_R\_CheckState\_FILESTORE, [970](#)  
 CF\_CFDP\_R\_CheckState\_FINACK, [971](#)  
 CF\_CFDP\_R\_CheckState\_VALIDATE, [971](#)  
 CF\_CFDP\_R\_HandleFileRetention, [972](#)  
 CF\_CFDP\_R\_Init, [973](#)  
 CF\_CFDP\_R\_ProcessFd, [974](#)  
 CF\_CFDP\_R\_SendNak, [975](#)  
 CF\_CFDP\_R\_SubstateRecvEof, [976](#)  
 CF\_CFDP\_R\_SubstateRecvFileData, [977](#)  
 CF\_CFDP\_R\_SubstateRecvMd, [977](#)  
 CF\_CFDP\_R\_Tick, [978](#)  
 CF\_CFDP\_R\_Tick\_Maintenance, [979](#)  
 cf\_cfdp\_r.h  
     CF\_CFDP\_R1\_Recv, [981](#)  
     CF\_CFDP\_R2\_GapCompute, [982](#)  
     CF\_CFDP\_R2\_Recv, [983](#)  
     CF\_CFDP\_R2\_SubstateRecvFinAck, [983](#)  
     CF\_CFDP\_R\_AckTimerTick, [984](#)  
     CF\_CFDP\_R\_CalcCrcChunk, [985](#)  
     CF\_CFDP\_R\_CalcCrcStart, [986](#)  
     CF\_CFDP\_R\_CheckComplete, [986](#)  
     CF\_CFDP\_R\_CheckCrc, [987](#)  
     CF\_CFDP\_R\_CheckState, [988](#)  
     CF\_CFDP\_R\_HandleFileRetention, [989](#)  
     CF\_CFDP\_R\_Init, [990](#)  
     CF\_CFDP\_R\_ProcessFd, [991](#)  
     CF\_CFDP\_R\_SendNak, [992](#)  
     CF\_CFDP\_R\_SubstateRecvEof, [993](#)  
     CF\_CFDP\_R\_SubstateRecvFileData, [994](#)  
     CF\_CFDP\_R\_SubstateRecvMd, [994](#)  
     CF\_CFDP\_R\_Tick, [995](#)  
     CF\_CFDP\_R\_Tick\_Maintenance, [996](#)  
 CF\_CFDP\_R1\_Recv  
     cf\_cfdp\_r.c, [961](#)  
     cf\_cfdp\_r.h, [981](#)  
 CF\_CFDP\_R2\_GapCompute  
     cf\_cfdp\_r.c, [962](#)  
     cf\_cfdp\_r.h, [982](#)  
 CF\_CFDP\_R2\_Recv  
     cf\_cfdp\_r.c, [962](#)  
     cf\_cfdp\_r.h, [983](#)  
 CF\_CFDP\_R2\_SubstateRecvFinAck  
     cf\_cfdp\_r.c, [963](#)  
     cf\_cfdp\_r.h, [983](#)  
 CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_EID  
     CFS CFDP Event IDs, [161](#)  
 CF\_CFDP\_R\_AckTimerTick  
     cf\_cfdp\_r.c, [964](#)  
     cf\_cfdp\_r.h, [984](#)  
 CF\_CFDP\_R\_CalcCrcChunk  
     cf\_cfdp\_r.c, [965](#)  
     cf\_cfdp\_r.h, [985](#)  
     cf\_cfdp\_r.c, [986](#)  
     cf\_cfdp\_r.h, [986](#)  
 CF\_CFDP\_R\_CalcCrcStart  
     cf\_cfdp\_r.c, [966](#)  
     cf\_cfdp\_r.h, [986](#)  
 CF\_CFDP\_R\_CheckComplete  
     cf\_cfdp\_r.c, [967](#)  
     cf\_cfdp\_r.h, [986](#)  
 CF\_CFDP\_R\_CheckCrc  
     cf\_cfdp\_r.c, [968](#)  
     cf\_cfdp\_r.h, [987](#)  
 CF\_CFDP\_R\_CheckState  
     cf\_cfdp\_r.c, [968](#)  
     cf\_cfdp\_r.h, [988](#)  
 CF\_CFDP\_R\_CheckState\_DATA\_EOF  
     cf\_cfdp\_r.c, [969](#)  
 CF\_CFDP\_R\_CheckState\_DATA\_NORMAL  
     cf\_cfdp\_r.c, [970](#)  
 CF\_CFDP\_R\_CheckState\_FILESTORE  
     cf\_cfdp\_r.c, [970](#)  
 CF\_CFDP\_R\_CheckState\_FINACK  
     cf\_cfdp\_r.c, [971](#)  
 CF\_CFDP\_R\_CheckState\_VALIDATE  
     cf\_cfdp\_r.c, [971](#)  
 CF\_CFDP\_R\_CRC\_ERR\_EID  
     CFS CFDP Event IDs, [161](#)  
 CF\_CFDP\_R\_CREAT\_ERR\_EID  
     CFS CFDP Event IDs, [161](#)  
 CF\_CFDP\_R\_DC\_INV\_ERR\_EID  
     CFS CFDP Event IDs, [162](#)  
 CF\_CFDP\_R\_DispatchRecv  
     cf\_cfdp\_dispatch.c, [948](#)  
     cf\_cfdp\_dispatch.h, [951](#)  
 CF\_CFDP\_R\_EOF\_MD\_SIZE\_ERR\_EID  
     CFS CFDP Event IDs, [162](#)  
 CF\_CFDP\_R\_FILE\_RETAINED\_EID  
     CFS CFDP Event IDs, [162](#)  
 CF\_CFDP\_R\_HandleFileRetention  
     cf\_cfdp\_r.c, [972](#)  
     cf\_cfdp\_r.h, [989](#)  
 CF\_CFDP\_R\_INACT\_TIMER\_ERR\_EID  
     CFS CFDP Event IDs, [162](#)  
 CF\_CFDP\_R\_Init  
     cf\_cfdp\_r.c, [973](#)  
     cf\_cfdp\_r.h, [990](#)  
 CF\_CFDP\_R\_NAK\_LIMIT\_ERR\_EID  
     CFS CFDP Event IDs, [163](#)  
 CF\_CFDP\_R\_NOT\_RETAINED\_EID  
     CFS CFDP Event IDs, [163](#)  
 CF\_CFDP\_R\_PDU\_EOF\_ERR\_EID  
     CFS CFDP Event IDs, [163](#)  
 CF\_CFDP\_R\_PDU\_FINACK\_ERR\_EID  
     CFS CFDP Event IDs, [163](#)  
 CF\_CFDP\_R\_ProcessFd  
     cf\_cfdp\_r.c, [974](#)

cf\_cfdp\_r.h, 991  
CF\_CFDP\_R\_READ\_ERR\_EID  
    CFS CFDP Event IDs, 164  
CF\_CFDP\_R\_RENAME\_ERR\_EID  
    CFS CFDP Event IDs, 164  
CF\_CFDP\_R\_REQUEST\_MD\_INF\_EID  
    CFS CFDP Event IDs, 164  
CF\_CFDP\_R\_SEEK\_CRC\_ERR\_EID  
    CFS CFDP Event IDs, 164  
CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID  
    CFS CFDP Event IDs, 165  
CF\_CFDP\_R\_SendNak  
    cf\_cfdp\_r.c, 975  
    cf\_cfdp\_r.h, 992  
CF\_CFDP\_R\_SIZE\_MISMATCH\_ERR\_EID  
    CFS CFDP Event IDs, 165  
CF\_CFDP\_R\_SubstateDispatchTable\_t, 533  
    state, 533  
CF\_CFDP\_R\_SubstateRecvEof  
    cf\_cfdp\_r.c, 976  
    cf\_cfdp\_r.h, 993  
CF\_CFDP\_R\_SubstateRecvFileData  
    cf\_cfdp\_r.c, 977  
    cf\_cfdp\_r.h, 994  
CF\_CFDP\_R\_SubstateRecvMd  
    cf\_cfdp\_r.c, 977  
    cf\_cfdp\_r.h, 994  
CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID  
    CFS CFDP Event IDs, 165  
CF\_CFDP\_R\_Tick  
    cf\_cfdp\_r.c, 978  
    cf\_cfdp\_r.h, 995  
CF\_CFDP\_R\_Tick\_Maintenance  
    cf\_cfdp\_r.c, 979  
    cf\_cfdp\_r.h, 996  
CF\_CFDP\_R\_WRITE\_ERR\_EID  
    CFS CFDP Event IDs, 165  
CF\_CFDP\_ReceiveMessage  
    cf\_cfdp\_sbintf.c, 1029  
    cf\_cfdp\_sbintf.h, 1033  
CF\_CFDP\_ReceivePdu  
    cf\_cfdp.c, 870  
    cf\_cfdp.h, 922  
CF\_CFDP\_RecvAck  
    cf\_cfdp.c, 870  
    cf\_cfdp.h, 922  
CF\_CFDP\_RecvDrop  
    cf\_cfdp.c, 871  
    cf\_cfdp.h, 923  
CF\_CFDP\_RecvEof  
    cf\_cfdp.c, 872  
    cf\_cfdp.h, 924  
CF\_CFDP\_RecvFd  
    cf\_cfdp.c, 873  
                cf\_cfdp.h, 925  
CF\_CFDP\_RecvFin  
    cf\_cfdp.c, 874  
    cf\_cfdp.h, 926  
CF\_CFDP\_RecvHold  
    cf\_cfdp.c, 874  
    cf\_cfdp.h, 926  
CF\_CFDP\_RecvInit  
    cf\_cfdp.c, 875  
    cf\_cfdp.h, 927  
CF\_CFDP\_RecvMd  
    cf\_cfdp.c, 876  
    cf\_cfdp.h, 928  
CF\_CFDP\_RecvNak  
    cf\_cfdp.c, 877  
    cf\_cfdp.h, 929  
CF\_CFDP\_RecvPh  
    cf\_cfdp.c, 878  
    cf\_cfdp.h, 930  
CF\_CFDP\_RecycleTransaction  
    cf\_cfdp.c, 879  
    cf\_cfdp.h, 931  
CF\_CFDP\_RX\_DROPPED\_ERR\_EID  
    CFS CFDP Event IDs, 166  
CF\_CFDP\_RxStateDispatch  
    cf\_cfdp\_dispatch.c, 948  
    cf\_cfdp\_dispatch.h, 952  
cf\_cfdp\_s.c  
    CF\_CFDP\_S1\_CheckState\_DATA\_EOF, 998  
    CF\_CFDP\_S1\_Recv, 999  
    CF\_CFDP\_S2\_CheckState\_DATA\_EOF, 999  
    CF\_CFDP\_S2\_Recv, 1000  
    CF\_CFDP\_S2\_SubstateEofAck, 1000  
    CF\_CFDP\_S2\_SubstateNak, 1001  
    CF\_CFDP\_S\_AckTimerTick, 1002  
    CF\_CFDP\_S\_CheckState, 1003  
    CF\_CFDP\_S\_CheckState\_DATA\_EOF, 1004  
    CF\_CFDP\_S\_CheckState\_DATA\_NORMAL, 1004  
    CF\_CFDP\_S\_CheckState\_FILESTORE, 1005  
    CF\_CFDP\_S\_HandleFileRetention, 1005  
    CF\_CFDP\_S\_Init, 1006  
    CF\_CFDP\_S\_SendFileData, 1007  
    CF\_CFDP\_S\_SubstateEarlyFin, 1008  
    CF\_CFDP\_S\_SubstateRecvFin, 1009  
    CF\_CFDP\_S\_SubstateSendFileData, 1009  
    CF\_CFDP\_S\_Tick, 1010  
    CF\_CFDP\_S\_Tick\_Maintenance, 1011  
    CF\_CFDP\_S\_Tick\_Nak, 1012  
cf\_cfdp\_s.h  
    CF\_CFDP\_S1\_Recv, 1014  
    CF\_CFDP\_S2\_Recv, 1014  
    CF\_CFDP\_S2\_SubstateEofAck, 1015  
    CF\_CFDP\_S2\_SubstateNak, 1016  
    CF\_CFDP\_S\_AckTimerTick, 1017

CF\_CFDP\_S\_CheckState, [1017](#)  
 CF\_CFDP\_S\_HandleFileRetention, [1018](#)  
 CF\_CFDP\_S\_Init, [1019](#)  
 CF\_CFDP\_S\_SendFileData, [1020](#)  
 CF\_CFDP\_S\_SubstateEarlyFin, [1021](#)  
 CF\_CFDP\_S\_SubstateRecvFin, [1022](#)  
 CF\_CFDP\_S\_SubstateSendFileData, [1023](#)  
 CF\_CFDP\_S\_Tick, [1023](#)  
 CF\_CFDP\_S\_Tick\_Maintenance, [1025](#)  
 CF\_CFDP\_S\_Tick\_Nak, [1026](#)  
**CF\_CFDP\_S1\_CheckState\_DATA\_EOF**  
 cf\_cfdp\_s.c, [998](#)  
**CF\_CFDP\_S1\_Recv**  
 cf\_cfdp\_s.c, [999](#)  
 cf\_cfdp\_s.h, [1014](#)  
**CF\_CFDP\_S2\_CheckState\_DATA\_EOF**  
 cf\_cfdp\_s.c, [999](#)  
**CF\_CFDP\_S2\_Recv**  
 cf\_cfdp\_s.c, [1000](#)  
 cf\_cfdp\_s.h, [1014](#)  
**CF\_CFDP\_S2\_SubstateEofAck**  
 cf\_cfdp\_s.c, [1000](#)  
 cf\_cfdp\_s.h, [1015](#)  
**CF\_CFDP\_S2\_SubstateNak**  
 cf\_cfdp\_s.c, [1001](#)  
 cf\_cfdp\_s.h, [1016](#)  
**CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID**  
 CFS CFDP Event IDs, [166](#)  
**CF\_CFDP\_S\_AckTimerTick**  
 cf\_cfdp\_s.c, [1002](#)  
 cf\_cfdp\_s.h, [1017](#)  
**CF\_CFDP\_S\_ALREADY\_OPEN\_ERR\_EID**  
 CFS CFDP Event IDs, [166](#)  
**CF\_CFDP\_S\_CheckState**  
 cf\_cfdp\_s.c, [1003](#)  
 cf\_cfdp\_s.h, [1017](#)  
**CF\_CFDP\_S\_CheckState\_DATA\_EOF**  
 cf\_cfdp\_s.c, [1004](#)  
**CF\_CFDP\_S\_CheckState\_DATA\_NORMAL**  
 cf\_cfdp\_s.c, [1004](#)  
**CF\_CFDP\_S\_CheckState\_FILESTORE**  
 cf\_cfdp\_s.c, [1005](#)  
**CF\_CFDP\_S\_DC\_INV\_ERR\_EID**  
 CFS CFDP Event IDs, [166](#)  
**CF\_CFDP\_S\_DispatchRecv**  
 cf\_cfdp\_dispatch.c, [949](#)  
 cf\_cfdp\_dispatch.h, [952](#)  
**CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID**  
 CFS CFDP Event IDs, [167](#)  
**CF\_CFDP\_S\_FILE\_MOVED\_EID**  
 CFS CFDP Event IDs, [167](#)  
**CF\_CFDP\_S\_FILE\_REMOVED\_EID**  
 CFS CFDP Event IDs, [167](#)  
**CF\_CFDP\_S\_HandleFileRetention**  
 cf\_cfdp\_s.c, [1005](#)  
 cf\_cfdp\_s.h, [1018](#)  
**CF\_CFDP\_S\_INACT\_TIMER\_ERR\_EID**  
 CFS CFDP Event IDs, [167](#)  
**CF\_CFDP\_S\_Init**  
 cf\_cfdp\_s.c, [1006](#)  
 cf\_cfdp\_s.h, [1019](#)  
**CF\_CFDP\_S\_INVALID\_SR\_ERR\_EID**  
 CFS CFDP Event IDs, [168](#)  
**CF\_CFDP\_S\_NON\_FD\_PDU\_ERR\_EID**  
 CFS CFDP Event IDs, [168](#)  
**CF\_CFDP\_S\_OPEN\_ERR\_EID**  
 CFS CFDP Event IDs, [168](#)  
**CF\_CFDP\_S\_PDU\_EOF\_ERR\_EID**  
 CFS CFDP Event IDs, [168](#)  
**CF\_CFDP\_S\_PDU\_NAK\_ERR\_EID**  
 CFS CFDP Event IDs, [169](#)  
**CF\_CFDP\_S\_READ\_ERR\_EID**  
 CFS CFDP Event IDs, [169](#)  
**CF\_CFDP\_S\_SEEK\_BEG\_ERR\_EID**  
 CFS CFDP Event IDs, [169](#)  
**CF\_CFDP\_S\_SEEK\_END\_ERR\_EID**  
 CFS CFDP Event IDs, [169](#)  
**CF\_CFDP\_S\_SEEK\_FD\_ERR\_EID**  
 CFS CFDP Event IDs, [170](#)  
**CF\_CFDP\_S\_SEND\_FD\_ERR\_EID**  
 CFS CFDP Event IDs, [170](#)  
**CF\_CFDP\_S\_SEND\_MD\_ERR\_EID**  
 CFS CFDP Event IDs, [170](#)  
**CF\_CFDP\_S\_SendFileData**  
 cf\_cfdp\_s.c, [1007](#)  
 cf\_cfdp\_s.h, [1020](#)  
**CF\_CFDP\_S\_START\_SEND\_INF\_EID**  
 CFS CFDP Event IDs, [170](#)  
**CF\_CFDP\_S\_SubstateEarlyFin**  
 cf\_cfdp\_s.c, [1008](#)  
 cf\_cfdp\_s.h, [1021](#)  
**CF\_CFDP\_S\_SubstateRecvDispatchTable\_t**, [534](#)  
 substate, [534](#)  
**CF\_CFDP\_S\_SubstateRecvFin**  
 cf\_cfdp\_s.c, [1009](#)  
 cf\_cfdp\_s.h, [1022](#)  
**CF\_CFDP\_S\_SubstateSendDispatchTable\_t**, [534](#)  
 substate, [534](#)  
**CF\_CFDP\_S\_SubstateSendFileData**  
 cf\_cfdp\_s.c, [1009](#)  
 cf\_cfdp\_s.h, [1023](#)  
**CF\_CFDP\_S\_Tick**  
 cf\_cfdp\_s.c, [1010](#)  
 cf\_cfdp\_s.h, [1023](#)  
**CF\_CFDP\_S\_Tick\_Maintenance**  
 cf\_cfdp\_s.c, [1011](#)  
 cf\_cfdp\_s.h, [1025](#)  
**CF\_CFDP\_S\_Tick\_Nak**

cf\_cfdp\_s.c, 1012  
cf\_cfdp\_s.h, 1026  
CF\_CFDP\_S\_Tick\_NewData  
  cf\_cfdp.c, 880  
  cf\_cfdp.h, 932  
cf\_cfdp\_sbintf.c  
  CF\_CFDP\_MsgOutGet, 1028  
  CF\_CFDP\_ReceiveMessage, 1029  
  CF\_CFDP\_Send, 1030  
cf\_cfdp\_sbintf.h  
  CF\_CFDP\_MsgOutGet, 1032  
  CF\_CFDP\_ReceiveMessage, 1033  
  CF\_CFDP\_Send, 1034  
  CF\_PduCmdMsg\_t, 1032  
  CF\_PduTlmMsg\_t, 1032  
CF\_CFDP\_SegmentRequest, 535  
  offset\_end, 535  
  offset\_start, 535  
CF\_CFDP\_SegmentRequest\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_Send  
  cf\_cfdp\_sbintf.c, 1030  
  cf\_cfdp\_sbintf.h, 1034  
CF\_CFDP\_SendAck  
  cf\_cfdp.c, 881  
  cf\_cfdp.h, 933  
CF\_CFDP\_SendEof  
  cf\_cfdp.c, 882  
  cf\_cfdp.h, 934  
CF\_CFDP\_SendEotPkt  
  cf\_cfdp.c, 883  
  cf\_cfdp.h, 935  
CF\_CFDP\_SendFd  
  cf\_cfdp.c, 884  
  cf\_cfdp.h, 936  
CF\_CFDP\_SendFin  
  cf\_cfdp.c, 885  
  cf\_cfdp.h, 937  
CF\_CFDP\_SendMd  
  cf\_cfdp.c, 886  
  cf\_cfdp.h, 938  
CF\_CFDP\_SendNak  
  cf\_cfdp.c, 887  
  cf\_cfdp.h, 939  
CF\_CFDP\_SetPduLength  
  cf\_cfdp.c, 888  
CF\_CFDP\_SetTxnStatus  
  cf\_cfdp.c, 888  
  cf\_cfdp.h, 940  
CF\_CFDP\_SetupRxTransaction  
  cf\_cfdp.c, 889  
  cf\_cfdp.h, 940  
CF\_CFDP\_SetupTxTransaction  
  cf\_cfdp.c, 890  
cf\_cfdp.h, 941  
CF\_CFDP\_StartFirstPending  
  cf\_cfdp.c, 891  
  cf\_cfdp.h, 942  
CF\_CFDP\_StartRxTransaction  
  cf\_cfdp.c, 892  
  cf\_cfdp.h, 943  
CF\_CFDP\_StateRecvFunc\_t  
  cf\_cfdp\_dispatch.h, 950  
CF\_CFDP\_StateSendFunc\_t  
  cf\_cfdp\_dispatch.h, 951  
CF\_CFDP\_Tick\_args, 535  
  chan, 536  
  fn, 536  
  resume\_point, 536  
CF\_CFDP\_Tick\_args\_t  
  cf\_cfdp.h, 900  
CF\_CFDP\_TickTransactions  
  cf\_cfdp.c, 893  
  cf\_cfdp.h, 944  
CF\_CFDP\_tlv, 536  
  length, 536  
  type, 537  
CF\_CFDP\_tlv\_t  
  cf\_cfdp\_pdu.h, 957  
CF\_CFDP\_TLV\_TYPE\_ENTITY\_ID  
  cf\_cfdp\_pdu.h, 960  
CF\_CFDP\_TLV\_TYPE\_FAULT\_HANDLER\_OVERRIDE  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_TLV\_TYPE\_FILESTORE\_REQUEST  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_TLV\_TYPE\_FILESTORE\_RESPONSE  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_TLV\_TYPE\_FLOW\_LABEL  
  cf\_cfdp\_pdu.h, 960  
CF\_CFDP\_TLV\_TYPE\_INVALID\_MAX  
  cf\_cfdp\_pdu.h, 960  
CF\_CFDP\_TLV\_TYPE\_MESSAGE\_TO\_USER  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_TlvType\_t  
  cf\_cfdp\_pdu.h, 959  
CF\_CFDP\_TxFile  
  cf\_cfdp.c, 894  
  cf\_cfdp.h, 945  
CF\_CFDP\_TxFile\_Initiate  
  cf\_cfdp.c, 895  
CF\_CFDP\_TxnlsOK  
  cf\_cfdp.c, 896  
  cf\_cfdp.h, 946  
CF\_CFDP\_TxnRecvDispatchTable\_t, 537  
  rx, 537  
CF\_CFDP\_TxnSendDispatchTable\_t, 537  
  tx, 538  
cf\_cfdp\_types.h

CF\_Channel\_t, [1038](#)  
 CF\_ChunkWrapper\_t, [1039](#)  
 CF\_Direction\_NUM, [1040](#)  
 CF\_Direction\_RX, [1040](#)  
 CF\_Direction\_t, [1040](#)  
 CF\_Direction\_TX, [1040](#)  
 CF\_Engine\_t, [1039](#)  
 CF\_Flags\_Common\_t, [1039](#)  
 CF\_Flags\_Rx\_t, [1039](#)  
 CF\_Flags\_Tx\_t, [1039](#)  
 CF\_History\_t, [1039](#)  
 CF\_Input\_t, [1039](#)  
 CF\_NUM\_CHUNKS\_ALL\_CHANNELS, [1038](#)  
 CF\_NUM\_HISTORIES, [1038](#)  
 CF\_NUM\_TRANSACTIONS, [1038](#)  
 CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL, [1038](#)  
 CF\_Output\_t, [1039](#)  
 CF\_Playback\_t, [1039](#)  
 CF\_Poll\_t, [1039](#)  
 CF\_RxSubState\_COMPLETE, [1040](#)  
 CF\_RxSubState\_DATA\_EOF, [1040](#)  
 CF\_RxSubState\_DATA\_NORMAL, [1040](#)  
 CF\_RxSubState\_FILESTORE, [1040](#)  
 CF\_RxSubState\_FINACK, [1040](#)  
 CF\_RxSubState\_NUM\_STATES, [1040](#)  
 CF\_RxSubState\_t, [1040](#)  
 CF\_RxSubState\_VALIDATE, [1040](#)  
 CF\_StateData\_t, [1039](#)  
 CF\_StateFlags\_t, [1039](#)  
 CF\_TickState\_COMPLETE, [1041](#)  
 CF\_TickState\_INIT, [1040](#)  
 CF\_TickState\_NUM\_TYPES, [1041](#)  
 CF\_TickState\_RX\_STATE, [1040](#)  
 CF\_TickState\_t, [1040](#)  
 CF\_TickState\_TX\_FILEDATA, [1040](#)  
 CF\_TickState\_TX\_NAK, [1040](#)  
 CF\_TickState\_TX\_PEND, [1041](#)  
 CF\_TickState\_TX\_STATE, [1040](#)  
 CF\_Transaction\_t, [1040](#)  
 CF\_TxnState\_DROP, [1041](#)  
 CF\_TxnState\_HOLD, [1041](#)  
 CF\_TxnState\_INIT, [1041](#)  
 CF\_TxnState\_INVALID, [1041](#)  
 CF\_TxnState\_R1, [1041](#)  
 CF\_TxnState\_R2, [1041](#)  
 CF\_TxnState\_S1, [1041](#)  
 CF\_TxnState\_S2, [1041](#)  
 CF\_TxnState\_t, [1041](#)  
 CF\_TxnState\_UNDEF, [1041](#)  
 CF\_TxnStatus\_ACK\_LIMIT\_NO\_EOF, [1042](#)  
 CF\_TxnStatus\_ACK\_LIMIT\_NO\_FIN, [1042](#)  
 CF\_TxnStatus\_CANCEL\_REQUEST\_RECEIVED, [1042](#)  
 CF\_TxnStatus\_CHECK\_LIMIT\_REACHED, [1041](#)  
 CF\_TxnStatus\_EARLY\_FIN, [1042](#)  
 CF\_TxnStatus\_FILE\_CHECKSUM\_FAILURE, [1041](#)  
 CF\_TxnStatus\_FILE\_SIZE\_ERROR, [1041](#)  
 CF\_TxnStatus\_FILESTORE\_REJECTION, [1041](#)  
 CF\_TxnStatus\_INACTIVITY\_DETECTED, [1041](#)  
 CF\_TxnStatus\_INVALID\_FILE\_STRUCTURE, [1041](#)  
 CF\_TxnStatus\_INVALID\_TRANSMISSION\_MODE, [1041](#)  
 CF\_TxnStatus\_KEEP\_ALIVE\_LIMIT\_REACHED, [1041](#)  
 CF\_TxnStatus\_MAX, [1042](#)  
 CF\_TxnStatus\_NAK\_LIMIT\_REACHED, [1041](#)  
 CF\_TxnStatus\_NAK\_RESPONSE\_ERROR, [1042](#)  
 CF\_TxnStatus\_NO\_ERROR, [1041](#)  
 CF\_TxnStatus\_NO\_RESOURCE, [1042](#)  
 CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED, [1041](#)  
 CF\_TxnStatus\_PROTOCOL\_ERROR, [1042](#)  
 CF\_TxnStatus\_READ\_FAILURE, [1042](#)  
 CF\_TxnStatus\_SEND\_EOF\_FAILURE, [1042](#)  
 CF\_TxnStatus\_SUSPEND\_REQUEST RECEIVED, [1042](#)  
 CF\_TxnStatus\_t, [1041](#)  
 CF\_TxnStatus\_UNDEFINED, [1041](#)  
 CF\_TxnStatus\_UNSUPPORTED\_CHECKSUM\_TYPE, [1042](#)  
 CF\_TxSubState\_COMPLETE, [1042](#)  
 CF\_TxSubState\_DATA\_EOF, [1042](#)  
 CF\_TxSubState\_DATA\_NORMAL, [1042](#)  
 CF\_TxSubState\_FILESTORE, [1042](#)  
 CF\_TxSubState\_NUM\_STATES, [1042](#)  
 CF\_TxSubState\_t, [1042](#)  
 CF\_CFDP\_uint16\_t, [538](#)  
 octets, [538](#)  
 CF\_CFDP\_uint32\_t, [538](#)  
 octets, [539](#)  
 CF\_CFDP\_uint64\_t, [539](#)  
 octets, [539](#)  
 CF\_CFDP\_uint8\_t, [539](#)  
 octets, [539](#)  
 CF\_CFDP\_UpdatePollPbCounted  
     cf\_cfdp.c, [897](#)  
 CF\_CH0\_RX\_MID  
     CFS CFDP Data Interface Message IDs, [229](#)  
 CF\_CH0\_TX\_MID  
     CFS CFDP Data Interface Message IDs, [229](#)  
 CF\_CH1\_RX\_MID  
     CFS CFDP Data Interface Message IDs, [229](#)  
 CF\_CH1\_TX\_MID  
     CFS CFDP Data Interface Message IDs, [229](#)  
 CF\_ChAction\_BooleanArg, [540](#)  
 barg, [540](#)  
 CF\_ChAction\_BooleanArg\_t  
     cf\_cmd.h, [1108](#)  
 CF\_ChAction\_BooleanMsgArg, [540](#)

barg, 540  
data, 540  
CF\_ChanAction\_BoolMsgArg\_t  
  cf\_cmd.h, 1108  
CF\_ChanAction\_MsgArg, 541  
  data, 541  
CF\_ChanAction\_MsgArg\_t  
  cf\_cmd.h, 1108  
CF\_ChanAction\_Status\_ERROR  
  cf\_cmd.h, 1108  
CF\_ChanAction\_Status\_IS\_SUCCESS  
  cf\_cmd.h, 1111  
CF\_ChanAction\_Status\_SUCCESS  
  cf\_cmd.h, 1108  
CF\_ChanAction\_Status\_t  
  cf\_cmd.h, 1108  
CF\_ChanAction\_SuspResArg, 541  
  action, 541  
  same, 541  
CF\_ChanAction\_SuspResArg\_t  
  cf\_cmd.h, 1108  
CF\_ChanActionFn\_t  
  cf\_cmd.h, 1108  
CF\_Channel, 542  
  cs, 542  
  num\_cmd\_tx, 542  
  outgoing\_counter, 542  
  pipe, 542  
  playback, 543  
  poll, 543  
  qs, 543  
  sem\_id, 543  
  tick\_resume, 543  
  tx\_blocked, 543  
CF\_CHANNEL\_NUM\_RX\_CHUNKS\_PER\_TRANSACTION  
  default\_cf\_platform\_cfg.h, 810  
CF\_CHANNEL\_NUM\_TX\_CHUNKS\_PER\_TRANSACTION  
  default\_cf\_platform\_cfg.h, 811  
CF\_CHANNEL\_PIPE\_PREFIX  
  cf\_app.h, 837  
CF\_Channel\_t  
  cf\_cfdp\_types.h, 1038  
CF\_ChannelConfig, 543  
  ack\_limit, 544  
  ack\_timer\_s, 544  
  dequeue\_enabled, 544  
  inactivity\_timer\_s, 544  
  max\_outgoing\_messages\_per\_wakeup, 545  
  mid\_input, 545  
  mid\_output, 545  
  move\_dir, 545  
  nak\_limit, 545  
  nak\_timer\_s, 545  
  pipe\_depth\_input, 545  
polldir, 545  
rx\_max\_messages\_per\_wakeup, 546  
sem\_name, 546  
CF\_ChannelConfig\_t  
  default\_cf\_tbldefs.h, 812  
CF\_CheckTables  
  cf\_app.c, 832  
  cf\_app.h, 840  
CF\_Chunk, 546  
  offset, 546  
  size, 546  
cf\_chunk.c  
  CF\_ChunkList\_ComputeGaps, 1043  
  CF\_ChunkList\_GetFirstChunk, 1044  
  CF\_ChunkList\_RemoveFromFirst, 1044  
  CF\_ChunkListAdd, 1045  
  CF\_ChunkListInit, 1046  
  CF\_ChunkListReset, 1046  
  CF\_Chunks\_CombineNext, 1047  
  CF\_Chunks\_CombinePrevious, 1048  
  CF\_Chunks\_EraseChunk, 1048  
  CF\_Chunks\_EraseRange, 1049  
  CF\_Chunks\_FindInsertPosition, 1049  
  CF\_Chunks\_FindSmallestSize, 1050  
  CF\_Chunks\_Insert, 1050  
  CF\_Chunks\_InsertChunk, 1051  
cf\_chunk.h  
  CF\_Chunk\_MAX, 1054  
  CF\_Chunk\_t, 1053  
  CF\_ChunkIdx\_t, 1053  
  CF\_ChunkList\_ComputeGapFn\_t, 1053  
  CF\_ChunkList\_ComputeGaps, 1054  
  CF\_ChunkList\_GetFirstChunk, 1055  
  CF\_ChunkList\_RemoveFromFirst, 1055  
  CF\_ChunkList\_t, 1053  
  CF\_ChunkListAdd, 1056  
  CF\_ChunkListInit, 1057  
  CF\_ChunkListReset, 1057  
  CF\_ChunkOffset\_t, 1054  
  CF\_Chunks\_CombineNext, 1058  
  CF\_Chunks\_CombinePrevious, 1059  
  CF\_Chunks\_EraseChunk, 1059  
  CF\_Chunks\_EraseRange, 1060  
  CF\_Chunks\_FindInsertPosition, 1060  
  CF\_Chunks\_FindSmallestSize, 1061  
  CF\_Chunks\_Insert, 1061  
  CF\_Chunks\_InsertChunk, 1062  
  CF\_ChunkSize\_t, 1054  
CF\_Chunk\_MAX  
  cf\_chunk.h, 1054  
CF\_Chunk\_t  
  cf\_chunk.h, 1053  
CF\_ChunkIdx\_t  
  cf\_chunk.h, 1053

CF\_ChunkList, 547  
 chunks, 547  
 count, 547  
 max\_chunks, 547  
**CF\_ChunkList\_ComputeGapFn\_t**  
 cf\_chunk.h, 1053  
**CF\_ChunkList\_ComputeGaps**  
 cf\_chunk.c, 1043  
 cf\_chunk.h, 1054  
**CF\_ChunkList\_GetFirstChunk**  
 cf\_chunk.c, 1044  
 cf\_chunk.h, 1055  
**CF\_ChunkList\_RemoveFromFirst**  
 cf\_chunk.c, 1044  
 cf\_chunk.h, 1055  
**CF\_ChunkList\_t**  
 cf\_chunk.h, 1053  
**CF\_ChunkListAdd**  
 cf\_chunk.c, 1045  
 cf\_chunk.h, 1056  
**CF\_ChunkListInit**  
 cf\_chunk.c, 1046  
 cf\_chunk.h, 1057  
**CF\_ChunkListReset**  
 cf\_chunk.c, 1046  
 cf\_chunk.h, 1057  
**CF\_ChunkOffset\_t**  
 cf\_chunk.h, 1054  
**CF\_Chunks\_CombineNext**  
 cf\_chunk.c, 1047  
 cf\_chunk.h, 1058  
**CF\_Chunks\_CombinePrevious**  
 cf\_chunk.c, 1048  
 cf\_chunk.h, 1059  
**CF\_Chunks\_EraseChunk**  
 cf\_chunk.c, 1048  
 cf\_chunk.h, 1059  
**CF\_Chunks\_EraseRange**  
 cf\_chunk.c, 1049  
 cf\_chunk.h, 1060  
**CF\_Chunks\_FindInsertPosition**  
 cf\_chunk.c, 1049  
 cf\_chunk.h, 1060  
**CF\_Chunks\_FindSmallestSize**  
 cf\_chunk.c, 1050  
 cf\_chunk.h, 1061  
**CF\_Chunks\_Insert**  
 cf\_chunk.c, 1050  
 cf\_chunk.h, 1061  
**CF\_Chunks\_InsertChunk**  
 cf\_chunk.c, 1051  
 cf\_chunk.h, 1062  
**CF\_ChunkSize\_t**  
 cf\_chunk.h, 1054  
  
**CF\_ChunkWrapper**, 547  
 chunks, 548  
 cl\_node, 548  
**CF\_ChunkWrapper\_t**  
 cf\_cfdp\_types.h, 1039  
**cf\_clist.c**  
 CF\_CList\_InitNode, 1063  
 CF\_CList\_InsertAfter, 1064  
 CF\_CList\_InsertBack, 1064  
 CF\_CList\_InsertFront, 1064  
 CF\_CList\_Pop, 1065  
 CF\_CList\_Remove, 1065  
 CF\_CList\_Traverse, 1066  
 CF\_CList\_Traverse\_R, 1067  
**cf\_clist.h**  
 CF\_CLIST\_CONT, 1069  
 CF\_CLIST\_EXIT, 1069  
 CF\_CList\_InitNode, 1070  
 CF\_CList\_InsertAfter, 1070  
 CF\_CList\_InsertBack, 1071  
 CF\_CList\_InsertFront, 1071  
 CF\_CList\_Pop, 1071  
 CF\_CList\_Remove, 1072  
 CF\_CList\_Traverse, 1073  
 CF\_CList\_Traverse\_R, 1073  
 CF\_CListFn\_t, 1069  
 CF\_CListNode\_t, 1069  
 CF\_CListTraverse\_Status\_CONTINUE, 1070  
 CF\_CListTraverse\_Status\_EXIT, 1070  
 CF\_CListTraverse\_Status\_IS\_CONTINUE, 1074  
 CF\_CListTraverse\_Status\_t, 1070  
 container\_of, 1069  
**CF\_CLIST\_CONT**  
 cf\_clist.h, 1069  
**CF\_CLIST\_EXIT**  
 cf\_clist.h, 1069  
**CF\_CList\_InitNode**  
 cf\_clist.c, 1063  
 cf\_clist.h, 1070  
**CF\_CList\_InsertAfter**  
 cf\_clist.c, 1064  
 cf\_clist.h, 1070  
**CF\_CList\_InsertAfter\_Ex**  
 cf\_utils.h, 1241  
**CF\_CList\_InsertBack**  
 cf\_clist.c, 1064  
 cf\_clist.h, 1071  
**CF\_CList\_InsertBack\_Ex**  
 cf\_utils.h, 1241  
**CF\_CList\_InsertFront**  
 cf\_clist.c, 1064  
 cf\_clist.h, 1071  
**CF\_CList\_Pop**  
 cf\_clist.c, 1065

cf\_clist.h, 1071  
CF\_CList\_Remove  
    cf\_clist.c, 1065  
    cf\_clist.h, 1072  
CF\_CList\_Remove\_Ex  
    cf\_utils.h, 1241  
CF\_CList\_Traverse  
    cf\_clist.c, 1066  
    cf\_clist.h, 1073  
CF\_CList\_Traverse\_R  
    cf\_clist.c, 1067  
    cf\_clist.h, 1073  
CF\_CListFn\_t  
    cf\_clist.h, 1069  
CF\_CListNode, 548  
    next, 548  
    prev, 548  
CF\_CListNode\_t  
    cf\_clist.h, 1069  
CF\_CListTraverse\_Status\_CONTINUE  
    cf\_clist.h, 1070  
CF\_CListTraverse\_Status\_EXIT  
    cf\_clist.h, 1070  
CF\_CListTraverse\_Status\_IS\_CONTINUE  
    cf\_clist.h, 1074  
CF\_CListTraverse\_Status\_t  
    cf\_clist.h, 1070  
cf\_cmd.c  
    CF\_Abandon\_TxnCmd, 1076  
    CF\_AbandonCmd, 1077  
    CF\_Cancel\_TxnCmd, 1078  
    CF\_CancelCmd, 1078  
    CF\_DisableDequeueCmd, 1079  
    CF\_DisableDirPollingCmd, 1080  
    CF\_DisableEngineCmd, 1080  
    CF\_DoChanAction, 1081  
    CF\_DoEnableDisableDequeue, 1082  
    CF\_DoEnableDisablePolldir, 1082  
    CF\_DoFreezeThaw, 1083  
    CF\_DoPurgeQueue, 1084  
    CF\_DoSuspRes, 1085  
    CF\_DoSuspRes\_Txn, 1086  
    CF\_EnableDequeueCmd, 1086  
    CF\_EnableDirPollingCmd, 1087  
    CF\_EnableEngineCmd, 1087  
    CF\_FindTransactionBySequenceNumberAllChannels,  
        1088  
    CF\_FreezeCmd, 1089  
    CF\_GetParamCmd, 1090  
    CF\_GetSetParamCmd, 1090  
    CF\_NoopCmd, 1092  
    CF\_PlaybackDirCmd, 1093  
    CF\_PurgeHistory, 1093  
    CF\_PurgeQueueCmd, 1094  
CF\_PurgeTransaction, 1095  
CF\_ResetCountersCmd, 1096  
CF\_ResumeCmd, 1097  
CF\_SendHkCmd, 1097  
CF\_SetParamCmd, 1098  
CF\_SuspendCmd, 1099  
CF\_ThawCmd, 1099  
CF\_TsnChanAction, 1100  
CF\_TxFileCmd, 1101  
CF\_ValidateChunkSizeCmd, 1102  
CF\_ValidateMaxOutgoingCmd, 1102  
CF\_WakeupCmd, 1103  
CF\_WriteQueueCmd, 1104  
cf\_cmd.h  
    CF\_Abandon\_TxnCmd, 1109  
    CF\_AbandonCmd, 1109  
    CF\_Cancel\_TxnCmd, 1110  
    CF\_CancelCmd, 1111  
    CF\_ChanAction\_BoolArg\_t, 1108  
    CF\_ChanAction\_BoolMsgArg\_t, 1108  
    CF\_ChanAction\_MsgArg\_t, 1108  
    CF\_ChanAction\_Status\_ERROR, 1108  
    CF\_ChanAction\_Status\_IS\_SUCCESS, 1111  
    CF\_ChanAction\_Status\_SUCCESS, 1108  
    CF\_ChanAction\_Status\_t, 1108  
    CF\_ChanAction\_SuspResArg\_t, 1108  
    CF\_ChanActionFn\_t, 1108  
    CF\_DisableDequeueCmd, 1111  
    CF\_DisableDirPollingCmd, 1112  
    CF\_DisableEngineCmd, 1113  
    CF\_DoChanAction, 1113  
    CF\_DoEnableDisableDequeue, 1114  
    CF\_DoEnableDisablePolldir, 1115  
    CF\_DoFreezeThaw, 1116  
    CF\_DoPurgeQueue, 1116  
    CF\_DoSuspRes, 1117  
    CF\_DoSuspRes\_Txn, 1118  
    CF\_EnableDequeueCmd, 1118  
    CF\_EnableDirPollingCmd, 1119  
    CF\_EnableEngineCmd, 1120  
    CF\_FindTransactionBySequenceNumberAllChannels,  
        1121  
    CF\_FreezeCmd, 1122  
    CF\_GetParamCmd, 1122  
    CF\_GetSetParamCmd, 1123  
    CF\_NoopCmd, 1124  
    CF\_PlaybackDirCmd, 1125  
    CF\_PurgeHistory, 1126  
    CF\_PurgeQueueCmd, 1126  
    CF\_PurgeTransaction, 1127  
    CF\_ResetCountersCmd, 1128  
    CF\_ResumeCmd, 1129  
    CF\_SendHkCmd, 1129  
    CF\_SetParamCmd, 1130

CF\_SuspendCmd, [1131](#)  
 CF\_ThawCmd, [1131](#)  
 CF\_TsnChanAction, [1132](#)  
 CF\_TsnChanAction\_fn\_t, [1108](#)  
 CF\_TxFileCmd, [1133](#)  
 CF\_ValidateChunkSizeCmd, [1134](#)  
 CF\_ValidateMaxOutgoingCmd, [1134](#)  
 CF\_WakeupCmd, [1135](#)  
 CF\_WriteQueueCmd, [1136](#)  
**CF\_CMD\_ABANDON\_CHAN\_ERR\_EID**  
     CFS CFDP Event IDs, [171](#)  
**CF\_CMD\_ABANDON\_INF\_EID**  
     CFS CFDP Event IDs, [171](#)  
**CF\_CMD\_BAD\_PARAM\_ERR\_EID**  
     CFS CFDP Event IDs, [171](#)  
**CF\_CMD\_CANCEL\_CHAN\_ERR\_EID**  
     CFS CFDP Event IDs, [171](#)  
**CF\_CMD\_CANCEL\_INF\_EID**  
     CFS CFDP Event IDs, [172](#)  
**CF\_CMD\_CHAN\_PARAM\_ERR\_EID**  
     CFS CFDP Event IDs, [172](#)  
**CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID**  
     CFS CFDP Event IDs, [172](#)  
**CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID**  
     CFS CFDP Event IDs, [172](#)  
**CF\_CMD\_DISABLE\_ENGINE\_INF\_EID**  
     CFS CFDP Event IDs, [173](#)  
**CF\_CMD\_DISABLE\_POLLDIR\_ERR\_EID**  
     CFS CFDP Event IDs, [173](#)  
**CF\_CMD\_DISABLE\_POLLDIR\_INF\_EID**  
     CFS CFDP Event IDs, [173](#)  
**CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID**  
     CFS CFDP Event IDs, [173](#)  
**CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID**  
     CFS CFDP Event IDs, [174](#)  
**CF\_CMD\_ENABLE\_ENGINE\_ERR\_EID**  
     CFS CFDP Event IDs, [174](#)  
**CF\_CMD\_ENABLE\_ENGINE\_INF\_EID**  
     CFS CFDP Event IDs, [174](#)  
**CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID**  
     CFS CFDP Event IDs, [174](#)  
**CF\_CMD\_ENABLE\_POLLDIR\_INF\_EID**  
     CFS CFDP Event IDs, [175](#)  
**CF\_CMD\_ENG\_ALREADY\_DIS\_INF\_EID**  
     CFS CFDP Event IDs, [175](#)  
**CF\_CMD\_ENG\_ALREADY\_ENA\_INF\_EID**  
     CFS CFDP Event IDs, [175](#)  
**CF\_CMD\_FREEZE\_ERR\_EID**  
     CFS CFDP Event IDs, [175](#)  
**CF\_CMD\_FREEZE\_INF\_EID**  
     CFS CFDP Event IDs, [176](#)  
**CF\_CMD\_GETSET1\_INF\_EID**  
     CFS CFDP Event IDs, [176](#)  
**CF\_CMD\_GETSET2\_INF\_EID**  
     CFS CFDP Event IDs, [176](#)  
**CFS CFDP Event IDs, [176](#)**  
**CF\_CMD\_GETSET\_CHAN\_ERR\_EID**  
     CFS CFDP Event IDs, [176](#)  
**CF\_CMD\_GETSET\_PARAM\_ERR\_EID**  
     CFS CFDP Event IDs, [177](#)  
**CF\_CMD\_GETSET\_VALIDATE\_ERR\_EID**  
     CFS CFDP Event IDs, [177](#)  
**CF\_CMD\_LEN\_ERR\_EID**  
     CFS CFDP Event IDs, [177](#)  
**CF\_CMD\_MID**  
     CFS CFDP Command Message IDs, [228](#)  
**CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID**  
     CFS CFDP Event IDs, [177](#)  
**CF\_CMD\_PLAYBACK\_DIR\_INF\_EID**  
     CFS CFDP Event IDs, [178](#)  
**CF\_CMD\_POLLDIR\_INVALID\_ERR\_EID**  
     CFS CFDP Event IDs, [178](#)  
**CF\_CMD\_PURGE\_ARG\_ERR\_EID**  
     CFS CFDP Event IDs, [178](#)  
**CF\_CMD\_PURGE\_QUEUE\_ERR\_EID**  
     CFS CFDP Event IDs, [178](#)  
**CF\_CMD\_PURGE\_QUEUE\_INF\_EID**  
     CFS CFDP Event IDs, [179](#)  
**CF\_CMD\_RESET\_INVALID\_ERR\_EID**  
     CFS CFDP Event IDs, [179](#)  
**CF\_CMD\_SUSPRES\_CHAN\_ERR\_EID**  
     CFS CFDP Event IDs, [179](#)  
**CF\_CMD\_SUSPRES\_INF\_EID**  
     CFS CFDP Event IDs, [179](#)  
**CF\_CMD\_SUSPRES\_SAME\_INF\_EID**  
     CFS CFDP Event IDs, [180](#)  
**CF\_CMD\_THAW\_ERR\_EID**  
     CFS CFDP Event IDs, [180](#)  
**CF\_CMD\_THAW\_INF\_EID**  
     CFS CFDP Event IDs, [180](#)  
**CF\_CMD\_TRANS\_NOT\_FOUND\_ERR\_EID**  
     CFS CFDP Event IDs, [180](#)  
**CF\_CMD\_TSN\_CHAN\_INVALID\_ERR\_EID**  
     CFS CFDP Event IDs, [181](#)  
**CF\_CMD\_TX\_FILE\_ERR\_EID**  
     CFS CFDP Event IDs, [181](#)  
**CF\_CMD\_TX\_FILE\_INF\_EID**  
     CFS CFDP Event IDs, [181](#)  
**CF\_CMD\_WHIST\_WRITE\_ERR\_EID**  
     CFS CFDP Event IDs, [181](#)  
**CF\_CMD\_WQ\_ARGS\_ERR\_EID**  
     CFS CFDP Event IDs, [182](#)  
**CF\_CMD\_WQ\_CHAN\_ERR\_EID**  
     CFS CFDP Event IDs, [182](#)  
**CF\_CMD\_WQ\_INF\_EID**  
     CFS CFDP Event IDs, [182](#)  
**CF\_CMD\_WQ\_OPEN\_ERR\_EID**  
     CFS CFDP Event IDs, [182](#)  
**CF\_CMD\_WQ\_WRITEHIST\_RX\_ERR\_EID**

CFS CFDP Event IDs, [183](#)  
CF\_CMD\_WQ\_WRITEHIST\_TX\_ERR\_EID  
    CFS CFDP Event IDs, [183](#)  
CF\_CMD\_WQ\_WRITEQ\_PEND\_ERR\_EID  
    CFS CFDP Event IDs, [183](#)  
CF\_CMD\_WQ\_WRITEQ\_RX\_ERR\_EID  
    CFS CFDP Event IDs, [183](#)  
CF\_CMD\_WQ\_WRITEQ\_TX\_ERR\_EID  
    CFS CFDP Event IDs, [184](#)  
cf\_codec.c  
    CF\_CFDP\_CodecCheckSize, [1140](#)  
    CF\_CFDP\_DecodeAck, [1141](#)  
    CF\_CFDP\_DecodeAllSegments, [1142](#)  
    CF\_CFDP\_DecodeAllTlv, [1142](#)  
    CF\_CFDP\_DecodeCrc, [1143](#)  
    CF\_CFDP\_DecodeEof, [1143](#)  
    CF\_CFDP\_DecodeFileDataHeader, [1144](#)  
    CF\_CFDP\_DecodeFileDirectiveHeader, [1145](#)  
    CF\_CFDP\_DecodeFin, [1145](#)  
    CF\_CFDP\_DecodeHeader, [1146](#)  
    CF\_CFDP\_DecodeLv, [1147](#)  
    CF\_CFDP\_DecodeMd, [1147](#)  
    CF\_CFDP\_DecodeNak, [1149](#)  
    CF\_CFDP\_DecodeSegmentRequest, [1149](#)  
    CF\_CFDP\_DecodeTlv, [1150](#)  
    CF\_CFDP\_DoDecodeChunk, [1151](#)  
    CF\_CFDP\_DoEncodeChunk, [1151](#)  
    CF\_CFDP\_EncodeAck, [1152](#)  
    CF\_CFDP\_EncodeAllSegments, [1153](#)  
    CF\_CFDP\_EncodeAllTlv, [1153](#)  
    CF\_CFDP\_EncodeCrc, [1154](#)  
    CF\_CFDP\_EncodeEof, [1154](#)  
    CF\_CFDP\_EncodeFileDataHeader, [1155](#)  
    CF\_CFDP\_EncodeFileDirectiveHeader, [1156](#)  
    CF\_CFDP\_EncodeFin, [1156](#)  
    CF\_CFDP\_EncodeHeaderFinalSize, [1157](#)  
    CF\_CFDP\_EncodeHeaderWithoutSize, [1158](#)  
    CF\_CFDP\_EncodeLv, [1159](#)  
    CF\_CFDP\_EncodeMd, [1159](#)  
    CF\_CFDP\_EncodeNak, [1160](#)  
    CF\_CFDP\_EncodeSegmentRequest, [1161](#)  
    CF\_CFDP\_EncodeTlv, [1161](#)  
    CF\_CFDP\_GetValueEncodedSize, [1162](#)  
    CF\_CFDP\_PduAck\_CC, [1165](#)  
    CF\_CFDP\_PduAck\_DIR\_CODE, [1165](#)  
    CF\_CFDP\_PduAck\_DIR\_SUBTYPE\_CODE, [1165](#)  
    CF\_CFDP\_PduAck\_TRANSACTION\_STATUS, [1166](#)  
    CF\_CFDP\_PduEof\_FLAGS\_CC, [1166](#)  
    CF\_CFDP\_PduFileData\_RECORD\_CONTINUATION\_STATE  
        [1166](#)  
    CF\_CFDP\_PduFileData\_SEGMENT\_METADATA\_LENGTH, [1166](#)  
    CF\_CFDP\_PduFin\_FLAGS\_CC, [1166](#)  
CF\_CFDP\_PduFin\_FLAGS\_DELIVERY\_CODE,  
    [1166](#)  
CF\_CFDP\_PduFin\_FLAGS\_FILE\_STATUS, [1166](#)  
CF\_CFDP\_PduHeader\_FLAGS\_CRC, [1166](#)  
CF\_CFDP\_PduHeader\_FLAGS\_DIR, [1167](#)  
CF\_CFDP\_PduHeader\_FLAGS\_LARGEFILE, [1167](#)  
CF\_CFDP\_PduHeader\_FLAGS\_MODE, [1167](#)  
CF\_CFDP\_PduHeader\_FLAGS\_TYPE, [1167](#)  
CF\_CFDP\_PduHeader\_FLAGS\_VERSION, [1167](#)  
CF\_CFDP\_PduHeader\_LENGTHS\_ENTITY, [1167](#)  
CF\_CFDP\_PduHeader\_LENGTHS\_TRANSACTION\_SEQUENCE,  
    [1167](#)  
CF\_CFDP\_PduHeader\_SEGMENT\_METADATA,  
    [1167](#)  
CF\_CFDP\_PduHeader\_SEGMENTATION\_CONTROL,  
    [1167](#)  
CF\_CFDP\_PduMd\_CHECKSUM\_TYPE, [1168](#)  
CF\_CFDP\_PduMd\_CLOSURE\_REQUESTED, [1168](#)  
CF\_Codec\_BitField\_t, [1140](#)  
CF\_Codec\_Load\_uint16, [1162](#)  
CF\_Codec\_Load\_uint32, [1162](#)  
CF\_Codec\_Load\_uint64, [1162](#)  
CF\_Codec\_Load\_uint8, [1163](#)  
CF\_Codec\_Store\_uint16, [1163](#)  
CF\_Codec\_Store\_uint32, [1163](#)  
CF\_Codec\_Store\_uint64, [1163](#)  
CF\_Codec\_Store\_uint8, [1163](#)  
CF\_DecodeIntegerInSize, [1163](#)  
CF\_EncodeIntegerInSize, [1164](#)  
CF\_FieldGetVal, [1165](#)  
CF\_FieldSetVal, [1165](#)  
CF\_INIT\_FIELD, [1140](#)  
FGV, [1140](#)  
FSV, [1140](#)  
cf\_codec.h  
    CF\_CFDP\_CodecCheckSize, [1173](#)  
    CF\_CFDP\_CodecGetPosition, [1174](#)  
    CF\_CFDP\_CodecGetRemain, [1174](#)  
    CF\_CFDP\_CodecGetSize, [1175](#)  
    CF\_CFDP\_CodeclsOK, [1175](#)  
    CF\_CFDP\_CodecReset, [1175](#)  
    CF\_CFDP\_CodecSetDone, [1176](#)  
    CF\_CFDP\_DecodeAck, [1176](#)  
    CF\_CFDP\_DecodeAllSegments, [1176](#)  
    CF\_CFDP\_DecodeAllTlv, [1177](#)  
    CF\_CFDP\_DecodeCrc, [1177](#)  
    CF\_CFDP\_DecodeEof, [1178](#)  
    CF\_CFDP\_DecodeFileDataHeader, [1178](#)  
    CF\_CFDP\_DecodeFin, [1180](#)  
    CF\_CFDP\_DecodeHeader, [1180](#)  
    CF\_CFDP\_DecodeLv, [1181](#)  
    CF\_CFDP\_DecodeMd, [1182](#)  
    CF\_CFDP\_DecodeNak, [1182](#)

CF\_CFDP\_DecodeSegmentRequest, 1183  
 CF\_CFDP\_DecodeTLV, 1184  
 CF\_CFDP\_DoDecodeChunk, 1184  
 CF\_CFDP\_DoEncodeChunk, 1185  
 CF\_CFDP\_EncodeAck, 1186  
 CF\_CFDP\_EncodeAllSegments, 1186  
 CF\_CFDP\_EncodeAllTlv, 1187  
 CF\_CFDP\_EncodeCrc, 1187  
 CF\_CFDP\_EncodeEof, 1188  
 CF\_CFDP\_EncodeFileDataHeader, 1188  
 CF\_CFDP\_EncodeFileDirectiveHeader, 1189  
 CF\_CFDP\_EncodeFin, 1190  
 CF\_CFDP\_EncodeHeaderFinalSize, 1190  
 CF\_CFDP\_EncodeHeaderWithoutSize, 1191  
 CF\_CFDP\_EncodeLV, 1192  
 CF\_CFDP\_EncodeMd, 1192  
 CF\_CFDP\_EncodeNak, 1193  
 CF\_CFDP\_EncodeSegmentRequest, 1194  
 CF\_CFDP\_EncodeTLV, 1194  
 CF\_CFDP\_GetValueEncodedSize, 1196  
 CF\_CODEC\_GET\_POSITION, 1171  
 CF\_CODEC\_GET\_REMAIN, 1171  
 CF\_CODEC\_GET\_SIZE, 1171  
 CF\_CODEC\_IS\_OK, 1171  
 CF\_CODEC\_SET\_DONE, 1172  
 CF\_CodecState\_t, 1173  
 CF\_DECODE\_FIXED\_CHUNK, 1172  
 CF\_DecodeIntegerInSize, 1196  
 CF\_DecoderState\_t, 1173  
 CF\_ENCODE\_FIXED\_CHUNK, 1172  
 CF\_EncodeIntegerInSize, 1197  
 CF\_EncoderState\_t, 1173  
 CF\_Codec\_BitField, 549  
     mask, 549  
     shift, 549  
 CF\_Codec\_BitField\_t  
     cf\_codec.c, 1140  
 CF\_CODEC\_GET\_POSITION  
     cf\_codec.h, 1171  
 CF\_CODEC\_GET\_REMAIN  
     cf\_codec.h, 1171  
 CF\_CODEC\_GET\_SIZE  
     cf\_codec.h, 1171  
 CF\_CODEC\_IS\_OK  
     cf\_codec.h, 1171  
 CF\_Codec\_Load\_uint16  
     cf\_codec.c, 1162  
 CF\_Codec\_Load\_uint32  
     cf\_codec.c, 1162  
 CF\_Codec\_Load\_uint64  
     cf\_codec.c, 1162  
 CF\_Codec\_Load\_uint8  
     cf\_codec.c, 1163  
 CF\_CODEC\_SET\_DONE  
     cf\_codec.h, 1172  
     cf\_codec.c, 1163  
 CF\_Codec\_Store\_uint16  
     cf\_codec.c, 1163  
 CF\_Codec\_Store\_uint32  
     cf\_codec.c, 1163  
 CF\_Codec\_Store\_uint64  
     cf\_codec.c, 1163  
 CF\_Codec\_Store\_uint8  
     cf\_codec.c, 1163  
 CF\_CodecState, 549  
     is\_valid, 550  
     max\_size, 550  
     next\_offset, 550  
 CF\_CodecState\_t  
     cf\_codec.h, 1173  
 CF\_COMPOUND\_KEY  
     default\_cf\_msg.h, 804  
 CF\_config\_table  
     cf\_def\_config.c, 1261  
 CF\_CONFIG\_TABLE\_FILENAME  
     CFS CFDP Platform Configuration, 211  
 CF\_CONFIG\_TABLE\_NAME  
     CFS CFDP Platform Configuration, 211  
 CF\_ConfigTable, 550  
     chan, 551  
     fail\_dir, 551  
     local\_eid, 551  
     outgoing\_file\_chunk\_size, 551  
     rx\_crc\_calc\_bytes\_per\_wakeup, 551  
     ticks\_per\_second, 551  
     tmp\_dir, 551  
 CF\_ConfigTable\_t  
     default\_cf\_tblstruct.h, 813  
 CF\_CR\_CHANNEL\_PIPE\_ERR\_EID  
     CFS CFDP Event IDs, 184  
 CF\_CR\_PIPE\_ERR\_EID  
     CFS CFDP Event IDs, 184  
 CF\_Crc, 552  
     index, 552  
     result, 552  
     working, 552  
 cf\_crc.c  
     CF\_CRC\_Digest, 1198  
     CF\_CRC\_Finalize, 1199  
     CF\_CRC\_Start, 1199  
 cf\_crc.h  
     CF\_CRC\_Digest, 1200  
     CF\_CRC\_Finalize, 1201  
     CF\_CRC\_Start, 1201  
     CF\_Crc\_t, 1200  
 CF\_CRC\_Digest  
     cf\_crc.c, 1198  
     cf\_crc.h, 1200  
 CF\_CRC\_Finalize

cf_crc.c, 1199	CFS CFDP Command Structures, 225
cf_crc.h, 1201	
CF_CRC_Start	
cf_crc.c, 1199	
cf_crc.h, 1201	
CF_Crc_t	
cf_crc.h, 1200	
CF_DECODE_FIXED_CHUNK	
cf_codec.h, 1172	
CF_DecodeIntegerInSize	
cf_codec.c, 1163	
cf_codec.h, 1196	
CF_DecoderState, 552	
base, 553	
codec_state, 553	
CF_DecoderState_t	
cf_codec.h, 1173	
cf_def_config.c	
CF_config_table, 1261	
CF_DequeueTransaction	
cf_utils.h, 1242	
CF_Direction_NUM	
cf_cfdp_types.h, 1040	
CF_Direction_RX	
cf_cfdp_types.h, 1040	
CF_Direction_t	
cf_cfdp_types.h, 1040	
CF_Direction_TX	
cf_cfdp_types.h, 1040	
CF_DISABLE_DEQUEUE_CC	
CFS CFDP Command Codes, 195	
CF_DISABLE_DIR_POLLING_CC	
CFS CFDP Command Codes, 196	
CF_DISABLE_ENGINE_CC	
CFS CFDP Command Codes, 197	
CF_DisableDequeueCmd, 553	
cf_cmd.c, 1079	
cf_cmd.h, 1111	
CommandHeader, 553	
Payload, 553	
CF_DisableDequeueCmd_t	
CFS CFDP Command Structures, 224	
CF_DisableDirPollingCmd, 554	
cf_cmd.c, 1080	
cf_cmd.h, 1112	
CommandHeader, 554	
Payload, 554	
CF_DisableDirPollingCmd_t	
CFS CFDP Command Structures, 225	
CF_DisableEngineCmd, 554	
cf_cmd.c, 1080	
cf_cmd.h, 1113	
CommandHeader, 554	
CF_DisableEngineCmd_t	

cf\_cmd.c, 1087  
 cf\_cmd.h, 1120  
 CommandHeader, 556  
**CF\_EnableEngineCmd\_t**  
 CFS CFDP Command Structures, 225  
**CF\_ENCODE\_FIXED\_CHUNK**  
 cf\_codec.h, 1172  
**CF\_EncodeIntegerInSize**  
 cf\_codec.c, 1164  
 cf\_codec.h, 1197  
**CF\_EncoderState**, 556  
 base, 557  
 codec\_state, 557  
**CF\_EncoderState\_t**  
 cf\_codec.h, 1173  
**CF\_Engine**, 557  
 channels, 558  
 chunk\_mem, 558  
 chunks, 558  
 enabled, 558  
 histories, 558  
 in, 558  
 out, 558  
 seq\_num, 558  
 transactions, 558  
**CF\_Engine\_t**  
 cf\_cfdp\_types.h, 1039  
**CF\_EntityId\_t**  
 default\_cf\_extern\_typedefs.h, 799  
 eds\_cf\_extern\_typedefs.h, 814  
**CF\_EOT\_TLM\_MID**  
 CFS CFDP Telemetry Message IDs, 229  
**CF\_EotPacket**, 559  
 Payload, 559  
 TelemetryHeader, 559  
**CF\_EotPacket\_Payload**, 559  
 channel, 560  
 crc\_result, 560  
 direction, 560  
 fnames, 560  
 fsize, 560  
 peer\_eid, 560  
 seq\_num, 561  
 src\_eid, 561  
 state, 561  
 txn\_stat, 561  
**CF\_EotPacket\_Payload\_t**  
 CFS CFDP Telemetry, 221  
**CF\_EotPacket\_t**  
 CFS CFDP Telemetry, 221  
**CF\_ERROR**  
 cf\_app.h, 837  
**CF\_FieldGetVal**  
 cf\_codec.c, 1165  
**CF\_FieldSetVal**  
 cf\_codec.c, 1165  
**CF\_FILENAME\_MAX\_LEN**  
 CFS CFDP Platform Configuration, 211  
**CF\_FILENAME\_MAX\_NAME**  
 CFS CFDP Platform Configuration, 211  
**CF\_FILENAME\_MAX\_PATH**  
 default\_cf\_mission\_cfg.h, 803  
**CF\_FileName\_t**  
 eds\_cf\_extern\_typedefs.h, 814  
**CF\_FILENAME\_TRUNCATED**  
 cf\_app.h, 837  
**CF\_FileSize\_t**  
 cf\_logical\_pdu.h, 1211  
**CF\_FindTransactionBySequenceNumber**  
 cf\_utils.c, 1220  
 cf\_utils.h, 1242  
**CF\_FindTransactionBySequenceNumber\_Impl**  
 cf\_utils.c, 1221  
 cf\_utils.h, 1243  
**CF\_FindTransactionBySequenceNumberAllChannels**  
 cf\_cmd.c, 1088  
 cf\_cmd.h, 1121  
**CF\_FindUnusedTransaction**  
 cf\_utils.c, 1221  
 cf\_utils.h, 1244  
**CF\_Flags\_Common**, 561  
 ack\_timer\_armed, 562  
 canceled, 562  
 close\_req, 562  
 crc\_complete, 562  
 inactivity\_fired, 562  
 is\_complete, 562  
 keep\_history, 562  
 q\_index, 562  
 suspended, 563  
**CF\_Flags\_Common\_t**  
 cf\_cfdp\_types.h, 1039  
**CF\_Flags\_Rx**, 563  
 com, 563  
 eof\_ack\_count, 563  
 eof\_count, 563  
 finack\_recv, 564  
 md\_recv, 564  
 send\_fin, 564  
 send\_nak, 564  
 tempfile\_created, 564  
**CF\_Flags\_Rx\_t**  
 cf\_cfdp\_types.h, 1039  
**CF\_Flags\_Tx**, 564  
 cmd\_tx, 565  
 com, 565  
 eof\_ack\_recv, 565  
 fd\_nak\_pending, 565

fin\_ack\_count, 565  
fin\_count, 565  
send\_eof, 565  
send\_md, 566  
**CF\_Flags\_Tx\_t**  
    cf\_cfdp\_types.h, 1039  
**CF\_FreeTransaction**  
    cf\_utils.c, 1222  
    cf\_utils.h, 1244  
**CF\_FREEZE\_CC**  
    CFS CFDP Command Codes, 200  
**CF\_FreezeCmd**, 566  
    cf\_cmd.c, 1089  
    cf\_cmd.h, 1122  
    CommandHeader, 566  
    Payload, 566  
**CF\_FreezeCmd\_t**  
    CFS CFDP Command Structures, 225  
**CF\_FunctionCode\_**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_ABANDON**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_CANCEL**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_DISABLE\_DEQUEUE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_DISABLE\_DIR\_POLLING**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_DISABLE\_ENGINE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_ENABLE\_DEQUEUE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_ENABLE\_DIR\_POLLING**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_ENABLE\_ENGINE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_FREEZE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_GET\_PARAM**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_NOOP**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_PLAYBACK\_DIR**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_PURGE\_QUEUE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_RESET\_COUNTERS**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_RESUME**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_SET\_PARAM**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_SUSPEND**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_THAW**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_TX\_FILE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_FunctionCode\_WRITE\_QUEUE**  
    default\_cf\_fcncode\_values.h, 802  
**CF\_GapComputeArgs\_t**, 566  
    nak, 567  
    txn, 567  
**CF\_GET\_PARAM\_CC**  
    CFS CFDP Command Codes, 201  
**CF\_GetChannelFromTxn**  
    cf\_utils.c, 1223  
    cf\_utils.h, 1245  
**CF\_GetChunkListHead**  
    cf\_utils.c, 1223  
    cf\_utils.h, 1245  
**CF\_GetParam\_Payload**, 567  
    chan\_num, 567  
    key, 568  
**CF\_GetParam\_Payload\_t**  
    CFS CFDP Command Structures, 225  
**CF\_GetParamCmd**, 568  
    cf\_cmd.c, 1090  
    cf\_cmd.h, 1122  
    CommandHeader, 568  
    Payload, 568  
**CF\_GetParamCmd\_t**  
    CFS CFDP Command Structures, 225  
**CF\_Set\_ValueID\_ack\_limit**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_ack\_timer\_s**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_chan\_max\_outgoing\_messages\_per\_wakeup**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_inactivity\_timer\_s**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_local\_eid**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_MAX**  
    CFS CFDP Command Structures, 227  
    eds\_cf\_extern\_typedefs.h, 814  
**CF\_Set\_ValueID\_nak\_limit**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_nak\_timer\_s**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_outgoing\_file\_chunk\_size**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_rx\_crc\_calc\_bytes\_per\_wakeup**  
    CFS CFDP Command Structures, 227  
**CF\_Set\_ValueID\_t**  
    CFS CFDP Command Structures, 227  
    eds\_cf\_extern\_typedefs.h, 814  
**CF\_Set\_ValueID\_ticks\_per\_second**

CFS CFDP Command Structures, 227  
**CF\_GetSetParamCmd**  
 cf\_cmd.c, 1090  
 cf\_cmd.h, 1123  
**CF\_History**, 568  
 cl\_node, 569  
 dir, 569  
 fnames, 569  
 peer\_eid, 569  
 seq\_num, 570  
 src\_eid, 570  
 txn\_stat, 570  
**CF\_History\_t**  
 cf\_cfdp\_types.h, 1039  
**CF\_HK\_TLM\_MID**  
 CFS CFDP Telemetry Message IDs, 229  
**CF\_HkChannel\_Data**, 570  
 counters, 571  
 frozen, 571  
 playback\_counter, 571  
 poll\_counter, 571  
 q\_size, 571  
 spare, 572  
**CF\_HkChannel\_Data\_t**  
 CFS CFDP Telemetry, 221  
**CF\_HkCmdCounters**, 572  
 cmd, 572  
 err, 572  
**CF\_HkCmdCounters\_t**  
 CFS CFDP Telemetry, 221  
**CF\_HkCounters**, 573  
 fault, 573  
 recv, 573  
 sent, 573  
**CF\_HkCounters\_t**  
 CFS CFDP Telemetry, 221  
**CF\_HkFault**, 573  
 ack\_limit, 574  
 crc\_mismatch, 574  
 directory\_read, 574  
 file\_open, 575  
 file\_read, 575  
 file\_rename, 575  
 file\_seek, 575  
 file\_size\_mismatch, 575  
 file\_write, 575  
 inactivity\_timer, 575  
 nak\_limit, 575  
 spare, 575  
**CF\_HkFault\_t**  
 CFS CFDP Telemetry, 221  
**CF\_HkPacket**, 576  
 Payload, 576  
 TelemetryHeader, 576  
**CF\_HkPacket\_Payload**, 577  
 channel\_hk, 577  
 counters, 577  
 spare, 577  
**CF\_HkPacket\_Payload\_t**  
 CFS CFDP Telemetry, 221  
**CF\_HkPacket\_t**  
 CFS CFDP Telemetry, 221  
**CF\_HkRecv**, 578  
 dropped, 578  
 error, 578  
 file\_data\_bytes, 578  
 nak\_segment\_requests, 578  
 pdu, 579  
 spurious, 579  
**CF\_HkRecv\_t**  
 CFS CFDP Telemetry, 221  
**CF\_HkSent**, 579  
 file\_data\_bytes, 579  
 nak\_segment\_requests, 579  
 pdu, 580  
**CF\_HkSent\_t**  
 CFS CFDP Telemetry, 221  
**CF\_INIT\_CRC\_ALIGN\_ERR\_EID**  
 CFS CFDP Event IDs, 185  
**CF\_INIT\_FIELD**  
 cf\_codec.c, 1140  
**CF\_INIT\_INF\_EID**  
 CFS CFDP Event IDs, 185  
**CF\_INIT\_MSG\_RECV\_ERR\_EID**  
 CFS CFDP Event IDs, 185  
**CF\_INIT\_OUTGOING\_SIZE\_ERR\_EID**  
 CFS CFDP Event IDs, 185  
**CF\_INIT\_SEM\_ERR\_EID**  
 CFS CFDP Event IDs, 186  
**CF\_INIT\_SUB\_ERR\_EID**  
 CFS CFDP Event IDs, 186  
**CF\_INIT\_TBL\_CHECK\_GA\_ERR\_EID**  
 CFS CFDP Event IDs, 186  
**CF\_INIT\_TBL\_CHECK\_MAN\_ERR\_EID**  
 CFS CFDP Event IDs, 186  
**CF\_INIT\_TBL\_CHECK\_REL\_ERR\_EID**  
 CFS CFDP Event IDs, 187  
**CF\_INIT\_TBL\_GETADDR\_ERR\_EID**  
 CFS CFDP Event IDs, 187  
**CF\_INIT\_TBL\_LOAD\_ERR\_EID**  
 CFS CFDP Event IDs, 187  
**CF\_INIT\_TBL\_MANAGE\_ERR\_EID**  
 CFS CFDP Event IDs, 187  
**CF\_INIT\_TBL\_REG\_ERR\_EID**  
 CFS CFDP Event IDs, 188  
**CF\_INIT\_TPS\_ERR\_EID**  
 CFS CFDP Event IDs, 188  
**CF\_Input**, 580

decode, 580  
msg, 580  
rx\_pdudata, 580  
**CF\_Input\_t**  
  cf\_cfdp\_types.h, 1039  
**CF\_InsertSortPrio**  
  cf\_utils.c, 1224  
  cf\_utils.h, 1246  
**CF\_INTERFACE\_CFGVAL**  
  default\_cf\_interface\_cfg\_values.h, 802  
  eds\_cf\_interface\_cfg\_values.h, 815  
**CF\_INTERNAL\_CFGVAL**  
  default\_cf\_internal\_cfg\_values.h, 803  
**CF\_Logical\_IntHeader**, 581  
  ack, 581  
  eof, 581  
  fd, 581  
  fin, 582  
  md, 582  
  nak, 582  
**CF\_Logical\_IntHeader\_t**  
  cf\_logical\_pdu.h, 1211  
**CF\_Logical\_Lv**, 582  
  data\_ptr, 582  
  length, 583  
**CF\_Logical\_Lv\_t**  
  cf\_logical\_pdu.h, 1211  
**cf\_logical\_pdu.h**  
  **CF\_FileSize\_t**, 1211  
  **CF\_Logical\_IntHeader\_t**, 1211  
  **CF\_Logical\_Lv\_t**, 1211  
  **CF\_Logical\_PduAck\_t**, 1211  
  **CF\_Logical\_PduBuffer\_t**, 1211  
  **CF\_Logical\_PduEof\_t**, 1211  
  **CF\_Logical\_PduFileDataHeader\_t**, 1211  
  **CF\_Logical\_PduFileDirectiveHeader\_t**, 1211  
  **CF\_Logical\_PduFin\_t**, 1212  
  **CF\_Logical\_PduHeader\_t**, 1212  
  **CF\_Logical\_PduMd\_t**, 1212  
  **CF\_Logical\_PduNak\_t**, 1212  
  **CF\_Logical\_SegmentList\_t**, 1212  
  **CF\_Logical\_SegmentRequest\_t**, 1212  
  **CF\_Logical\_Tlv\_t**, 1212  
  **CF\_Logical\_TlvData\_t**, 1212  
  **CF\_Logical\_TlvList\_t**, 1212  
  **CF\_PDU\_MAX\_SEGMENTS**, 1210  
  **CF\_PDU\_MAX\_TLV**, 1211  
**CF\_Logical\_PduAck**, 583  
  ack\_directive\_code, 583  
  ack\_subtype\_code, 583  
  cc, 583  
  txn\_status, 584  
**CF\_Logical\_PduAck\_t**  
  cf\_logical\_pdu.h, 1211  
**CF\_Logical\_PduBuffer**, 584  
  content\_crc, 584  
  fdirective, 584  
  int\_header, 585  
  pdec, 585  
  pdu\_header, 585  
  penc, 585  
**CF\_Logical\_PduBuffer\_t**  
  cf\_logical\_pdu.h, 1211  
**CF\_Logical\_PduEof**, 585  
  cc, 586  
  crc, 586  
  size, 586  
  tlv\_list, 586  
**CF\_Logical\_PduEof\_t**  
  cf\_logical\_pdu.h, 1211  
**CF\_Logical\_PduFileDataHeader**, 586  
  continuation\_state, 587  
  data\_len, 587  
  data\_ptr, 587  
  offset, 587  
  segment\_list, 587  
**CF\_Logical\_PduFileDataHeader\_t**  
  cf\_logical\_pdu.h, 1211  
**CF\_Logical\_PduFileDirectiveHeader**, 587  
  directive\_code, 588  
**CF\_Logical\_PduFileDirectiveHeader\_t**  
  cf\_logical\_pdu.h, 1211  
**CF\_Logical\_PduFin**, 588  
  cc, 588  
  delivery\_code, 588  
  file\_status, 589  
  tlv\_list, 589  
**CF\_Logical\_PduFin\_t**  
  cf\_logical\_pdu.h, 1212  
**CF\_Logical\_PduHeader**, 589  
  crc\_flag, 590  
  data\_encoded\_length, 590  
  destination\_eid, 590  
  direction, 590  
  eid\_length, 590  
  header\_encoded\_length, 591  
  large\_flag, 591  
  pdu\_type, 591  
  segment\_meta\_flag, 591  
  segmentation\_control, 591  
  sequence\_num, 591  
  source\_eid, 591  
  txm\_mode, 591  
  txn\_seq\_length, 592  
  version, 592  
**CF\_Logical\_PduHeader\_t**  
  cf\_logical\_pdu.h, 1212  
**CF\_Logical\_PduMd**, 592

checksum\_type, 592  
 close\_req, 592  
 dest\_filename, 593  
 size, 593  
 source\_filename, 593  
**CF\_Logical\_PduMd\_t**  
*cf\_logical\_pdu.h*, 1212  
**CF\_Logical\_PduNak**, 593  
*scope\_end*, 593  
*scope\_start*, 593  
*segment\_list*, 593  
**CF\_Logical\_PduNak\_t**  
*cf\_logical\_pdu.h*, 1212  
**CF\_Logical\_SegmentList**, 594  
*num\_segments*, 594  
*segments*, 594  
**CF\_Logical\_SegmentList\_t**  
*cf\_logical\_pdu.h*, 1212  
**CF\_Logical\_SegmentRequest**, 594  
*offset\_end*, 595  
*offset\_start*, 595  
**CF\_Logical\_SegmentRequest\_t**  
*cf\_logical\_pdu.h*, 1212  
**CF\_Logical\_Tlv**, 595  
*data*, 595  
*length*, 595  
*type*, 596  
**CF\_Logical\_Tlv\_t**  
*cf\_logical\_pdu.h*, 1212  
**CF\_Logical\_TlvData**, 596  
*data\_ptr*, 596  
*eid*, 596  
**CF\_Logical\_TlvData\_t**  
*cf\_logical\_pdu.h*, 1212  
**CF\_Logical\_TlvList**, 597  
*num\_tlv*, 597  
*tlv*, 597  
**CF\_Logical\_TlvList\_t**  
*cf\_logical\_pdu.h*, 1212  
**CF\_MAJOR\_VERSION**  
 CFS CFDP Version, 219  
**CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PERCHAN**  
 CFS CFDP Platform Configuration, 212  
**CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN**  
 CFS CFDP Platform Configuration, 212  
**CF\_MAX\_PDU\_SIZE**  
 CFS CFDP Platform Configuration, 212  
**CF\_MAX\_POLLING\_DIR\_PER\_CHAN**  
 CFS CFDP Platform Configuration, 212  
**CF\_MAX\_SIMULTANEOUS\_RX**  
 CFS CFDP Platform Configuration, 213  
**CF\_MID\_ERR\_EID**  
 CFS CFDP Event IDs, 188  
**CF\_MINOR\_VERSION**  
 CFS CFDP Version, 219  
**CF\_MISSION\_REV**  
*default\_cf\_platform\_cfg.h*, 811  
**CF\_MoveTransaction**  
*cf\_utils.h*, 1247  
**CF\_NAK\_MAX\_SEGMENTS**  
 CFS CFDP Platform Configuration, 213  
**CF\_NOOP\_CC**  
 CFS CFDP Command Codes, 201  
**CF\_NOOP\_INF\_EID**  
 CFS CFDP Event IDs, 188  
**CF\_NoopCmd**, 597  
*cf\_cmd.c*, 1092  
*cf\_cmd.h*, 1124  
*CommandHeader*, 597  
**CF\_NoopCmd\_t**  
 CFS CFDP Command Structures, 225  
**CF\_NUM\_CHANNELS**  
 CFS CFDP Platform Configuration, 213  
**CF\_NUM\_CHUNKS\_ALL\_CHANNELS**  
*cf\_cfdp\_types.h*, 1038  
**CF\_NUM\_COMMANDS**  
 CFS CFDP Command Codes, 202  
**CF\_NUM\_HISTORIES**  
*cf\_cfdp\_types.h*, 1038  
**CF\_NUM\_HISTORIES\_PER\_CHANNEL**  
 CFS CFDP Platform Configuration, 213  
**CF\_NUM\_TRANSACTIONS**  
*cf\_cfdp\_types.h*, 1038  
**CF\_NUM\_TRANSACTIONS\_PER\_CHANNEL**  
*cf\_cfdp\_types.h*, 1038  
**CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK**  
 CFS CFDP Platform Configuration, 214  
**CF\_Output**, 598  
*encode*, 598  
*msg*, 598  
*tx\_pdudata*, 598  
**CF\_Output\_t**  
*cf\_cfdp\_types.h*, 1039  
**CF\_PathName\_t**  
*eds\_cf\_extern\_typedefs.h*, 814  
**CF\_PDU\_ACK\_SHORT\_ERR\_EID**  
 CFS CFDP Event IDs, 189  
**CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES**  
 CFS CFDP Platform Configuration, 214  
**CF\_PDU\_EOF\_SHORT\_ERR\_EID**  
 CFS CFDP Event IDs, 189  
**CF\_PDU\_FD\_SHORT\_ERR\_EID**  
 CFS CFDP Event IDs, 189  
**CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID**  
 CFS CFDP Event IDs, 189  
**CF\_PDU\_FIN\_SHORT\_ERR\_EID**  
 CFS CFDP Event IDs, 190  
**CF\_PDU\_INVALID\_DST\_LEN\_ERR\_EID**

CFS CFDP Event IDs, 190  
CF\_PDU\_INVALID\_SRC\_LEN\_ERR\_EID  
    CFS CFDP Event IDs, 190  
CF\_PDU\_LARGE\_FILE\_ERR\_EID  
    CFS CFDP Event IDs, 190  
CF\_PDU\_MAX\_SEGMENTS  
    cf\_logical\_pdu.h, 1210  
CF\_PDU\_MAX\_TLV  
    cf\_logical\_pdu.h, 1211  
CF\_PDU\_MD\_RECVD\_INF\_EID  
    CFS CFDP Event IDs, 191  
CF\_PDU\_MD\_SHORT\_ERR\_EID  
    CFS CFDP Event IDs, 191  
CF\_PDU\_METADATA\_ERROR  
    cf\_app.h, 837  
CF\_PDU\_NAK\_SHORT\_ERR\_EID  
    CFS CFDP Event IDs, 191  
CF\_PDU\_SHORT\_HEADER\_ERR\_EID  
    CFS CFDP Event IDs, 191  
CF\_PDU\_TRUNCATION\_ERR\_EID  
    CFS CFDP Event IDs, 192  
CF\_PduCmdMsg, 598  
    hdr, 599  
    ph, 599  
CF\_PduCmdMsg\_t  
    cf\_cfdp\_sbintf.h, 1032  
CF\_PduTlmMsg, 599  
    hdr, 600  
    ph, 600  
CF\_PduTlmMsg\_t  
    cf\_cfdp\_sbintf.h, 1032  
CF\_PERF\_ID\_APPMAIN  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_CREAT  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_CYCLE\_ENG  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_DIRREAD  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_FCLOSE  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_FOPEN  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_FREAD  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_FSEEK  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_FWRITE  
    CFS CFDP Mission Configuration, 218  
CF\_PERF\_ID\_PDURCVD  
    CFS CFDP Mission Configuration, 219  
CF\_PERF\_ID\_PDUSENT  
    CFS CFDP Mission Configuration, 219  
CF\_PERF\_ID\_RENAME

CFS CFDP Mission Configuration, 219  
CF\_PIPE\_DEPTH  
    CFS CFDP Platform Configuration, 214  
CF\_PIPE\_NAME  
    cf\_app.h, 837  
CF\_Playback, 600  
    busy, 600  
    cfdp\_class, 600  
    counted, 601  
    dest\_id, 601  
    dir\_id, 601  
    diropen, 601  
    fnames, 601  
    keep, 601  
    num\_ts, 601  
    pending\_file, 601  
    priority, 601  
CF\_PLAYBACK\_DIR\_CC  
    CFS CFDP Command Codes, 202  
CF\_Playback\_t  
    cf\_cfdp\_types.h, 1039  
CF\_PlaybackDirCmd, 602  
    cf\_cmd.c, 1093  
    cf\_cmd.h, 1125  
    CommandHeader, 602  
    Payload, 602  
CF\_PlaybackDirCmd\_t  
    CFS CFDP Command Structures, 225  
CF\_Poll, 602  
    interval\_timer, 602  
    pb, 603  
    timer\_set, 603  
CF\_Poll\_t  
    cf\_cfdp\_types.h, 1039  
CF\_PollDir, 603  
    cfdp\_class, 603  
    dest\_eid, 603  
    dst\_dir, 604  
    enabled, 604  
    interval\_sec, 604  
    priority, 604  
    src\_dir, 604  
CF\_PollDir\_t  
    default\_cf\_tbldefs.h, 812  
CF\_PrioSearch  
    cf\_utils.c, 1224  
    cf\_utils.h, 1247  
CF\_ProcessGroundCommand  
    cf\_dispatch.c, 1203  
    cf\_dispatch.h, 1205  
CF\_PURGE\_QUEUE\_CC  
    CFS CFDP Command Codes, 203  
CF\_PurgeHistory  
    cf\_cmd.c, 1093

cf\_cmd.h, 1126  
**CF\_PurgeQueueCmd**, 604  
 cf\_cmd.c, 1094  
 cf\_cmd.h, 1126  
 CommandHeader, 605  
 Payload, 605  
**CF\_PurgeQueueCmd\_t**  
 CFS CFDP Command Structures, 225  
**CF\_PurgeTransaction**  
 cf\_cmd.c, 1095  
 cf\_cmd.h, 1127  
**CF\_Queue\_active**  
 CFS CFDP Command Structures, 227  
**CF\_Queue\_all**  
 CFS CFDP Command Structures, 227  
**CF\_Queue\_history**  
 CFS CFDP Command Structures, 227  
**CF\_Queue\_pend**  
 CFS CFDP Command Structures, 227  
**CF\_Queue\_t**  
 CFS CFDP Command Structures, 227  
**CF\_QueueIdx\_FREE**  
 default\_cf\_extern\_typedefs.h, 801  
**CF\_QueueIdx\_HIST**  
 default\_cf\_extern\_typedefs.h, 801  
**CF\_QueueIdx\_HIST\_FREE**  
 default\_cf\_extern\_typedefs.h, 801  
**CF\_QueueIdx\_NUM**  
 default\_cf\_extern\_typedefs.h, 801  
 eds\_cf\_extern\_typedefs.h, 814  
**CF\_QueueIdx\_PEND**  
 default\_cf\_extern\_typedefs.h, 801  
**CF\_QueueIdx\_RX**  
 default\_cf\_extern\_typedefs.h, 801  
**CF\_QueueIdx\_t**  
 default\_cf\_extern\_typedefs.h, 801  
 eds\_cf\_extern\_typedefs.h, 814  
**CF\_QueueIdx\_TX**  
 default\_cf\_extern\_typedefs.h, 801  
**CF\_R2\_CRC\_CHUNK\_SIZE**  
 CFS CFDP Platform Configuration, 215  
**CF\_RCVMSG\_TIMEOUT**  
 CFS CFDP Platform Configuration, 215  
**CF\_REC\_PDU\_BAD\_EOF\_ERROR**  
 cf\_app.h, 837  
**CF\_REC\_PDU\_FSIZE\_MISMATCH\_ERROR**  
 cf\_app.h, 837  
**CF\_Reset\_all**  
 CFS CFDP Command Structures, 227  
**CF\_RESET\_CC**  
 CFS CFDP Command Codes, 204  
**CF\_Reset\_command**  
 CFS CFDP Command Structures, 227  
**CF\_Reset\_down**

CFS CFDP Command Structures, 227  
**CF\_Reset\_fault**  
 CFS CFDP Command Structures, 227  
**CF\_RESET\_FREED\_XACT\_DBG\_EID**  
 CFS CFDP Event IDs, 192  
**CF\_RESET\_INF\_EID**  
 CFS CFDP Event IDs, 192  
**CF\_Reset\_t**  
 CFS CFDP Command Structures, 227  
**CF\_Reset\_up**  
 CFS CFDP Command Structures, 227  
**CF\_ResetCountersCmd**, 605  
 cf\_cmd.c, 1096  
 cf\_cmd.h, 1128  
 CommandHeader, 605  
 Payload, 605  
**CF\_ResetCountersCmd\_t**  
 CFS CFDP Command Structures, 225  
**CF\_ResetHistory**  
 cf\_utils.c, 1225  
 cf\_utils.h, 1248  
**CF\_RESUME\_CC**  
 CFS CFDP Command Codes, 205  
**CF\_ResumeCmd**, 606  
 cf\_cmd.c, 1097  
 cf\_cmd.h, 1129  
 CommandHeader, 606  
 Payload, 606  
**CF\_ResumeCmd\_t**  
 CFS CFDP Command Structures, 225  
**CF\_REVISION**  
 CFS CFDP Version, 219  
**CF\_RxSubState\_COMPLETE**  
 cf\_cfdp\_types.h, 1040  
**CF\_RxSubState\_DATA\_EOF**  
 cf\_cfdp\_types.h, 1040  
**CF\_RxSubState\_DATA\_NORMAL**  
 cf\_cfdp\_types.h, 1040  
**CF\_RxSubState\_FILESTORE**  
 cf\_cfdp\_types.h, 1040  
**CF\_RxSubState\_FINACK**  
 cf\_cfdp\_types.h, 1040  
**CF\_RxSubState\_NUM\_STATES**  
 cf\_cfdp\_types.h, 1040  
**CF\_RxSubState\_t**  
 cf\_cfdp\_types.h, 1040  
**CF\_RxSubState\_VALIDATE**  
 cf\_cfdp\_types.h, 1040  
**CF\_SEND\_HK\_MID**  
 CFS CFDP Command Message IDs, 228  
**CF\_SEND\_PDU\_ERROR**  
 cf\_app.h, 837  
**CF\_SEND\_PDU\_NO\_BUF\_AVAIL\_ERROR**  
 cf\_app.h, 837

CF\_SendHkCmd, 606  
  cf\_cmd.c, 1097  
  cf\_cmd.h, 1129  
  CommandHeader, 607

CF\_SendHkCmd\_t  
  CFS CFDP Command Structures, 226

CF\_SET\_PARAM\_CC  
  CFS CFDP Command Codes, 205

CF\_SetParam\_Payload, 607  
  chan\_num, 607  
  key, 607  
  spare, 607  
  value, 607

CF\_SetParam\_Payload\_t  
  CFS CFDP Command Structures, 226

CF\_SetParamCmd, 608  
  cf\_cmd.c, 1098  
  cf\_cmd.h, 1130  
  CommandHeader, 608  
  Payload, 608

CF\_SetParamCmd\_t  
  CFS CFDP Command Structures, 226

CF\_SHORT\_PDU\_ERROR  
  cf\_app.h, 837

CF\_STARTUP\_SEM\_MAX\_RETRIES  
  CFS CFDP Platform Configuration, 215

CF\_STARTUP\_SEM\_TASK\_DELAY  
  CFS CFDP Platform Configuration, 215

CF\_StateData, 608  
  acknak\_count, 609  
  cached\_pos, 609  
  eof\_crc, 609  
  eof\_size, 609  
  fin\_dc, 609  
  fin\_fs, 609  
  peer\_cc, 610  
  sub\_state, 610

CF\_StateData\_t  
  cf\_cfdp\_types.h, 1039

CF\_StateFlags, 610  
  com, 610  
  rx, 611  
  tx, 611

CF\_StateFlags\_t  
  cf\_cfdp\_types.h, 1039

CF\_SUSPEND\_CC  
  CFS CFDP Command Codes, 206

CF\_SuspendCmd, 611  
  cf\_cmd.c, 1099  
  cf\_cmd.h, 1131  
  CommandHeader, 611  
  Payload, 611

CF\_SuspendCmd\_t  
  CFS CFDP Command Structures, 226

CF\_TableInit  
  cf\_app.c, 833  
  cf\_app.h, 841

CF\_TC\_DISPATCH\_TABLE  
  cf\_eds\_dispatch.c, 1208

CF\_THAW\_CC  
  CFS CFDP Command Codes, 207

CF\_ThawCmd, 612  
  cf\_cmd.c, 1099  
  cf\_cmd.h, 1131  
  CommandHeader, 612  
  Payload, 612

CF\_ThawCmd\_t  
  CFS CFDP Command Structures, 226

CF\_TickState\_COMPLETE  
  cf\_cfdp\_types.h, 1041

CF\_TickState\_INIT  
  cf\_cfdp\_types.h, 1040

CF\_TickState\_NUM\_TYPES  
  cf\_cfdp\_types.h, 1041

CF\_TickState\_RX\_STATE  
  cf\_cfdp\_types.h, 1040

CF\_TickState\_t  
  cf\_cfdp\_types.h, 1040

CF\_TickState\_TX\_FILEDATA  
  cf\_cfdp\_types.h, 1040

CF\_TickState\_TX\_NAK  
  cf\_cfdp\_types.h, 1040

CF\_TickState\_TX\_PEND  
  cf\_cfdp\_types.h, 1041

CF\_TickState\_TX\_STATE  
  cf\_cfdp\_types.h, 1040

CF\_Timer, 612  
  tick, 613

cf\_timer.c  
  CF\_Timer\_Expired, 1213  
  CF\_Timer\_InitRelSec, 1214  
  CF\_Timer\_Sec2Ticks, 1214  
  CF\_Timer\_Tick, 1215

cf\_timer.h  
  CF\_Timer\_Expired, 1216  
  CF\_Timer\_InitRelSec, 1217  
  CF\_Timer\_Sec2Ticks, 1217  
  CF\_Timer\_Seconds\_t, 1216  
  CF\_Timer\_t, 1216  
  CF\_Timer\_Tick, 1218  
  CF\_Timer\_Ticks\_t, 1216

CF\_Timer\_Expired  
  cf\_timer.c, 1213  
  cf\_timer.h, 1216

CF\_Timer\_InitRelSec  
  cf\_timer.c, 1214  
  cf\_timer.h, 1217

CF\_Timer\_Sec2Ticks

cf\_timer.c, 1214  
 cf\_timer.h, 1217  
**CF\_Timer\_Seconds\_t**  
 cf\_timer.h, 1216  
**CF\_Timer\_t**  
 cf\_timer.h, 1216  
**CF\_Timer\_Tick**  
 cf\_timer.c, 1215  
 cf\_timer.h, 1218  
**CF\_Timer\_Ticks\_t**  
 cf\_timer.h, 1216  
**cf\_topicids.h**  
 CFE\_MISSION\_CF\_CH0\_RX\_TOPICID, 827  
 CFE\_MISSION\_CF\_CH0\_TX\_TOPICID, 827  
 CFE\_MISSION\_CF\_CH1\_RX\_TOPICID, 827  
 CFE\_MISSION\_CF\_CH1\_TX\_TOPICID, 827  
 CFE\_MISSION\_CF\_CMD\_TOPICID, 828  
 CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID, 828  
 CFE\_MISSION\_CF\_HK\_TLM\_TOPICID, 828  
 CFE\_MISSION\_CF\_SEND\_HK\_TOPICID, 828  
 CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID, 828  
 DEFAULT\_CFE\_MISSION\_CF\_CH0\_RX\_TOPICID,  
 828  
 DEFAULT\_CFE\_MISSION\_CF\_CH0\_TX\_TOPICID,  
 828  
 DEFAULT\_CFE\_MISSION\_CF\_CH1\_RX\_TOPICID,  
 828  
 DEFAULT\_CFE\_MISSION\_CF\_CH1\_TX\_TOPICID,  
 828  
 DEFAULT\_CFE\_MISSION\_CF\_CMD\_TOPICID, 828  
 DEFAULT\_CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID,  
 829  
 DEFAULT\_CFE\_MISSION\_CF\_HK\_TLM\_TOPICID,  
 829  
 DEFAULT\_CFE\_MISSION\_CF\_SEND\_HK\_TOPICID,  
 829  
 DEFAULT\_CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID,  
 829  
**CF\_TOTAL\_CHUNKS**  
 default\_cf\_platform\_cfg.h, 811  
**CF\_TRACE**  
 cf\_assert.h, 844  
**CF\_Transaction**, 613  
 ack\_timer, 614  
 chan\_num, 614  
 chunks, 614  
 cl\_node, 614  
 crc, 614  
 fd, 614  
 flags, 615  
 foofs, 615  
 fsize, 615  
 history, 615  
 inactivity\_timer, 616  
 keep, 616  
 pb, 616  
 priority, 616  
 reliable\_mode, 616  
 state, 616  
 state\_data, 616  
**CF\_Transaction\_Payload**, 617  
 chan, 617  
 eid, 617  
 spare, 617  
 ts, 618  
**CF\_Transaction\_Payload\_t**  
 CFS CFDP Command Structures, 226  
**CF\_Transaction\_t**  
 cf\_cfdp\_types.h, 1040  
**CF\_TransactionSeq\_t**  
 default\_cf\_extern\_typedefs.h, 800  
 eds\_cf\_extern\_typedefs.h, 814  
**CF\_Traverse\_PriorityArg**, 618  
 priority, 618  
 txn, 618  
**CF\_Traverse\_PriorityArg\_t**  
 cf\_utils.h, 1239  
**CF\_Traverse\_TransSeqArg**, 618  
 src\_eid, 619  
 transaction\_sequence\_number, 619  
 txn, 619  
**CF\_Traverse\_TransSeqArg\_t**  
 cf\_utils.h, 1239  
**CF\_Traverse\_WriteHistoryFileArg**, 619  
 counter, 620  
 error, 620  
 fd, 620  
 filter\_dir, 620  
**CF\_Traverse\_WriteHistoryFileArg\_t**  
 cf\_utils.h, 1240  
**CF\_Traverse\_WriteHistoryQueueEntryToFile**  
 cf\_utils.c, 1226  
 cf\_utils.h, 1248  
**CF\_Traverse\_WriteTxnFileArg**, 620  
 counter, 621  
 error, 621  
 fd, 621  
**CF\_Traverse\_WriteTxnFileArg\_t**  
 cf\_utils.h, 1240  
**CF\_Traverse\_WriteTxnQueueEntryToFile**  
 cf\_utils.c, 1227  
 cf\_utils.h, 1249  
**CF\_TraverseAll\_Arg**, 621  
 context, 621  
 counter, 622  
 fn, 622  
**CF\_TraverseAll\_Arg\_t**  
 cf\_utils.h, 1240

CF\_TraverseAllTransactions  
  cf\_utils.c, 1227  
  cf\_utils.h, 1250  
CF\_TraverseAllTransactions\_All\_Channels  
  cf\_utils.c, 1228  
  cf\_utils.h, 1251  
CF\_TraverseAllTransactions\_fn\_t  
  cf\_utils.h, 1240  
CF\_TraverseAllTransactions\_Impl  
  cf\_utils.c, 1229  
  cf\_utils.h, 1251  
CF\_TsnChanAction  
  cf\_cmd.c, 1100  
  cf\_cmd.h, 1132  
CF\_TsnChanAction\_fn\_t  
  cf\_cmd.h, 1108  
CF\_TX\_FILE\_CC  
  CFS CFDP Command Codes, 208  
CF\_TxFile\_Payload, 622  
  cfdp\_class, 622  
  chan\_num, 623  
  dest\_id, 623  
  dst\_filename, 623  
  keep, 623  
  priority, 623  
  src\_filename, 623  
CF\_TxFile\_Payload\_t  
  CFS CFDP Command Structures, 226  
CF\_TxFileCmd, 623  
  cf\_cmd.c, 1101  
  cf\_cmd.h, 1133  
  CommandHeader, 624  
  Payload, 624  
CF\_TxFileCmd\_t  
  CFS CFDP Command Structures, 226  
CF\_TxnFilenames, 624  
  dst\_filename, 624  
  src\_filename, 624  
CF\_TxnFilenames\_t  
  default\_cf\_extern\_typedefs.h, 800  
CF\_TxnState\_DROP  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_HOLD  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_INIT  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_INVALID  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_R1  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_R2  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_S1  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_S2  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_t  
  cf\_cfdp\_types.h, 1041  
CF\_TxnState\_UNDEF  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_ACK\_LIMIT\_NO\_EOF  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_ACK\_LIMIT\_NO\_FIN  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_CANCEL\_REQUEST\_RECEIVED  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_CHECK\_LIMIT\_REACHED  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_EARLY\_FIN  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_FILE\_CHECKSUM\_FAILURE  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_FILE\_SIZE\_ERROR  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_FILESTORE\_REJECTION  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_From\_ConditionCode  
  cf\_utils.c, 1229  
  cf\_utils.h, 1252  
CF\_TxnStatus\_INACTIVITY\_DETECTED  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_INVALID\_FILE\_STRUCTURE  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_INVALID\_TRANSMISSION\_MODE  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_IsError  
  cf\_utils.h, 1252  
CF\_TxnStatus\_KEEP\_ALIVE\_LIMIT\_REACHED  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_MAX  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_NAK\_LIMIT\_REACHED  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_NAK\_RESPONSE\_ERROR  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_NO\_ERROR  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_NO\_RESOURCE  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_POS\_ACK\_LIMIT\_REACHED  
  cf\_cfdp\_types.h, 1041  
CF\_TxnStatus\_PROTOCOL\_ERROR  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_READ\_FAILURE  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_SEND\_EOF\_FAILURE  
  cf\_cfdp\_types.h, 1042  
CF\_TxnStatus\_SUSPEND\_REQUEST\_RECEIVED

cf\_cfdp\_types.h, 1042  
 CF\_TxnStatus\_t  
   cf\_cfdp\_types.h, 1041  
 CF\_TxnStatus\_To\_ConditionCode  
   cf\_utils.c, 1230  
   cf\_utils.h, 1253  
 CF\_TxnStatus\_UNDEFINED  
   cf\_cfdp\_types.h, 1041  
 CF\_TxnStatus\_UNSUPPORTED\_CHECKSUM\_TYPE  
   cf\_cfdp\_types.h, 1042  
 CF\_TxSubState\_COMPLETE  
   cf\_cfdp\_types.h, 1042  
 CF\_TxSubState\_DATA\_EOF  
   cf\_cfdp\_types.h, 1042  
 CF\_TxSubState\_DATA\_NORMAL  
   cf\_cfdp\_types.h, 1042  
 CF\_TxSubState\_FILESTORE  
   cf\_cfdp\_types.h, 1042  
 CF\_TxSubState\_NUM\_STATES  
   cf\_cfdp\_types.h, 1042  
 CF\_TxSubState\_t  
   cf\_cfdp\_types.h, 1042  
 CF\_Type\_all  
   CFS CFDP Command Structures, 228  
 CF\_Type\_down  
   CFS CFDP Command Structures, 228  
 CF\_Type\_t  
   CFS CFDP Command Structures, 228  
 CF\_Type\_up  
   CFS CFDP Command Structures, 228  
 CF\_UnionArgs\_Payload, 625  
   byte, 625  
   dword, 625  
   hword, 625  
 CF\_UnionArgs\_Payload\_t  
   CFS CFDP Command Structures, 226  
 cf\_utils.c  
   CF\_CFDP\_GetAckTxnStatus, 1219  
   CF\_FindTransactionBySequenceNumber, 1220  
   CF\_FindTransactionBySequenceNumber\_Impl, 1221  
   CF\_FindUnusedTransaction, 1221  
   CF\_FreeTransaction, 1222  
   CF\_GetChannelFromTxn, 1223  
   CF\_GetChunkListHead, 1223  
   CF\_InsertSortPrio, 1224  
   CF\_PrioSearch, 1224  
   CF\_ResetHistory, 1225  
   CF\_Traverse\_WriteHistoryQueueEntryToFile, 1226  
   CF\_Traverse\_WriteTxnQueueEntryToFile, 1227  
   CF\_TraverseAllTransactions, 1227  
   CF\_TraverseAllTransactions\_All\_Channels, 1228  
   CF\_TraverseAllTransactions\_Impl, 1229  
   CF\_TxnStatus\_From\_ConditionCode, 1229  
   CF\_TxnStatus\_To\_ConditionCode, 1230  
   CF\_WrappedClose, 1231  
   CF\_WrappedLseek, 1231  
   CF\_WrappedOpenCreate, 1232  
   CF\_WrappedRead, 1233  
   CF\_WrappedWrite, 1234  
   CF\_WriteHistoryEntryToFile, 1235  
   CF\_WriteHistoryQueueDataToFile, 1236  
   CF\_WriteTxnQueueDataToFile, 1237  
 cf\_utils.h  
   CF\_CFDP\_GetAckTxnStatus, 1240  
   CF\_CList\_InsertAfter\_Ex, 1241  
   CF\_CList\_InsertBack\_Ex, 1241  
   CF\_CList\_Remove\_Ex, 1241  
   CF\_DequeueTransaction, 1242  
   CF\_FindTransactionBySequenceNumber, 1242  
   CF\_FindTransactionBySequenceNumber\_Impl, 1243  
   CF\_FindUnusedTransaction, 1244  
   CF\_FreeTransaction, 1244  
   CF\_GetChannelFromTxn, 1245  
   CF\_GetChunkListHead, 1245  
   CF\_InsertSortPrio, 1246  
   CF\_MoveTransaction, 1247  
   CF\_PrioSearch, 1247  
   CF\_ResetHistory, 1248  
   CF\_Traverse\_PriorityArg\_t, 1239  
   CF\_Traverse\_TransSeqArg\_t, 1239  
   CF\_Traverse\_WriteHistoryFileArg\_t, 1240  
   CF\_Traverse\_WriteHistoryQueueEntryToFile, 1248  
   CF\_Traverse\_WriteTxnFileArg\_t, 1240  
   CF\_Traverse\_WriteTxnQueueEntryToFile, 1249  
   CF\_TraverseAll\_Arg\_t, 1240  
   CF\_TraverseAllTransactions, 1250  
   CF\_TraverseAllTransactions\_All\_Channels, 1251  
   CF\_TraverseAllTransactions\_fn\_t, 1240  
   CF\_TraverseAllTransactions\_Impl, 1251  
   CF\_TxnStatus\_From\_ConditionCode, 1252  
   CF\_TxnStatus\_IsError, 1252  
   CF\_TxnStatus\_To\_ConditionCode, 1253  
   CF\_WrappedClose, 1254  
   CF\_WrappedLseek, 1254  
   CF\_WrappedOpenCreate, 1255  
   CF\_WrappedRead, 1256  
   CF\_WrappedWrite, 1257  
   CF\_WriteHistoryEntryToFile, 1258  
   CF\_WriteHistoryQueueDataToFile, 1259  
   CF\_WriteTxnQueueDataToFile, 1260  
 CF\_ValidateChunkSizeCmd  
   cf\_cmd.c, 1102  
   cf\_cmd.h, 1134  
 CF\_ValidateConfigTable  
   cf\_app.c, 834  
   cf\_app.h, 842  
 CF\_ValidateMaxOutgoingCmd  
   cf\_cmd.c, 1102

cf\_cmd.h, 1134  
CF\_WAKE\_UP\_MID  
    CFS CFDP Command Message IDs, 228  
CF\_WakeupCmd, 625  
    cf\_cmd.c, 1103  
    cf\_cmd.h, 1135  
    CommandHeader, 626  
CF\_WakeupCmd\_t  
    CFS CFDP Command Structures, 226  
CF\_WrappedClose  
    cf\_utils.c, 1231  
    cf\_utils.h, 1254  
CF\_WrappedLseek  
    cf\_utils.c, 1231  
    cf\_utils.h, 1254  
CF\_WrappedOpenCreate  
    cf\_utils.c, 1232  
    cf\_utils.h, 1255  
CF\_WrappedRead  
    cf\_utils.c, 1233  
    cf\_utils.h, 1256  
CF\_WrappedWrite  
    cf\_utils.c, 1234  
    cf\_utils.h, 1257  
CF\_WRITE\_QUEUE\_CC  
    CFS CFDP Command Codes, 208  
CF\_WriteHistoryEntryToFile  
    cf\_utils.c, 1235  
    cf\_utils.h, 1258  
CF\_WriteHistoryQueueDataToFile  
    cf\_utils.c, 1236  
    cf\_utils.h, 1259  
CF\_WriteQueue\_Payload, 626  
    chan, 626  
    filename, 626  
    queue, 627  
    spare, 627  
    type, 627  
CF\_WriteQueue\_Payload\_t  
    CFS CFDP Command Structures, 226  
CF\_WriteQueueCmd, 627  
    cf\_cmd.c, 1104  
    cf\_cmd.h, 1136  
    CommandHeader, 627  
    Payload, 627  
CF\_WriteQueueCmd\_t  
    CFS CFDP Command Structures, 226  
CF\_WriteTxnQueueDataToFile  
    cf\_utils.c, 1237  
    cf\_utils.h, 1260  
cfdp\_class  
    CF\_Playback, 600  
    CF\_PollDir, 603  
    CF\_TxFile\_Payload, 622  
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, 258  
    CFE\_ES\_ExitApp, 259  
    CFE\_ES\_IncrementTaskCounter, 259  
    CFE\_ES\_RunLoop, 260  
    CFE\_ES\_WaitForStartupSync, 260  
    CFE\_ES\_WaitForSystemState, 261  
cFE Application Control APIs, 256  
    CFE\_ES\_DeleteApp, 256  
    CFE\_ES\_ReloadApp, 257  
    CFE\_ES\_RestartApp, 258  
cFE Child Task APIs, 270  
    CFE\_ES\_CreateChildTask, 271  
    CFE\_ES\_DeleteChildTask, 272  
    CFE\_ES\_ExitChildTask, 272  
    CFE\_ES\_GetTaskIDByName, 273  
    CFE\_ES\_GetTaskName, 273  
cFE Clock State Flag Defines, 393  
    CFE\_TIME\_FLAG\_ADD1HZ, 394  
    CFE\_TIME\_FLAG\_ADDADJ, 394  
    CFE\_TIME\_FLAG\_ADDTCL, 394  
    CFE\_TIME\_FLAG\_CLKSET, 394  
    CFE\_TIME\_FLAG\_CMDFLY, 394  
    CFE\_TIME\_FLAG\_FLYING, 394  
    CFE\_TIME\_FLAG\_GDTONE, 395  
    CFE\_TIME\_FLAG\_REFERR, 395  
    CFE\_TIME\_FLAG\_SERVER, 395  
    CFE\_TIME\_FLAG\_SIGPRI, 395  
    CFE\_TIME\_FLAG\_SRCINT, 395  
    CFE\_TIME\_FLAG\_SRVFLY, 395  
    CFE\_TIME\_FLAG\_UNUSED, 395  
cFE Critical Data Store APIs, 276  
    CFE\_ES\_CopyToCDS, 277  
    CFE\_ES\_GetCDSBlockIDByName, 277  
    CFE\_ES\_GetCDSBlockName, 278  
    CFE\_ES\_RegisterCDS, 279  
    CFE\_ES\_RestoreFromCDS, 280  
cFE Entry/Exit APIs, 254  
    CFE\_ES\_Main, 255  
    CFE\_ES\_ResetCFE, 255  
cFE External Time Source APIs, 386  
    CFE\_TIME\_ExternalGPS, 386  
    CFE\_TIME\_ExternalMET, 387  
    CFE\_TIME\_ExternalTime, 388  
    CFE\_TIME\_ExternalTone, 388  
    CFE\_TIME\_RegisterSynchCallback, 389  
    CFE\_TIME\_UnregisterSynchCallback, 390  
cFE File Header Management APIs, 303  
    CFE\_FS\_InitHeader, 303  
    CFE\_FS\_ReadHeader, 303

- CFE\_FS\_SetTimestamp, 304
- CFE\_FS\_WriteHeader, 305
- cFE File Utility APIs, 306
  - CFE\_FS\_BackgroundFileDumpIsPending, 307
  - CFE\_FS\_BackgroundFileDumpRequest, 307
  - CFE\_FS\_ExtractFilenameFromPath, 308
  - CFE\_FS\_GetDefaultExtension, 308
  - CFE\_FS\_GetDefaultMountPoint, 309
  - CFE\_FS\_ParseInputFileName, 309
  - CFE\_FS\_ParseInputFileNameEx, 310
- cFE Generic Counter APIs, 291
  - CFE\_ES\_DeleteGenCounter, 291
  - CFE\_ES\_GetGenCount, 292
  - CFE\_ES\_GetGenCounterIDByName, 292
  - CFE\_ES\_GetGenCounterName, 293
  - CFE\_ES\_IncrementGenCounter, 294
  - CFE\_ES\_RegisterGenCounter, 294
  - CFE\_ES\_SetGenCount, 295
- cFE Generic Message APIs, 311
  - CFE\_MSG\_Init, 311
- cFE Get Current Time APIs, 377
  - CFE\_TIME\_GetMET, 377
  - CFE\_TIME\_GetMETseconds, 377
  - CFE\_TIME\_GetMETsubsecs, 378
  - CFE\_TIME\_GetTAI, 378
  - CFE\_TIME\_GetTime, 379
  - CFE\_TIME\_GetUTC, 379
- cFE Get Table Information APIs, 372
  - CFE\_TBL\_GetInfo, 372
  - CFE\_TBL\_GetStatus, 373
  - CFE\_TBL\_NotifyByMessage, 374
- cFE Get Time Information APIs, 380
  - CFE\_TIME\_GetClockInfo, 380
  - CFE\_TIME\_GetClockState, 380
  - CFE\_TIME\_GetLeapSeconds, 381
  - CFE\_TIME\_GetSTCF, 381
- cFE Information APIs, 262
  - CFE\_ES\_GetAppID, 262
  - CFE\_ES\_GetAppIDByName, 263
  - CFE\_ES\_GetAppInfo, 264
  - CFE\_ES\_GetAppName, 264
  - CFE\_ES\_GetLibIDByName, 265
  - CFE\_ES\_GetLibInfo, 266
  - CFE\_ES\_GetLibName, 267
  - CFE\_ES\_GetModuleInfo, 267
  - CFE\_ES\_GetResetType, 268
  - CFE\_ES\_GetTaskID, 269
  - CFE\_ES\_GetTaskInfo, 270
- cFE Manage Table Content APIs, 362
  - CFE\_TBL\_DumpToBuffer, 362
  - CFE\_TBL\_Load, 363
  - CFE\_TBL\_Manage, 365
  - CFE\_TBL\_Modified, 365
  - CFE\_TBL\_Update, 366
- CFE\_TBL\_Validate, 367
- cFE Memory Manager APIs, 281
  - CFE\_ES\_GetMemPoolStats, 281
  - CFE\_ES\_GetPoolBuf, 282
  - CFE\_ES\_GetPoolBufInfo, 283
  - CFE\_ES\_PoolCreate, 283
  - CFE\_ES\_PoolCreateEx, 284
  - CFE\_ES\_PoolCreateEx\_WithAlignment, 285
  - CFE\_ES\_PoolCreateNoSem, 286
  - CFE\_ES\_PoolDelete, 287
  - CFE\_ES\_PutPoolBuf, 288
- cFE Message Characteristics APIs, 348
  - CFE\_SB\_GetUserData, 348
  - CFE\_SB\_GetUserDataLength, 349
  - CFE\_SB\_MessageStringGet, 349
  - CFE\_SB\_MessageStringSet, 350
  - CFE\_SB\_SetUserDataLength, 351
  - CFE\_SB\_TimeStampMsg, 351
- cFE Message Extended Header APIs, 320
  - CFE\_MSG\_GetEDSVersion, 321
  - CFE\_MSG\_GetEndian, 321
  - CFE\_MSG\_GetPlaybackFlag, 322
  - CFE\_MSG\_GetSubsystem, 322
  - CFE\_MSG\_GetSystem, 323
  - CFE\_MSG\_SetEDSVersion, 323
  - CFE\_MSG\_SetEndian, 324
  - CFE\_MSG\_SetPlaybackFlag, 324
  - CFE\_MSG\_SetSubsystem, 325
  - CFE\_MSG\_SetSystem, 325
- cFE Message ID APIs, 352
  - CFE\_SB\_CmdTopicIdToMsgId, 352
  - CFE\_SB\_GlobalCmdTopicIdToMsgId, 353
  - CFE\_SB\_GlobalTImTopicIdToMsgId, 353
  - CFE\_SB\_IsValidMsgId, 354
  - CFE\_SB\_LocalCmdTopicIdToMsgId, 354
  - CFE\_SB\_LocalTImTopicIdToMsgId, 355
  - CFE\_SB\_MsgId\_Equal, 355
  - CFE\_SB\_MsgIdToValue, 356
  - CFE\_SB\_TImTopicIdToMsgId, 356
  - CFE\_SB\_ValueToMsgId, 356
- cFE Message Id APIs, 330
  - CFE\_MSG\_GetMsgId, 330
  - CFE\_MSG\_GetTypeFromMsgId, 331
  - CFE\_MSG\_SetMsgId, 331
- cFE Message Integrity APIs, 332
  - CFE\_MSG\_OriginationAction, 332
  - CFE\_MSG\_VerificationAction, 333
- cFE Message Primary Header APIs, 312
  - CFE\_MSG\_GetApId, 312
  - CFE\_MSG\_GetHasSecondaryHeader, 313
  - CFE\_MSG\_GetHeaderVersion, 313
  - CFE\_MSG\_GetNextSequenceCount, 314
  - CFE\_MSG\_GetSegmentationFlag, 314
  - CFE\_MSG\_GetSequenceCount, 315

CFE\_MSG\_GetSize, 315  
CFE\_MSG.GetType, 316  
CFE\_MSG\_SetApId, 316  
CFE\_MSG\_SetHasSecondaryHeader, 317  
CFE\_MSG\_SetHeaderVersion, 317  
CFE\_MSG\_SetSegmentationFlag, 318  
CFE\_MSG\_SetSequenceCount, 318  
CFE\_MSG\_SetSize, 319  
CFE\_MSG\_SetType, 319  
cFE Message Secondary Header APIs, 326  
CFE\_MSG\_GenerateChecksum, 326  
CFE\_MSG\_GetFcnCode, 327  
CFE\_MSG\_GetMsgTime, 327  
CFE\_MSG\_SetFcnCode, 328  
CFE\_MSG\_SetMsgTime, 329  
CFE\_MSG\_ValidateChecksum, 329  
cFE Message Subscription Control APIs, 339  
CFE\_SB\_Subscribe, 339  
CFE\_SB\_SubscribeEx, 340  
CFE\_SB\_SubscribeLocal, 341  
CFE\_SB\_Unsubscribe, 342  
CFE\_SB\_UnsubscribeLocal, 342  
cFE Miscellaneous APIs, 274  
CFE\_ES\_BackgroundWakeup, 274  
CFE\_ES\_CalculateCRC, 275  
CFE\_ES\_ProcessAsyncEvent, 275  
CFE\_ES\_WriteToSysLog, 276  
cFE Miscellaneous Time APIs, 390  
CFE\_TIME\_Local1HzISR, 390  
CFE\_TIME\_Print, 391  
cFE Performance Monitor APIs, 289  
CFE\_ES\_PerfLogAdd, 290  
CFE\_ES\_PerfLogEntry, 289  
CFE\_ES\_PerfLogExit, 289  
cFE Pipe Management APIs, 334  
CFE\_SB\_CreatePipe, 334  
CFE\_SB\_DeletePipe, 335  
CFE\_SB\_GetPipeIdByName, 336  
CFE\_SB\_GetPipeName, 336  
CFE\_SB\_GetPipeOpts, 337  
CFE\_SB\_PipeId\_ToIndex, 338  
CFE\_SB\_SetPipeOpts, 338  
cFE Registration APIs, 296, 357  
CFE\_EVS\_Register, 296  
CFE\_TBL\_Register, 358  
CFE\_TBL\_Share, 360  
CFE\_TBL\_Unregister, 361  
cFE Reset Event Filter APIs, 301  
CFE\_EVS\_ResetAllFilters, 301  
CFE\_EVS\_ResetFilter, 302  
cFE Resource ID APIs, 252  
CFE\_ES\_AppID\_ToIndex, 252  
CFE\_ES\_CounterID\_ToIndex, 253  
CFE\_ES\_LibID\_ToIndex, 253  
CFE\_ES\_TaskID\_ToIndex, 254  
cFE Resource ID base values, 392  
CFE\_CONFIGID\_BASE, 393  
CFE\_ES\_APPID\_BASE, 393  
CFE\_ES\_CDSDBLOCKID\_BASE, 393  
CFE\_ES\_COUNTID\_BASE, 393  
CFE\_ES\_LIBID\_BASE, 393  
CFE\_ES\_POOLID\_BASE, 393  
CFE\_ES\_TASKID\_BASE, 393  
CFE\_RESOURCEID\_CONFIGID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_ES\_APPID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_ES\_CDSDBLOCKID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_ES\_COUNTID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_ES\_LIBID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_ES\_POOLID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_ES\_TASKID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_SB\_PIPEID\_RESOURCE\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_TBL\_ACCESSID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_TBL\_DUMPCTRLID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_TBL\_LOADBUFFID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_TBL\_REGID\_BASE\_OFFSET, 393  
CFE\_RESOURCEID\_TBL\_VALRESULTID\_BASE\_OFFSET, 393  
CFE\_SB\_PIPEID\_BASE, 393  
CFE\_TBL\_DUMPCTRLID\_BASE, 393  
CFE\_TBL\_HANDLE\_BASE, 393  
CFE\_TBL\_LOADBUFFID\_BASE, 393  
CFE\_TBL\_REGID\_BASE, 393  
CFE\_TBL\_VALRESULTID\_BASE, 393  
cFE Return Code Defines, 229  
CFE\_ES\_APP\_CLEANUP\_ERR, 235  
CFE\_ES\_BAD\_ARGUMENT, 235  
CFE\_ES\_BIN\_SEM\_DELETE\_ERR, 235  
CFE\_ES\_BUFFER\_NOT\_IN\_POOL, 235  
CFE\_ES\_CDS\_ACCESS\_ERROR, 235  
CFE\_ES\_CDS\_ALREADY\_EXISTS, 235  
CFE\_ES\_CDS\_BLOCK\_CRC\_ERR, 236  
CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY, 236  
CFE\_ES\_CDS\_INVALID, 236  
CFE\_ES\_CDS\_INVALID\_NAME, 236  
CFE\_ES\_CDS\_INVALID\_SIZE, 236  
CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR, 236

CFE\_ES\_CDS\_WRONG\_TYPE\_ERR, 236  
 CFE\_ES\_COUNT\_SEM\_DELETE\_ERR, 236  
 CFE\_ES\_ERR\_APP\_CREATE, 237  
 CFE\_ES\_ERR\_APP\_REGISTER, 237  
 CFE\_ES\_ERR\_CHILD\_TASK\_CREATE, 237  
 CFE\_ES\_ERR\_CHILD\_TASK\_DELETE, 237  
 CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK,  
     237  
 CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER, 237  
 CFE\_ES\_ERR\_DUPLICATE\_NAME, 237  
 CFE\_ES\_ERR\_LOAD\_LIB, 237  
 CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE, 238  
 CFE\_ES\_ERR\_NAME\_NOT\_FOUND, 238  
 CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID, 238  
 CFE\_ES\_ERR\_SYS\_LOG\_FULL, 238  
 CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED, 238  
 CFE\_ES\_FILE\_CLOSE\_ERR, 238  
 CFE\_ES\_FILE\_IO\_ERR, 238  
 CFE\_ES\_LIB\_ALREADY\_LOADED, 238  
 CFE\_ES\_MUT\_SEM\_DELETE\_ERR, 238  
 CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE, 239  
 CFE\_ES\_NOT\_IMPLEMENTED, 239  
 CFE\_ES\_OPERATION\_TIMED\_OUT, 239  
 CFE\_ES\_POOL\_BLOCK\_INVALID, 239  
 CFE\_ES\_QUEUE\_DELETE\_ERR, 239  
 CFE\_ES\_RST\_ACCESS\_ERR, 239  
 CFE\_ES\_TASK\_DELETE\_ERR, 239  
 CFE\_ES\_TIMER\_DELETE\_ERR, 239  
 CFE\_EVS\_APP\_FILTER\_OVERLOAD, 240  
 CFE\_EVS\_APP\_ILLEGAL\_APP\_ID, 240  
 CFE\_EVS\_APP\_NOT\_REGISTERED, 240  
 CFE\_EVS\_APP\_SQUELCHED, 240  
 CFE\_EVS\_EVT\_NOT\_REGISTERED, 240  
 CFE\_EVS\_FILE\_WRITE\_ERROR, 240  
 CFE\_EVS\_INVALID\_PARAMETER, 240  
 CFE\_EVS\_NOT\_IMPLEMENTED, 240  
 CFE\_EVS\_RESET\_AREA\_POINTER, 241  
 CFE\_EVS\_UNKNOWN\_FILTER, 241  
 CFE\_FS\_BAD\_ARGUMENT, 241  
 CFE\_FS\_FNAME\_TOO\_LONG, 241  
 CFE\_FS\_INVALID\_PATH, 241  
 CFE\_FS\_NOT\_IMPLEMENTED, 241  
 CFE\_SB\_BAD\_ARGUMENT, 241  
 CFE\_SB\_BUF\_ALOC\_ERR, 241  
 CFE\_SB\_BUFFER\_INVALID, 242  
 CFE\_SB\_INTERNAL\_ERR, 242  
 CFE\_SB\_MAX\_DESTS\_MET, 242  
 CFE\_SB\_MAX\_MSGS\_MET, 242  
 CFE\_SB\_MAX\_PIPES\_MET, 242  
 CFE\_SB\_MSG\_TOO\_BIG, 242  
 CFE\_SB\_NO\_MESSAGE, 242  
 CFE\_SB\_NOT\_IMPLEMENTED, 243  
 CFE\_SB\_PIPE\_CR\_ERR, 243  
 CFE\_SB\_PIPE\_RD\_ERR, 243  
 CFE\_SB\_TIME\_OUT, 243  
 CFE\_SB\_WRONG\_MSG\_TYPE, 243  
 CFE\_STATUS\_BAD\_COMMAND\_CODE, 243  
 CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL, 243  
 CFE\_STATUS\_INCORRECT\_STATE, 244  
 CFE\_STATUS\_NO\_COUNTER\_INCREMENT, 244  
 CFE\_STATUS\_NOT\_IMPLEMENTED, 244  
 CFE\_STATUS\_RANGE\_ERROR, 244  
 CFE\_STATUS\_REQUEST\_ALREADY\_PENDING,  
     244  
 CFE\_STATUS\_UNKNOWN\_MSG\_ID, 244  
 CFE\_STATUS\_VALIDATION\_FAILURE, 244  
 CFE\_STATUS\_WRONG\_MSG\_LENGTH, 244  
 CFE\_SUCCESS, 245  
 CFE\_TBL\_BAD\_ARGUMENT, 245  
 CFE\_TBL\_ERR\_ACCESS, 245  
 CFE\_TBL\_ERR\_BAD\_CONTENT\_ID, 245  
 CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID, 245  
 CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID, 245  
 CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID, 245  
 CFE\_TBL\_ERR\_DUMP\_ONLY, 245  
 CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE, 246  
 CFE\_TBL\_ERR\_DUPLICATE\_NOT\_OWNED, 246  
 CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE, 246  
 CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT, 246  
 CFE\_TBL\_ERR\_FILE\_TOO\_LARGE, 246  
 CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG, 246  
 CFE\_TBL\_ERR\_HANDLES\_FULL, 246  
 CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE, 246  
 CFE\_TBL\_ERR\_INVALID\_HANDLE, 247  
 CFE\_TBL\_ERR\_INVALID\_NAME, 247  
 CFE\_TBL\_ERR\_INVALID\_OPTIONS, 247  
 CFE\_TBL\_ERR\_INVALID\_SIZE, 247  
 CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS, 247  
 CFE\_TBL\_ERR\_LOAD\_INCOMPLETE, 247  
 CFE\_TBL\_ERR\_NEVER\_LOADED, 248  
 CFE\_TBL\_ERR\_NO\_ACCESS, 248  
 CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL, 248  
 CFE\_TBL\_ERR\_NO\_STD\_HEADER, 248  
 CFE\_TBL\_ERR\_NO\_TBL\_HEADER, 248  
 CFE\_TBL\_ERR\_PARTIAL\_LOAD, 248  
 CFE\_TBL\_ERR\_REGISTRY\_FULL, 248  
 CFE\_TBL\_ERR\_SHORT\_FILE, 248  
 CFE\_TBL\_ERR\_UNREGISTERED, 249  
 CFE\_TBL\_INFO\_DUMP\_PENDING, 249  
 CFE\_TBL\_INFO\_NO\_DUMP\_PENDING, 249  
 CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING, 249  
 CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING, 249  
 CFE\_TBL\_INFO\_RECOVERED\_TBL, 249  
 CFE\_TBL\_INFO\_TABLE\_LOCKED, 249  
 CFE\_TBL\_INFO\_UPDATE\_PENDING, 249  
 CFE\_TBL\_INFO\_UPDATED, 250  
 CFE\_TBL\_INFO\_VALIDATION\_PENDING, 250  
 CFE\_TBL\_MESSAGE\_ERROR, 250

CFE\_TBL\_NOT\_IMPLEMENTED, 250  
CFE\_TBL\_WARN\_DUPLICATE, 250  
CFE\_TBL\_WARN\_NOT\_CRITICAL, 250  
CFE\_TBL\_WARN\_PARTIAL\_LOAD, 250  
CFE\_TBL\_WARN\_SHORT\_FILE, 251  
CFE\_TIME\_BAD\_ARGUMENT, 251  
CFE\_TIME\_CALLBACK\_NOT\_REGISTERED, 251  
CFE\_TIME\_INTERNAL\_ONLY, 251  
CFE\_TIME\_NOT\_IMPLEMENTED, 251  
CFE\_TIME\_OUT\_OF\_RANGE, 251  
CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS, 251  
cFE SB Pipe options, 357  
  CFE\_SB\_PIPEOPTS\_IGNOREMINE, 357  
cFE Send Event APIs, 297  
  CFE\_EVS\_SendEvent, 297  
  CFE\_EVS\_SendEventWithApplID, 299  
  CFE\_EVS\_SendTimedEvent, 300  
cFE Send/Receive Message APIs, 343  
  CFE\_SB\_ReceiveBuffer, 343  
  CFE\_SB\_TransmitMsg, 344  
cFE Table Type Defines, 375  
  CFE\_TBL\_OPT\_BUFFER\_MSK, 375  
  CFE\_TBL\_OPT\_CRITICAL, 375  
  CFE\_TBL\_OPT\_CRITICAL\_MSK, 375  
  CFE\_TBL\_OPT\_DBL\_BUFFER, 376  
  CFE\_TBL\_OPT\_DEFAULT, 376  
  CFE\_TBL\_OPT\_DUMP\_ONLY, 376  
  CFE\_TBL\_OPT\_LD\_DMP\_MSK, 376  
  CFE\_TBL\_OPT\_LOAD\_DUMP, 376  
  CFE\_TBL\_OPT\_NOT\_CRITICAL, 376  
  CFE\_TBL\_OPT\_NOT\_USR\_DEF, 376  
  CFE\_TBL\_OPT\_SNGL\_BUFFER, 376  
  CFE\_TBL\_OPT\_USR\_DEF\_ADDR, 376  
  CFE\_TBL\_OPT\_USR\_DEF\_MSK, 376  
cFE Time Arithmetic APIs, 382  
  CFE\_TIME\_Add, 382  
  CFE\_TIME\_Compare, 383  
  CFE\_TIME\_Subtract, 383  
cFE Time Conversion APIs, 384  
  CFE\_TIME\_MET2SCTime, 384  
  CFE\_TIME\_Micro2SubSecs, 385  
  CFE\_TIME\_Sub2MicroSecs, 385  
cFE Zero Copy APIs, 345  
  CFE\_SB\_AllocateMessageBuffer, 346  
  CFE\_SB\_ReleaseMessageBuffer, 346  
  CFE\_SB\_TransmitBuffer, 347  
cfe/cmake/sample\_defs/cfe\_perfids.h, 1267  
cfe/cmake/sample\_defs/example\_2x32bit\_cfe\_es\_memaddress.h, 1269  
cfe/cmake/sample\_defs/example\_32bit\_cfe\_es\_memaddress.h, 1271  
cfe/cmake/sample\_defs/example\_64bit\_cfe\_es\_memaddress.h, 1273  
cfe/cmake/sample\_defs/example\_mission\_cfg.h, 1274  
cfe/cmake/sample\_defs/example\_platform\_cfg.h, 1285  
cfe/docs/src/cfe\_api.dox, 1329  
cfe/docs/src/cfe\_es.dox, 1329  
cfe/docs/src/cfe\_evs.dox, 1329  
cfe/docs/src/cfe\_frontpage.dox, 1329  
cfe/docs/src/cfe\_glossary.dox, 1329  
cfe/docs/src/cfe\_sb.dox, 1329  
cfe/docs/src/cfe\_tbl.dox, 1329  
cfe/docs/src/cfe\_time.dox, 1329  
cfe/docs/src/cfe\_xref.dox, 1329  
cfe/docs/src/cfs\_versions.dox, 1329  
cfe/modules/config/fsw/inc/cfe\_config\_external.h, 1329  
cfe/modules/config/fsw/inc/cfe\_config\_init.h, 1329  
cfe/modules/config/fsw/inc/cfe\_config\_lookup.h, 1330  
cfe/modules/config/fsw/inc/cfe\_config\_nametable.h, 1330  
cfe/modules/config/fsw/inc/cfe\_config\_set.h, 1331  
cfe/modules/config/fsw/inc/cfe\_config\_table.h, 1332  
cfe/modules/core\_api/config/default\_cfe\_core\_api\_base\_msgid\_values.h, 1333  
cfe/modules/core\_api/config/default\_cfe\_core\_api\_interface\_cfg\_values.h, 1333  
cfe/modules/core\_api/config/default\_cfe\_core\_api\_mapping.h, 1334  
cfe/modules/core\_api/config/default\_cfe\_mission\_cfg.h, 1335  
cfe/modules/core\_api/config/default\_cfe\_msgids.h, 1336  
cfe/modules/core\_api/fsw/inc/cfe.h, 1336  
cfe/modules/core\_api/fsw/inc/cfe\_config.h, 1336  
cfe/modules/core\_api/fsw/inc/cfe\_config\_api\_typedefs.h, 1340  
cfe/modules/core\_api/fsw/inc/cfe\_core\_api\_base\_msgids.h, 1341  
cfe/modules/core\_api/fsw/inc/cfe\_core\_api\_interface\_cfg.h, 1342  
cfe/modules/core\_api/fsw/inc/cfe\_endian.h, 1344  
cfe/modules/core\_api/fsw/inc/cfe\_error.h, 1345  
cfe/modules/core\_api/fsw/inc/cfe\_es.h, 1353  
cfe/modules/core\_api/fsw/inc/cfe\_es\_api\_typedefs.h, 1357  
cfe/modules/core\_api/fsw/inc/cfe\_evs.h, 1362  
cfe/modules/core\_api/fsw/inc/cfe\_evs\_api\_typedefs.h, 1363  
cfe/modules/core\_api/fsw/inc/cfe\_fs.h, 1366  
cfe/modules/core\_api/fsw/inc/cfe\_fs\_api\_typedefs.h, 1366  
cfe/modules/core\_api/fsw/inc/cfe\_msg.h, 1369  
cfe/modules/core\_api/fsw/inc/cfe\_msg\_api\_typedefs.h, 1371  
cfe/modules/core\_api/fsw/inc/cfe\_resourceid.h, 1375  
cfe/modules/core\_api/fsw/inc/cfe\_resourceid\_api\_typedefs.h, 1375  
cfe/modules/core\_api/fsw/inc/cfe\_sb.h, 1382  
cfe/modules/core\_api/fsw/inc/cfe\_sb.h, 1383  
cfe/modules/core\_api/fsw/inc/cfe\_sb\_api\_typedefs.h, 1386

cfe/modules/core\_api/fsw/inc/cfe\_tbl.h, 1389  
 cfe/modules/core\_api/fsw/inc/cfe\_tbl\_api\_typedefs.h,  
   1391  
 cfe/modules/core\_api/fsw/inc/cfe\_tbl\_filedef.h, 1394  
 cfe/modules/core\_api/fsw/inc/cfe\_time.h, 1395  
 cfe/modules/core\_api/fsw/inc/cfe\_time\_api\_typedefs.h,  
   1397  
 cfe/modules/core\_api/fsw/inc/cfe\_version.h, 1399  
 cfe/modules/es/config/default\_cfe\_es\_extern\_typedefs.h,  
   1401  
 cfe/modules/es/config/default\_cfe\_es\_fcncode\_values.h,  
   1408  
 cfe/modules/es/config/default\_cfe\_es\_interface\_cfg\_values.h, 1410  
 cfe/modules/es/config/default\_cfe\_es\_internal\_cfg\_values.h, 1410  
 cfe/modules/es/config/default\_cfe\_es\_memaddress.h, 1410  
 cfe/modules/es/config/default\_cfe\_es\_mission\_cfg.h, 1412  
 cfe/modules/es/config/default\_cfe\_es\_msg.h, 1412  
 cfe/modules/es/config/default\_cfe\_es\_msgdefs.h, 1413  
 cfe/modules/es/config/default\_cfe\_esmsgid\_values.h, 1416  
 cfe/modules/es/config/default\_cfe\_es\_msgids.h, 1417  
 cfe/modules/es/config/default\_cfe\_es\_msgstruct.h, 1418  
 cfe/modules/es/config/default\_cfe\_es\_platform\_cfg.h, 1422  
 cfe/modules/es/config/default\_cfe\_es\_topicid\_values.h, 1422  
 cfe/modules/es/fsw/inc/cfe\_es\_eventids.h, 1422  
 cfe/modules/es/fsw/inc/cfe\_es\_fcncodes.h, 1448  
 cfe/modules/es/fsw/inc/cfe\_es\_interface\_cfg.h, 1469  
 cfe/modules/es/fsw/inc/cfe\_es\_internal\_cfg.h, 1473  
 cfe/modules/es/fsw/inc/cfe\_es\_topicids.h, 1505  
 cfe/modules/evs/config/default\_cfe\_evs\_extern\_typedefs.h, 1506  
 cfe/modules/evs/config/default\_cfe\_evs\_fcncode\_values.h, 1509  
 cfe/modules/evs/config/default\_cfe\_evs\_interface\_cfg\_values.h, 1511  
 cfe/modules/evs/config/default\_cfe\_evs\_internal\_cfg\_values.h, 1511  
 cfe/modules/evs/config/default\_cfe\_evs\_mission\_cfg.h, 1511  
 cfe/modules/evs/config/default\_cfe\_evs\_msg.h, 1512  
 cfe/modules/evs/config/default\_cfe\_evs\_msgdefs.h, 1512  
 cfe/modules/evs/config/default\_cfe\_evsmsgid\_values.h, 1516  
 cfe/modules/evs/config/default\_cfe\_evs\_msgids.h, 1516  
 cfe/modules/evs/config/default\_cfe\_evs\_msgstruct.h, 1517  
 cfe/modules/evs/config/default\_cfe\_evs\_platform\_cfg.h, 1520  
 cfe/modules/evs/config/default\_cfe\_evs\_topcid\_values.h, 1520  
 cfe/modules/evs/fsw/inc/cfe\_evs\_eventids.h, 1521  
 cfe/modules/evs/fsw/inc/cfe\_evs\_fcncodes.h, 1533  
 cfe/modules/evs/fsw/inc/cfe\_evs\_interface\_cfg.h, 1551  
 cfe/modules/evs/fsw/inc/cfe\_evs\_internal\_cfg.h, 1552  
 cfe/modules/evs/fsw/inc/cfe\_evs\_topcid.h, 1558  
 cfe/modules/fs/config/default\_cfe\_fs\_extern\_typedefs.h, 1560  
 cfe/modules/fs/config/default\_cfe\_fs\_filedef.h, 1560  
 cfe/modules/fs/config/default\_cfe\_fs\_interface\_cfg\_values.h, 1562  
 cfe/modules/fs/config/default\_cfe\_fs\_mission\_cfg.h, 1562  
 cfe/modules/fs/fsw/inc/cfe\_fs\_interface\_cfg.h, 1563  
 cfe/modules/msg/fsw/inc/ccsds\_hdr.h, 1564  
 cfe/modules/resourceid/fsw/inc/cfe\_core\_resourceid\_basevalues.h, 1564  
 cfe/modules/resourceid/fsw/inc/cfe\_resourceid\_basevalue.h, 1565  
 cfe/modules/sb/config/default\_cfe\_sb\_extern\_typedefs.h, 1566  
 cfe/modules/sb/config/default\_cfe\_sb\_fcncode\_values.h, 1568  
 cfe/modules/sb/config/default\_cfe\_sb\_interface\_cfg\_values.h, 1569  
 cfe/modules/sb/config/default\_cfe\_sb\_internal\_cfg\_values.h, 1570  
 cfe/modules/sb/config/default\_cfe\_sb\_mission\_cfg.h, 1570  
 cfe/modules/sb/config/default\_cfe\_sb\_msg.h, 1570  
 cfe/modules/sb/config/default\_cfe\_sb\_msgdefs.h, 1571  
 cfe/modules/sb/config/default\_cfe\_sbmsgid\_values.h, 1573  
 cfe/modules/sb/config/default\_cfe\_sb\_msgids.h, 1574  
 cfe/modules/sb/config/default\_cfe\_sb\_msgstruct.h, 1575  
 cfe/modules/sb/config/default\_cfe\_sb\_platform\_cfg.h, 1577  
 cfe/modules/sb/fsw/inc/cfe\_sb\_eventids.h, 1577  
 cfe/modules/sb/fsw/inc/cfe\_sb\_fcncodes.h, 1597  
 cfe/modules/sb/fsw/inc/cfe\_sb\_interface\_cfg.h, 1606  
 cfe/modules/sb/fsw/inc/cfe\_sb\_internal\_cfg.h, 1608  
 cfe/modules/sb/fsw/inc/cfe\_sb\_topicids.h, 1623  
 cfe/modules/tbl/config/default\_cfe\_tbl\_extern\_typedefs.h, 1625  
 cfe/modules/tbl/config/default\_cfe\_tbl\_fcncode\_values.h, 1627  
 cfe/modules/tbl/config/default\_cfe\_tbl\_interface\_cfg\_values.h, 1628  
 cfe/modules/tbl/config/default\_cfe\_tbl\_internal\_cfg\_values.h, 1628  
 cfe/modules/tbl/config/default\_cfe\_tbl\_mission\_cfg.h, 1629

cfe/modules/tbl/config/default\_cfe\_tbl\_msg.h, 1630  
cfe/modules/tbl/config/default\_cfe\_tbl\_msgdefs.h, 1630  
cfe/modules/tbl/config/default\_cfe\_tblmsgid\_values.h, 1633  
cfe/modules/tbl/config/default\_cfe\_tbl\_msgids.h, 1633  
cfe/modules/tbl/config/default\_cfe\_tbl\_msgstruct.h, 1634  
cfe/modules/tbl/config/default\_cfe\_tbl\_platform\_cfg.h, 1636  
cfe/modules/tbl/config/default\_cfe\_tbl\_topicid\_values.h, 1645  
cfe/modules/tbl/fsw/inc/cfe\_tbl\_eventids.h, 1645  
cfe/modules/tbl/fsw/inc/cfe\_tbl\_fcncodes.h, 1665  
cfe/modules/tbl/fsw/inc/cfe\_tbl\_interface\_cfg.h, 1674  
cfe/modules/tbl/fsw/inc/cfe\_tbl\_internal\_cfg.h, 1676  
cfe/modules/tbl/fsw/inc/cfe\_tbl\_topicids.h, 1685  
cfe/modules/time/config/default\_cfe\_time\_extern\_typedefs.h, 1686  
cfe/modules/time/config/default\_cfe\_time\_fcncode\_values.h, CFE\_Config\_ArrayValue, 628  
1691  
cfe/modules/time/config/default\_cfe\_time\_interface\_cfg\_values.h, NumElements, 628  
1692  
cfe/modules/time/config/default\_cfe\_time\_internal\_cfg\_values.h, cfe\_config\_api\_typedefs.h, 1341  
1692  
cfe/modules/time/config/default\_cfe\_time\_mission\_cfg.h, 1693  
cfe/modules/time/config/default\_cfe\_time\_msg.h, 1693  
cfe/modules/time/config/default\_cfe\_time\_msgdefs.h, 1694  
cfe/modules/time/config/default\_cfe\_time\_msgid\_values.h, 1696  
cfe/modules/time/config/default\_cfe\_time\_msgids.h, 1697  
cfe/modules/time/config/default\_cfe\_time\_msgstruct.h, 1698  
cfe/modules/time/config/default\_cfe\_time\_platform\_cfg.h, 1701  
cfe/modules/time/config/default\_cfe\_time\_topicid\_values.h, 1701  
cfe/modules/time/fsw/inc/cfe\_time\_eventids.h, 1702  
cfe/modules/time/fsw/inc/cfe\_time\_fcncodes.h, 1712  
cfe/modules/time/fsw/inc/cfe\_time\_interface\_cfg.h, 1727  
cfe/modules/time/fsw/inc/cfe\_time\_internal\_cfg.h, 1735  
cfe/modules/time/fsw/inc/cfe\_time\_topicids.h, 1744  
CFE\_BIT  
  cfe\_sb.h, 1385  
CFE\_BUILD\_BASELINE  
  cfe\_version.h, 1400  
CFE\_BUILD\_CODENAME  
  cfe\_version.h, 1400  
CFE\_BUILD\_DEV\_CYCLE  
  cfe\_version.h, 1400  
CFE\_BUILD\_NUMBER  
  cfe\_version.h, 1400  
CFE\_CFG\_MAX\_VERSION\_STR\_LEN  
  cfe\_version.h, 1400

CFE\_CLR  
  cfe\_sb.h, 1385  
cfe\_config.h  
  CFE\_Config\_GetArrayValue, 1337  
  CFE\_Config\_GetIdByName, 1337  
  CFE\_Config\_getName, 1338  
  CFE\_Config\_GetObjPointer, 1338  
  CFE\_Config\_getString, 1339  
  CFE\_Config\_getValue, 1339  
  CFE\_Config\_getVersionString, 1339  
  CFE\_Config\_IterateAll, 1340  
cfe\_config\_api\_typedefs.h  
  CFE\_Config\_ArrayValue\_t, 1341  
  CFE\_Config\_Callback\_t, 1341  
  CFE\_CONFIGID\_C, 1341  
  CFE\_ConfigId\_t, 1341  
  CFE\_CONFIGID\_UNDEFINED, 1341  
  CFE\_Config\_ArrayValue, 628  
  ElementPtr, 628  
  CFE\_Config\_ArrayValue\_t  
  cfe\_config\_api\_typedefs.h, 1341  
  CFE\_Config\_Callback\_t  
    cfe\_config\_api\_typedefs.h, 1341  
  cfe\_config\_external.h  
    CFE\_Config\_SetupPlatformConfigInfo, 1329  
  CFE\_Config\_GetArrayValue  
    cfe\_config.h, 1337  
  CFE\_Config\_GetIdByName  
    cfe\_config.h, 1337  
  CFE\_Config\_getName  
    cfe\_config.h, 1338  
  CFE\_Config\_GetObjPointer  
    cfe\_config.h, 1338  
  CFE\_Config\_getString  
    cfe\_config.h, 1339  
  CFE\_Config\_getValue  
    cfe\_config.h, 1339  
  CFE\_Config\_getVersionString  
    cfe\_config.h, 1339  
  CFE\_Config\_IdNameEntry, 628  
    Name, 628  
  CFE\_Config\_IdNameEntry\_t  
    cfe\_config\_nametable.h, 1331  
  CFE\_Config\_Init  
    cfe\_config\_init.h, 1330  
  cfe\_config\_init.h  
    CFE\_Config\_Init, 1330  
    CFE\_Config\_SetupBasicBuildInfo, 1330  
  CFE\_Config\_IterateAll  
    cfe\_config.h, 1340  
  CFE\_Config\_LocateConfigRecordByID  
    cfe\_config\_lookup.h, 1330  
  cfe\_config\_lookup.h



CFE\_ES\_CDSHANDLE\_C, 1359  
CFE\_ES\_ChildTaskMainFuncPtr\_t, 1360  
CFE\_ES\_COUNTERID\_C, 1359  
CFE\_ES\_COUNTERID\_UNDEFINED, 1359  
CFE\_ES\_CrcType\_16\_ARC, 1362  
CFE\_ES\_CrcType\_CRC\_16, 1362  
CFE\_ES\_CrcType\_CRC\_32, 1362  
CFE\_ES\_CrcType\_CRC\_8, 1362  
CFE\_ES\_CrcType\_Enum, 1361  
CFE\_ES\_CrcType\_Enum\_t, 1360  
CFE\_ES\_CrcType\_MAX, 1362  
CFE\_ES\_CrcType\_NONE, 1361  
CFE\_ES\_LIBID\_C, 1359  
CFE\_ES\_LIBID\_UNDEFINED, 1359  
CFE\_ES\_LibraryEntryFuncPtr\_t, 1360  
CFE\_ES\_MEMHANDLE\_C, 1359  
CFE\_ES\_MEMHANDLE\_UNDEFINED, 1359  
CFE\_ES\_MEMPOOLBUF\_C, 1359  
CFE\_ES\_MemPoolBuf\_t, 1361  
CFE\_ES\_NO\_MUTEX, 1359  
CFE\_ES\_PoolAlign\_t, 1361  
CFE\_ES\_StackPointer\_t, 1361  
CFE\_ES\_STATIC\_POOL\_TYPE, 1359  
CFE\_ES\_TASK\_STACK\_ALLOCATE, 1360  
CFE\_ES\_TaskEntryFuncPtr\_t, 1361  
CFE\_ES\_TASKID\_C, 1360  
CFE\_ES\_TASKID\_UNDEFINED, 1360  
CFE\_ES\_USE\_MUTEX, 1360  
CFE\_ES\_APP\_CLEANUP\_ERR  
    cFE Return Code Defines, 235  
CFE\_ES\_APP\_RESTART  
    cfe\_es\_api\_typedefs.h, 1358  
CFE\_ES\_APP\_TLM\_MID  
    default\_cfe\_es\_msgids.h, 1417  
CFE\_ES\_APPID\_BASE  
    cFE Resource ID base values, 393  
CFE\_ES\_APPID\_C  
    cfe\_es\_api\_typedefs.h, 1358  
CFE\_ES\_AppId\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1403  
CFE\_ES\_AppID\_ToIndex  
    cFE Resource ID APIs, 252  
CFE\_ES\_APPID\_UNDEFINED  
    cfe\_es\_api\_typedefs.h, 1358  
CFE\_ES\_AppInfo, 630  
    AddressesAreValid, 631  
    BSSAddress, 631  
    BSSSize, 631  
    CodeAddress, 631  
    CodeSize, 631  
    DataAddress, 631  
    DataSize, 632  
    EntryPoint, 632  
    ExceptionAction, 632  
    ExecutionCounter, 632  
    FileName, 632  
    MainTaskId, 632  
    MainTaskName, 632  
    Name, 633  
    NumOfChildTasks, 633  
    Priority, 633  
    Resourceld, 633  
    StackSize, 633  
    StartAddress, 633  
    Type, 633  
CFE\_ES\_AppInfo\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1403  
CFE\_ES\_AppNameCmd\_Payload, 634  
    Application, 634  
CFE\_ES\_AppNameCmd\_Payload\_t  
    default\_cfe\_es\_msgdefs.h, 1414  
CFE\_ES\_AppReloadCmd\_Payload, 634  
    AppFileName, 635  
    Application, 635  
CFE\_ES\_AppReloadCmd\_Payload\_t  
    default\_cfe\_es\_msgdefs.h, 1414  
CFE\_ES\_AppState  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppState\_EARLY\_INIT  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppState\_Enum\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1403  
CFE\_ES\_AppState\_LATE\_INIT  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppState\_MAX  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppState\_RUNNING  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppState\_STOPPED  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppState\_UNDEFINED  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppState\_WAITING  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppType  
    default\_cfe\_es\_extern\_typedefs.h, 1406  
CFE\_ES\_AppType\_CORE  
    default\_cfe\_es\_extern\_typedefs.h, 1407  
CFE\_ES\_AppType\_Enum\_t  
    default\_cfe\_es\_extern\_typedefs.h, 1403  
CFE\_ES\_AppType\_EXTERNAL  
    default\_cfe\_es\_extern\_typedefs.h, 1407  
CFE\_ES\_AppType\_LIBRARY  
    default\_cfe\_es\_extern\_typedefs.h, 1407  
CFE\_ES\_BackgroundWakeUp  
    cFE Miscellaneous APIs, 274  
CFE\_ES\_BAD\_ARGUMENT  
    cFE Return Code Defines, 235

CFE\_ES\_BIN\_SEM\_DELETE\_ERR  
cFE Return Code Defines, 235

CFE\_ES\_BlockStats, 635  
BlockSize, 635  
NumCreated, 635  
NumFree, 635

CFE\_ES\_BlockStats\_t  
default\_cfe\_es\_extern\_typedefs.h, 1404

CFE\_ES\_BOOT\_ERR\_EID  
cfe\_es\_eventids.h, 1426

CFE\_ES\_BUFFER\_NOT\_IN\_POOL  
cFE Return Code Defines, 235

CFE\_ES\_BUILD\_INF\_EID  
cfe\_es\_eventids.h, 1426

CFE\_ES\_CalculateCRC  
cFE Miscellaneous APIs, 275

CFE\_ES\_CC1\_ERR\_EID  
cfe\_es\_eventids.h, 1427

CFE\_ES\_CCVAL  
default\_cfe\_es\_fcncode\_values.h, 1409

CFE\_ES\_CDS\_ACCESS\_ERROR  
cFE Return Code Defines, 235

CFE\_ES\_CDS\_ALREADY\_EXISTS  
cFE Return Code Defines, 235

CFE\_ES\_CDS\_BAD\_HANDLE  
cfe\_es\_api\_typedefs.h, 1358

CFE\_ES\_CDS\_BLOCK\_CRC\_ERR  
cFE Return Code Defines, 236

CFE\_ES\_CDS\_DELETE\_ERR\_EID  
cfe\_es\_eventids.h, 1427

CFE\_ES\_CDS\_DELETE\_TBL\_ERR\_EID  
cfe\_es\_eventids.h, 1427

CFE\_ES\_CDS\_DELETED\_INFO\_EID  
cfe\_es\_eventids.h, 1427

CFE\_ES\_CDS\_DUMP\_ERR\_EID  
cfe\_es\_eventids.h, 1428

CFE\_ES\_CDS\_INSUFFICIENT\_MEMORY  
cFE Return Code Defines, 236

CFE\_ES\_CDS\_INVALID  
cFE Return Code Defines, 236

CFE\_ES\_CDS\_INVALID\_NAME  
cFE Return Code Defines, 236

CFE\_ES\_CDS\_INVALID\_SIZE  
cFE Return Code Defines, 236

CFE\_ES\_CDS\_NAME\_ERR\_EID  
cfe\_es\_eventids.h, 1428

CFE\_ES\_CDS\_OWNER\_ACTIVE\_EID  
cfe\_es\_eventids.h, 1428

CFE\_ES\_CDS\_OWNER\_ACTIVE\_ERR  
cFE Return Code Defines, 236

CFE\_ES\_CDS\_REG\_DUMP\_INF\_EID  
cfe\_es\_eventids.h, 1428

CFE\_ES\_CDS\_REGISTER\_ERR\_EID  
cfe\_es\_eventids.h, 1429

CFE\_ES\_CDS\_WRONG\_TYPE\_ERR  
cFE Return Code Defines, 236

CFE\_ES\_CDSBLOCKID\_BASE  
cFE Resource ID base values, 393

CFE\_ES\_CDSHANDLE\_C  
cfe\_es\_api\_typedefs.h, 1359

CFE\_ES\_CDCHandle\_t  
default\_cfe\_es\_extern\_typedefs.h, 1404

CFE\_ES\_CDSRegDumpRec, 636  
ByteAlignSpare, 636  
Handle, 636  
Name, 636  
Size, 637  
Table, 637

CFE\_ES\_CDSRegDumpRec\_t  
default\_cfe\_es\_extern\_typedefs.h, 1404

CFE\_ES\_ChildTaskMainFuncPtr\_t  
cfe\_es\_api\_typedefs.h, 1360

CFE\_ES\_CLEAR\_ER\_LOG\_CC  
cfe\_es\_fcncodes.h, 1448

CFE\_ES\_CLEAR\_SYS\_LOG\_CC  
cfe\_es\_fcncodes.h, 1449

CFE\_ES\_ClearERLogCmd, 637  
CommandHeader, 637

CFE\_ES\_ClearERLogCmd\_t  
default\_cfe\_es\_msgstruct.h, 1420

CFE\_ES\_ClearSysLogCmd, 637  
CommandHeader, 638

CFE\_ES\_ClearSysLogCmd\_t  
default\_cfe\_es\_msgstruct.h, 1420

CFE\_ES\_CMD\_MID  
default\_cfe\_es\_msgids.h, 1417

CFE\_ES\_CopyToCDS  
cFE Critical Data Store APIs, 277

CFE\_ES\_COUNT\_SEM\_DELETE\_ERR  
cFE Return Code Defines, 236

CFE\_ES\_COUNTERID\_C  
cfe\_es\_api\_typedefs.h, 1359

CFE\_ES\_CounterId\_t  
default\_cfe\_es\_extern\_typedefs.h, 1404

CFE\_ES\_CounterID\_ToIndex  
cFE Resource ID APIs, 253

CFE\_ES\_COUNTERID\_UNDEFINED  
cfe\_es\_api\_typedefs.h, 1359

CFE\_ES\_COUNTID\_BASE  
cFE Resource ID base values, 393

CFE\_ES\_CrcType\_16\_ARC  
cfe\_es\_api\_typedefs.h, 1362

CFE\_ES\_CrcType\_CRC\_16  
cfe\_es\_api\_typedefs.h, 1362

CFE\_ES\_CrcType\_CRC\_32  
cfe\_es\_api\_typedefs.h, 1362

CFE\_ES\_CrcType\_CRC\_8  
cfe\_es\_api\_typedefs.h, 1362

CFE\_ES\_CrcType\_Enum  
  cfe\_es\_api\_typedefs.h, 1361

CFE\_ES\_CrcType\_Enum\_t  
  cfe\_es\_api\_typedefs.h, 1360

CFE\_ES\_CrcType\_MAX  
  cfe\_es\_api\_typedefs.h, 1362

CFE\_ES\_CrcType\_NONE  
  cfe\_es\_api\_typedefs.h, 1361

CFE\_ES\_CreateChildTask  
  cFE Child Task APIs, 271

CFE\_ES\_CREATING\_CDS\_DUMP\_ERR\_EID  
  cfe\_es\_eventids.h, 1429

CFE\_ES\_DELETE\_CDS\_CC  
  cfe\_es\_fcncodes.h, 1450

CFE\_ES\_DeleteApp  
  cFE Application Control APIs, 256

CFE\_ES\_DeleteCDSCmd, 638  
  CommandHeader, 638  
  Payload, 638

CFE\_ES\_DeleteCDSCmd\_Payload, 638  
  CdsName, 639

CFE\_ES\_DeleteCDSCmd\_Payload\_t  
  default\_cfe\_es\_msgdefs.h, 1414

CFE\_ES\_DeleteCDSCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1420

CFE\_ES\_DeleteChildTask  
  cFE Child Task APIs, 272

CFE\_ES\_DeleteGenCounter  
  cFE Generic Counter APIs, 291

CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC  
  cfe\_es\_fcncodes.h, 1450

CFE\_ES\_DumpCDSRegistryCmd, 639  
  CommandHeader, 639  
  Payload, 639

CFE\_ES\_DumpCDSRegistryCmd\_Payload, 639  
  DumpFilename, 640

CFE\_ES\_DumpCDSRegistryCmd\_Payload\_t  
  default\_cfe\_es\_msgdefs.h, 1415

CFE\_ES\_DumpCDSRegistryCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1420

CFE\_ES\_ERLOG1\_INF\_EID  
  cfe\_es\_eventids.h, 1429

CFE\_ES\_ERLOG2\_EID  
  cfe\_es\_eventids.h, 1429

CFE\_ES\_ERLOG2\_ERR\_EID  
  cfe\_es\_eventids.h, 1430

CFE\_ES\_ERLOG\_PENDING\_ERR\_EID  
  cfe\_es\_eventids.h, 1430

CFE\_ES\_ERR\_APP\_CREATE  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_APP\_REGISTER  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_CHILD\_TASK\_CREATE  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_CHILD\_TASK\_DELETE  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_CHILD\_TASK\_DELETE\_MAIN\_TASK  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_CHILD\_TASK\_REGISTER  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_DUPLICATE\_NAME  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_LOAD\_LIB  
  cFE Return Code Defines, 237

CFE\_ES\_ERR\_MEM\_BLOCK\_SIZE  
  cFE Return Code Defines, 238

CFE\_ES\_ERR\_NAME\_NOT\_FOUND  
  cFE Return Code Defines, 238

CFE\_ES\_ERR\_RESOURCEID\_NOT\_VALID  
  cFE Return Code Defines, 238

CFE\_ES\_ERR\_SYS\_LOG\_FULL  
  cFE Return Code Defines, 238

CFE\_ES\_ERR\_SYS\_LOG\_TRUNCATED  
  cFE Return Code Defines, 238

CFE\_ES\_ERR\_SYSLOGMODE\_EID  
  cfe\_es\_eventids.h, 1430

CFE\_ES\_ERREXIT\_APP\_ERR\_EID  
  cfe\_es\_eventids.h, 1430

CFE\_ES\_ERREXIT\_APP\_INF\_EID  
  cfe\_es\_eventids.h, 1431

cfe\_es\_eventids.h

  CFE\_ES\_ALL\_APPS\_EID, 1426

  CFE\_ES\_BOOT\_ERR\_EID, 1426

  CFE\_ES\_BUILD\_INF\_EID, 1426

  CFE\_ES\_CC1\_ERR\_EID, 1427

  CFE\_ES\_CDS\_DELETE\_ERR\_EID, 1427

  CFE\_ES\_CDS\_DELETE\_TBL\_ERR\_EID, 1427

  CFE\_ES\_CDS\_DELETED\_INFO\_EID, 1427

  CFE\_ES\_CDS\_DUMP\_ERR\_EID, 1428

  CFE\_ES\_CDS\_NAME\_ERR\_EID, 1428

  CFE\_ES\_CDS\_OWNER\_ACTIVE\_EID, 1428

  CFE\_ES\_CDS\_REG\_DUMP\_INF\_EID, 1428

  CFE\_ES\_CDS\_REGISTER\_ERR\_EID, 1429

  CFE\_ES\_CREATING\_CDS\_DUMP\_ERR\_EID, 1429

  CFE\_ES\_ERLOG1\_INF\_EID, 1429

  CFE\_ES\_ERLOG2\_EID, 1429

  CFE\_ES\_ERLOG2\_ERR\_EID, 1430

  CFE\_ES\_ERLOG\_PENDING\_ERR\_EID, 1430

  CFE\_ES\_ERR\_SYSLOGMODE\_EID, 1430

  CFE\_ES\_ERREXIT\_APP\_ERR\_EID, 1430

  CFE\_ES\_ERREXIT\_APP\_INF\_EID, 1431

  CFE\_ES\_EXIT\_APP\_ERR\_EID, 1431

  CFE\_ES\_EXIT\_APP\_INF\_EID, 1431

  CFE\_ES\_FILEWRITE\_ERR\_EID, 1431

  CFE\_ES\_INIT\_INF\_EID, 1432

  CFE\_ES\_INITSTATS\_INF\_EID, 1432

  CFE\_ES\_INVALID\_POOL\_HANDLE\_ERR\_EID, 1432

CFE\_ES\_LEN\_ERR\_EID, [1432](#)  
 CFE\_ES\_MID\_ERR\_EID, [1433](#)  
 CFE\_ES\_NOOP\_INF\_EID, [1433](#)  
 CFE\_ES\_ONE\_APP\_EID, [1433](#)  
 CFE\_ES\_ONE\_APPID\_ERR\_EID, [1433](#)  
 CFE\_ES\_ONE\_ERR\_EID, [1434](#)  
 CFE\_ES\_OSCREATE\_ERR\_EID, [1434](#)  
 CFE\_ES\_PCR\_ERR1\_EID, [1434](#)  
 CFE\_ES\_PCR\_ERR2\_EID, [1434](#)  
 CFE\_ES\_PERF\_DATAWRITTEN\_EID, [1435](#)  
 CFE\_ES\_PERF\_FILTERMSKCMD\_EID, [1435](#)  
 CFE\_ES\_PERF\_FILTERMSKERR\_EID, [1435](#)  
 CFE\_ES\_PERF\_LOG\_ERR\_EID, [1435](#)  
 CFE\_ES\_PERF\_STARTCMD\_EID, [1436](#)  
 CFE\_ES\_PERF\_STARTCMD\_ERR\_EID, [1436](#)  
 CFE\_ES\_PERF\_STARTCMD\_TRIG\_ERR\_EID, [1436](#)  
 CFE\_ES\_PERF\_STOPCMD\_EID, [1436](#)  
 CFE\_ES\_PERF\_STOPCMD\_ERR2\_EID, [1437](#)  
 CFE\_ES\_PERF\_TRIGGERMSKCMD\_EID, [1437](#)  
 CFE\_ES\_PERF\_TRIGGERMSKERR\_EID, [1437](#)  
 CFE\_ES\_RELOAD\_APP\_DBG\_EID, [1437](#)  
 CFE\_ES\_RELOAD\_APP\_ERR1\_EID, [1438](#)  
 CFE\_ES\_RELOAD\_APP\_ERR2\_EID, [1438](#)  
 CFE\_ES\_RELOAD\_APP\_ERR3\_EID, [1438](#)  
 CFE\_ES\_RELOAD\_APP\_ERR4\_EID, [1438](#)  
 CFE\_ES\_RELOAD\_APP\_INF\_EID, [1439](#)  
 CFE\_ES\_RESET\_INF\_EID, [1439](#)  
 CFE\_ES\_RESET\_PR\_COUNT\_EID, [1439](#)  
 CFE\_ES\_RESTART\_APP\_DBG\_EID, [1439](#)  
 CFE\_ES\_RESTART\_APP\_ERR1\_EID, [1440](#)  
 CFE\_ES\_RESTART\_APP\_ERR2\_EID, [1440](#)  
 CFE\_ES\_RESTART\_APP\_ERR3\_EID, [1440](#)  
 CFE\_ES\_RESTART\_APP\_ERR4\_EID, [1440](#)  
 CFE\_ES\_RESTART\_APP\_INF\_EID, [1441](#)  
 CFE\_ES\_SET\_MAX\_PR\_COUNT\_EID, [1441](#)  
 CFE\_ES\_START\_ERR\_EID, [1441](#)  
 CFE\_ES\_START\_EXC\_ACTION\_ERR\_EID, [1441](#)  
 CFE\_ES\_START\_INF\_EID, [1442](#)  
 CFE\_ES\_START\_INVALID\_ENTRY\_POINT\_ERR\_EID, [1442](#)  
 CFE\_ES\_START\_INVALID\_FILENAME\_ERR\_EID, [1442](#)  
 CFE\_ES\_START\_NULL\_APP\_NAME\_ERR\_EID, [1442](#)  
 CFE\_ES\_START\_PRIORITY\_ERR\_EID, [1443](#)  
 CFE\_ES\_STOP\_DBG\_EID, [1443](#)  
 CFE\_ES\_STOP\_ERR1\_EID, [1443](#)  
 CFE\_ES\_STOP\_ERR2\_EID, [1443](#)  
 CFE\_ES\_STOP\_ERR3\_EID, [1444](#)  
 CFE\_ES\_STOP\_INF\_EID, [1444](#)  
 CFE\_ES\_SYSLOG1\_INF\_EID, [1444](#)  
 CFE\_ES\_SYSLOG2\_EID, [1444](#)  
 CFE\_ES\_SYSLOG2\_ERR\_EID, [1445](#)  
 CFE\_ES\_SYSLOGMODE\_EID, [1445](#)  
 CFE\_ES\_TASKINFO\_EID, [1445](#)  
 CFE\_ES\_TASKINFO\_OSCREATE\_ERR\_EID, [1445](#)  
 CFE\_ES\_TASKINFO\_WR\_ERR\_EID, [1446](#)  
 CFE\_ES\_TASKINFO\_WRHDR\_ERR\_EID, [1446](#)  
 CFE\_ES\_TASKWR\_ERR\_EID, [1446](#)  
 CFE\_ES\_TLM\_POOL\_STATS\_INFO\_EID, [1446](#)  
 CFE\_ES\_VERSION\_INF\_EID, [1447](#)  
 CFE\_ES\_WRHDR\_ERR\_EID, [1447](#)  
 CFE\_ES\_WRITE\_CFE\_HDR\_ERR\_EID, [1447](#)  
**CFE\_ES\_ExceptionAction**  
 default\_cfe\_es\_extern\_typedefs.h, [1407](#)  
**CFE\_ES\_ExceptionAction\_Enum\_t**  
 default\_cfe\_es\_extern\_typedefs.h, [1404](#)  
**CFE\_ES\_ExceptionAction\_PROC\_RESTART**  
 default\_cfe\_es\_extern\_typedefs.h, [1407](#)  
**CFE\_ES\_ExceptionAction\_RESTART\_APP**  
 default\_cfe\_es\_extern\_typedefs.h, [1407](#)  
**CFE\_ES\_EXIT\_APP\_ERR\_EID**  
 cfe\_es\_eventids.h, [1431](#)  
**CFE\_ES\_EXIT\_APP\_INF\_EID**  
 cfe\_es\_eventids.h, [1431](#)  
**CFE\_ES\_ExitApp**  
 cFE Application Behavior APIs, [259](#)  
**CFE\_ES\_ExitChildTask**  
 cFE Child Task APIs, [272](#)  
**cfe\_es\_fcncodes.h**  
 CFE\_ES\_CLEAR\_ER\_LOG\_CC, [1448](#)  
 CFE\_ES\_CLEAR\_SYS\_LOG\_CC, [1449](#)  
 CFE\_ES\_DELETE\_CDS\_CC, [1450](#)  
 CFE\_ES\_DUMP\_CDS\_REGISTRY\_CC, [1450](#)  
 CFE\_ES\_NOOP\_CC, [1451](#)  
 CFE\_ES\_OVER\_WRITE\_SYS\_LOG\_CC, [1452](#)  
 CFE\_ES\_QUERY\_ALL\_CC, [1453](#)  
 CFE\_ES\_QUERY\_ALL\_TASKS\_CC, [1454](#)  
 CFE\_ES\_QUERY\_ONE\_CC, [1455](#)  
 CFE\_ES\_RELOAD\_APP\_CC, [1455](#)  
 CFE\_ES\_RESET\_COUNTERS\_CC, [1456](#)  
 CFE\_ES\_RESET\_PR\_COUNT\_CC, [1457](#)  
 CFE\_ES\_RESTART\_APP\_CC, [1458](#)  
 CFE\_ES\_RESTART\_CC, [1459](#)  
 CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC, [1460](#)  
 CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC, [1461](#)  
 CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC, [1461](#)  
 CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC, [1462](#)  
 CFE\_ES\_START\_APP\_CC, [1463](#)  
 CFE\_ES\_START\_PERF\_DATA\_CC, [1464](#)  
 CFE\_ES\_STOP\_APP\_CC, [1465](#)  
 CFE\_ES\_STOP\_PERF\_DATA\_CC, [1466](#)  
 CFE\_ES\_WRITE\_ER\_LOG\_CC, [1467](#)  
 CFE\_ES\_WRITE\_SYS\_LOG\_CC, [1468](#)  
**CFE\_ES\_FILE\_CLOSE\_ERR**  
 cFE Return Code Defines, [238](#)  
**CFE\_ES\_FILE\_IO\_ERR**

cFE Return Code Defines, [238](#)  
CFE\_ES\_FileNameCmd, [640](#)  
    CommandHeader, [640](#)  
    Payload, [640](#)  
CFE\_ES\_FileNameCmd\_Payload, [641](#)  
    FileName, [641](#)  
CFE\_ES\_FileNameCmd\_Payload\_t  
    default\_cfe\_es\_msgdefs.h, [1415](#)  
CFE\_ES\_FileNameCmd\_t  
    default\_cfe\_es\_msgstruct.h, [1420](#)  
CFE\_ES\_FILEWRITE\_ERR\_EID  
    cfe\_es\_eventids.h, [1431](#)  
CFE\_ES\_FunctionCode\_  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_CLEAR\_ER\_LOG  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_CLEAR\_SYS\_LOG  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_DELETE\_CDS  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_DUMP\_CDS\_REGISTRY  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_NOOP  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_OVER\_WRITE\_SYS\_LOG  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_QUERY\_ALL  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_QUERY\_ALL\_TASKS  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_QUERY\_ONE  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_RELOAD\_APP  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_RESET\_COUNTERS  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_RESET\_PR\_COUNT  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_RESTART  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_RESTART\_APP  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_SEND\_MEM\_POOL\_STATS  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_SET\_MAX\_PR\_COUNT  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_SET\_PERF\_FILTER\_MASK  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_SET\_PERF\_TRIGGER\_MASK  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_START\_APP  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_START\_PERF\_DATA  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_STOP\_APP  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_STOP\_PERF\_DATA  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_WRITE\_ER\_LOG  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_FunctionCode\_WRITE\_SYS\_LOG  
    default\_cfe\_es\_fcncode\_values.h, [1409](#)  
CFE\_ES\_GetAppID  
    cFE Information APIs, [262](#)  
CFE\_ES\_GetAppIDByName  
    cFE Information APIs, [263](#)  
CFE\_ES\_GetAppInfo  
    cFE Information APIs, [264](#)  
CFE\_ES\_GetAppName  
    cFE Information APIs, [264](#)  
CFE\_ES\_GetCDSBlockIDByName  
    cFE Critical Data Store APIs, [277](#)  
CFE\_ES\_GetCDSBlockName  
    cFE Critical Data Store APIs, [278](#)  
CFE\_ES\_GetGenCount  
    cFE Generic Counter APIs, [292](#)  
CFE\_ES\_GetGenCounterIDByName  
    cFE Generic Counter APIs, [292](#)  
CFE\_ES\_GetGenCounterName  
    cFE Generic Counter APIs, [293](#)  
CFE\_ES\_GetLibIDByName  
    cFE Information APIs, [265](#)  
CFE\_ES\_GetLibInfo  
    cFE Information APIs, [266](#)  
CFE\_ES\_GetLibName  
    cFE Information APIs, [267](#)  
CFE\_ES\_GetMemPoolStats  
    cFE Memory Manager APIs, [281](#)  
CFE\_ES\_GetModuleInfo  
    cFE Information APIs, [267](#)  
CFE\_ES\_GetPoolBuf  
    cFE Memory Manager APIs, [282](#)  
CFE\_ES\_GetPoolBufInfo  
    cFE Memory Manager APIs, [283](#)  
CFE\_ES\_GetResetType  
    cFE Information APIs, [268](#)  
CFE\_ES\_GetTaskID  
    cFE Information APIs, [269](#)  
CFE\_ES\_GetTaskIDByName  
    cFE Child Task APIs, [273](#)  
CFE\_ES\_GetTaskInfo  
    cFE Information APIs, [270](#)  
CFE\_ES\_GetTaskName  
    cFE Child Task APIs, [273](#)  
CFE\_ES\_HK\_TLM\_MID  
    default\_cfe\_es\_msgids.h, [1417](#)  
CFE\_ES\_HousekeepingTlm, [641](#)  
    Payload, [641](#)

TelemetryHeader, 641  
CFE\_ES\_HousekeepingTlm\_Payload, 642  
  BootSource, 644  
  CFECoreChecksum, 644  
  CFEMajorVersion, 644  
  CFEMinorVersion, 644  
  CFEMissionRevision, 644  
  CFERevision, 644  
  CommandCounter, 644  
  CommandErrorCounter, 645  
  ERLogEntries, 645  
  ERLogIndex, 645  
  HeapBlocksFree, 645  
  HeapBytesFree, 645  
  HeapMaxBlockSize, 645  
  MaxProcessorResets, 645  
  OSALMajorVersion, 646  
  OSALMinorVersion, 646  
  OSALMissionRevision, 646  
  OSALRevision, 646  
  PerfDataCount, 646  
  PerfDataEnd, 646  
  PerfDataStart, 646  
  PerfDataToWrite, 647  
  PerfFilterMask, 647  
  PerfMode, 647  
  PerfState, 647  
  PerfTriggerCount, 647  
  PerfTriggerMask, 647  
  ProcessorResets, 647  
  PSPMajorVersion, 648  
  PSPMinorVersion, 648  
  PSPMissionRevision, 648  
  PSPRevision, 648  
  RegisteredCoreApps, 648  
  RegisteredExternalApps, 648  
  RegisteredLibs, 648  
  RegisteredTasks, 649  
  ResetSubtype, 649  
  ResetType, 649  
  SysLogBytesUsed, 649  
  SysLogEntries, 649  
  SysLogMode, 649  
  SysLogSize, 649  
CFE\_ES\_HousekeepingTlm\_Payload\_t  
  default\_cfe\_es\_msgdefs.h, 1415  
CFE\_ES\_HousekeepingTlm\_t  
  default\_cfe\_es\_msgstruct.h, 1420  
CFE\_ES\_IncrementGenCounter  
  cFE Generic Counter APIs, 294  
CFE\_ES\_IncrementTaskCounter  
  cFE Application Behavior APIs, 259  
CFE\_ES\_INIT\_INF\_EID  
  cfe\_es\_eventids.h, 1432  
CFE\_ES\_INITSTATS\_INF\_EID  
  cfe\_es\_eventids.h, 1432  
cfe\_es\_interface\_cfg.h  
  CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN,  
    1470  
  CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH,  
    1470  
  CFE\_MISSION\_ES\_CRC\_16, 1470  
  CFE\_MISSION\_ES\_CRC\_32, 1470  
  CFE\_MISSION\_ES\_CRC\_8, 1471  
  CFE\_MISSION\_ES\_DEFAULT\_CRC, 1471  
  CFE\_MISSION\_ES\_MAX\_APPLICATIONS, 1471  
  CFE\_MISSION\_ES\_PERF\_MAX\_IDS, 1471  
  CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS, 1472  
  DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN,  
    1472  
  DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH,  
    1472  
  DEFAULT\_CFE\_MISSION\_ES\_CRC\_16, 1472  
  DEFAULT\_CFE\_MISSION\_ES\_CRC\_32, 1472  
  DEFAULT\_CFE\_MISSION\_ES\_CRC\_8, 1472  
  DEFAULT\_CFE\_MISSION\_ES\_DEFAULT\_CRC,  
    1472  
  DEFAULT\_CFE\_MISSION\_ES\_MAX\_APPLICATIONS,  
    1472  
  DEFAULT\_CFE\_MISSION\_ES\_PERF\_MAX\_IDS,  
    1473  
  DEFAULT\_CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS,  
    1473  
cfe\_es\_internal\_cfg.h  
  CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT, 1478  
  CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE, 1478  
  CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE,  
    1479  
  CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES,  
    1479  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01,  
    1479  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02,  
    1479  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03,  
    1479  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04,  
    1479  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05,  
    1480  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06,  
    1480  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07,  
    1480  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08,  
    1480  
  CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09,  
    1480

CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10,  
1480  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11,  
1480  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12,  
1480  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13,  
1480  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14,  
1480  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15,  
1481  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16,  
1481  
CFE\_PLATFORM\_ES\_CDS\_SIZE, 1481  
CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE,  
1481  
CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE, CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14,  
1481  
CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE,  
1482  
CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME, CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16,  
1482  
CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE, CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15,  
1482  
CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE, CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING,  
1483  
CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE,  
1483  
CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE,  
1483  
CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE,  
1484  
CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES, 1484  
CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE,  
1484  
CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS, 1485  
CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE, 1485  
CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS,  
1485  
CFE\_PLATFORM\_ES\_MAX\_LIBRARIES, 1485  
CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS,  
1486  
CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS,  
1486  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01,  
1486  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02,  
1487  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03,  
1487  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04,  
1487  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05,  
1487  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06,  
1487  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07,  
1487  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08,  
1487  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09,  
1488  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10,  
1488  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11,  
1488  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12,  
1488  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13,  
1488  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15,  
1488  
CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN,  
1488  
CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE,  
1489  
CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE, 1489  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY,  
1489  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY,  
1490  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE,  
1490  
CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE,  
1490  
CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS,  
1491  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL, 1491  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT,  
1491  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE,  
1491  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL,  
1492  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT,  
1492  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE,  
1492  
CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS,  
1492  
CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING,

1493  
CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS,  
1493  
CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED,  
1493  
CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE,  
1494  
CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY,  
1494  
CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE,  
1494  
CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC  
1495  
CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC,  
1495  
CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE, 1495  
CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE,  
1496  
CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE,  
1496  
DEFAULT\_CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT,  
1496  
DEFAULT\_CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE,  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE,  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES,  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01, 1500  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02, 1500  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03, 1500  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04, 1500  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05, 1500  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06, 1500  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07, 1500  
1497  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08, 1500  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09, 1500  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10, 1500  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11, 1501  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12, 1501  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13, 1501  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14,  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15,  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16,  
1498  
DEFAULT\_CFE\_PLATFORM\_ES\_CDS\_SIZE, 1498  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES,  
1499  
DEFAULT\_CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_LIBRARIES,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03,  
1500  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04,  
1501  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05,  
1501  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06,  
1501  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07,

1501  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08, DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED  
1501  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09, DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE,  
1501  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10, DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY,  
1501  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11, DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE,  
1501  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12, DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSE  
1501  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13, DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC,  
1501  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14, DEFAULT\_CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE,  
1502  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15, DEFAULT\_CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE,  
1502  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16, DEFAULT\_CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE,  
1502  
1504  
1504  
DEFAULT\_CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE, INVALID\_POOL\_HANDLE\_ERR\_EID  
1502  
cfe\_es\_eventids.h, 1432  
1502  
INVALID\_POOL\_HANDLE\_ERR\_EID  
cfe\_es\_eventids.h, 1432  
1502  
INVALID\_ES\_LIB\_ALREADY\_LOADED  
cFE Return Code Defines, 238  
1502  
INVALID\_ES\_LIBID\_BASE  
cFE Resource ID base values, 393  
1502  
INVALID\_ES\_LIBID\_C  
cfe\_es\_api\_typedefs.h, 1359  
1502  
INVALID\_ES\_LibId\_t  
default\_cfe\_es\_extern\_typedefs.h, 1404  
1502  
INVALID\_ES\_LibID\_TolIndex  
cFE Resource ID APIs, 253  
1503  
INVALID\_ES\_LIBID\_UNDEFINED  
cfe\_es\_api\_typedefs.h, 1359  
1503  
INVALID\_ES\_libraryEntryFuncPtr\_t  
cfe\_es\_api\_typedefs.h, 1360  
1503  
INVALID\_ES\_LogEntryType  
default\_cfe\_es\_extern\_typedefs.h, 1407  
1503  
INVALID\_ES\_LogEntryType\_APPLICATION  
default\_cfe\_es\_extern\_typedefs.h, 1407  
1503  
INVALID\_ES\_LogEntryType\_CORE  
default\_cfe\_es\_extern\_typedefs.h, 1407  
1503  
INVALID\_ES\_LogMode  
default\_cfe\_es\_extern\_typedefs.h, 1407  
1503  
INVALID\_ES\_LogMode\_DISCARD  
default\_cfe\_es\_extern\_typedefs.h, 1407  
1503  
INVALID\_ES\_LogMode\_Enum\_t  
default\_cfe\_es\_extern\_typedefs.h, 1405  
1503  
INVALID\_ES\_LogMode\_OVERWRITE  
default\_cfe\_es\_extern\_typedefs.h, 1407  
1503  
INVALID\_ES\_Main  
Main

cFE Entry/Exit APIs, 255

**CFE\_ES\_MEMADDRESS\_C**  
 default\_cfe\_es\_memaddress.h, 1411  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1270  
 example\_32bit\_cfe\_es\_memaddress.h, 1272  
 example\_64bit\_cfe\_es\_memaddress.h, 1274

**CFE\_ES\_MemAddress\_FromNative**  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1271

**CFE\_ES\_MemAddress\_t**, 650  
 bits, 650  
 default\_cfe\_es\_memaddress.h, 1412  
 example\_32bit\_cfe\_es\_memaddress.h, 1272  
 example\_64bit\_cfe\_es\_memaddress.h, 1274

**CFE\_ES\_MEMADDRESS\_TO\_PTR**  
 default\_cfe\_es\_memaddress.h, 1411  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1270  
 example\_32bit\_cfe\_es\_memaddress.h, 1272  
 example\_64bit\_cfe\_es\_memaddress.h, 1274

**CFE\_ES\_MemAddress\_ToNative**  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1271

**CFE\_ES\_MEMHANDLE\_C**  
 cfe\_es\_api\_typedefs.h, 1359

**CFE\_ES\_MemHandle\_t**  
 default\_cfe\_es\_extern\_typedefs.h, 1405

**CFE\_ES\_MEMHANDLE\_UNDEFINED**  
 cfe\_es\_api\_typedefs.h, 1359

**CFE\_ES\_MEMOFFSET\_C**  
 default\_cfe\_es\_memaddress.h, 1411  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1270  
 example\_32bit\_cfe\_es\_memaddress.h, 1272  
 example\_64bit\_cfe\_es\_memaddress.h, 1274

**CFE\_ES\_MemOffset\_FromNative**  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1271

**CFE\_ES\_MemOffset\_t**, 650  
 bits, 651  
 default\_cfe\_es\_memaddress.h, 1412  
 example\_32bit\_cfe\_es\_memaddress.h, 1273  
 example\_64bit\_cfe\_es\_memaddress.h, 1274

**CFE\_ES\_MEMOFFSET\_TO\_SIZE\_T**  
 default\_cfe\_es\_memaddress.h, 1411  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1270  
 example\_32bit\_cfe\_es\_memaddress.h, 1272  
 example\_64bit\_cfe\_es\_memaddress.h, 1274

**CFE\_ES\_MemOffset\_ToNative**  
 example\_2x32bit\_cfe\_es\_memaddress.h, 1271

**CFE\_ES\_MEMPOOLBUF\_C**  
 cfe\_es\_api\_typedefs.h, 1359

**CFE\_ES\_MemPoolBuf\_t**  
 cfe\_es\_api\_typedefs.h, 1361

**CFE\_ES\_MemPoolStats**, 651  
 BlockStats, 652  
 CheckErrCtr, 652  
 NumBlocksRequested, 652  
 NumFreeBytes, 652

PoolSize, 652

**CFE\_ES\_MemPoolStats\_t**  
 default\_cfe\_es\_extern\_typedefs.h, 1405

**CFE\_ES\_MEMSTATS\_TLM\_MID**  
 default\_cfe\_es\_msgids.h, 1417

**CFE\_ES\_MemStatsTlm**, 652  
 Payload, 653  
 TelemetryHeader, 653

**CFE\_ES\_MemStatsTlm\_t**  
 default\_cfe\_es\_msgstruct.h, 1420

**CFE\_ES\_MID\_ERR\_EID**  
 cfe\_es\_eventids.h, 1433

**CFE\_ES\_MUT\_SEM\_DELETE\_ERR**  
 cFE Return Code Defines, 238

**CFE\_ES\_NO\_MUTEX**  
 cfe\_es\_api\_typedefs.h, 1359

**CFE\_ES\_NO\_RESOURCE\_IDS\_AVAILABLE**  
 cFE Return Code Defines, 239

**CFE\_ES\_NOOP\_CC**  
 cfe\_es\_fcncodes.h, 1451

**CFE\_ES\_NOOP\_INF\_EID**  
 cfe\_es\_eventids.h, 1433

**CFE\_ES\_NoopCmd**, 653  
 CommandHeader, 653

**CFE\_ES\_NoopCmd\_t**  
 default\_cfe\_es\_msgstruct.h, 1420

**CFE\_ES\_NOT\_IMPLEMENTED**  
 cFE Return Code Defines, 239

**CFE\_ES\_ONE\_APP\_EID**  
 cfe\_es\_eventids.h, 1433

**CFE\_ES\_ONE\_APPID\_ERR\_EID**  
 cfe\_es\_eventids.h, 1433

**CFE\_ES\_ONE\_ERR\_EID**  
 cfe\_es\_eventids.h, 1434

**CFE\_ES\_OneAppTlm**, 653  
 Payload, 654  
 TelemetryHeader, 654

**CFE\_ES\_OneAppTlm\_Payload**, 654  
 ApplInfo, 654

**CFE\_ES\_OneAppTlm\_Payload\_t**  
 default\_cfe\_es\_msgdefs.h, 1415

**CFE\_ES\_OneAppTlm\_t**  
 default\_cfe\_es\_msgstruct.h, 1420

**CFE\_ES\_OPERATION\_TIMED\_OUT**  
 cFE Return Code Defines, 239

**CFE\_ES\_OSCREATE\_ERR\_EID**  
 cfe\_es\_eventids.h, 1434

**CFE\_ES\_OVER\_WRITE\_SYS\_LOG\_CC**  
 cfe\_es\_fcncodes.h, 1452

**CFE\_ES\_OverWriteSysLogCmd**, 654  
 CommandHeader, 655  
 Payload, 655

**CFE\_ES\_OverWriteSysLogCmd\_Payload**, 655  
 Mode, 655

CFE\_ES\_OverWriteSysLogCmd\_Payload\_t  
  default\_cfe\_es\_msgdefs.h, 1415

CFE\_ES\_OverWriteSysLogCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1420

CFE\_ES\_PCR\_ERR1\_EID  
  cfe\_es\_eventids.h, 1434

CFE\_ES\_PCR\_ERR2\_EID  
  cfe\_es\_eventids.h, 1434

CFE\_ES\_PERF\_DATAWRITTEN\_EID  
  cfe\_es\_eventids.h, 1435

CFE\_ES\_PERF\_FILTMSKCMD\_EID  
  cfe\_es\_eventids.h, 1435

CFE\_ES\_PERF\_FILTMSKERR\_EID  
  cfe\_es\_eventids.h, 1435

CFE\_ES\_PERF\_LOG\_ERR\_EID  
  cfe\_es\_eventids.h, 1435

CFE\_ES\_PERF\_STARTCMD\_EID  
  cfe\_es\_eventids.h, 1436

CFE\_ES\_PERF\_STARTCMD\_ERR\_EID  
  cfe\_es\_eventids.h, 1436

CFE\_ES\_PERF\_STARTCMD\_TRIG\_ERR\_EID  
  cfe\_es\_eventids.h, 1436

CFE\_ES\_PERF\_STOPCMD\_EID  
  cfe\_es\_eventids.h, 1436

CFE\_ES\_PERF\_STOPCMD\_ERR2\_EID  
  cfe\_es\_eventids.h, 1437

CFE\_ES\_PERF\_TRIGMSKCMD\_EID  
  cfe\_es\_eventids.h, 1437

CFE\_ES\_PERF\_TRIGMSKERR\_EID  
  cfe\_es\_eventids.h, 1437

CFE\_ES\_PerfLogAdd  
  cFE Performance Monitor APIs, 290

CFE\_ES\_PerfLogEntry  
  cFE Performance Monitor APIs, 289

CFE\_ES\_PerfLogExit  
  cFE Performance Monitor APIs, 289

CFE\_ES\_PerfMode  
  default\_cfe\_es\_msgdefs.h, 1416

CFE\_ES\_PerfMode\_Enum\_t  
  default\_cfe\_es\_msgdefs.h, 1415

CFE\_ES\_PerfTrigger\_CENTER  
  default\_cfe\_es\_msgdefs.h, 1416

CFE\_ES\_PerfTrigger\_END  
  default\_cfe\_es\_msgdefs.h, 1416

CFE\_ES\_PerfTrigger\_START  
  default\_cfe\_es\_msgdefs.h, 1416

CFE\_ES\_POOL\_BLOCK\_INVALID  
  cFE Return Code Defines, 239

CFE\_ES\_PoolAlign, 656

  LongDouble, 656

  LongInt, 656

  Ptr, 656

CFE\_ES\_PoolAlign\_t  
  cfe\_es\_api\_typedefs.h, 1361

CFE\_ES\_PoolCreate  
  cFE Memory Manager APIs, 283

CFE\_ES\_PoolCreateEx  
  cFE Memory Manager APIs, 284

CFE\_ES\_PoolCreateEx\_WithAlignment  
  cFE Memory Manager APIs, 285

CFE\_ES\_PoolCreateNoSem  
  cFE Memory Manager APIs, 286

CFE\_ES\_PoolDelete  
  cFE Memory Manager APIs, 287

CFE\_ES\_POOLID\_BASE  
  cFE Resource ID base values, 393

CFE\_ES\_PoolStatsTlm\_Payload, 656

  PoolHandle, 657

  PoolStats, 657

CFE\_ES\_PoolStatsTlm\_Payload\_t  
  default\_cfe\_es\_msgdefs.h, 1415

CFE\_ES\_ProcessAsyncEvent  
  cFE Miscellaneous APIs, 275

CFE\_ES\_PutPoolBuf  
  cFE Memory Manager APIs, 288

CFE\_ES\_QUERY\_ALL\_CC  
  cfe\_es\_fcncodes.h, 1453

CFE\_ES\_QUERY\_ALL\_TASKS\_CC  
  cfe\_es\_fcncodes.h, 1454

CFE\_ES\_QUERY\_ONE\_CC  
  cfe\_es\_fcncodes.h, 1455

CFE\_ES\_QueryAllCmd, 657

  CommandHeader, 657

  Payload, 657

CFE\_ES\_QueryAllCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1420

CFE\_ES\_QueryAllTasksCmd, 658

  CommandHeader, 658

  Payload, 658

CFE\_ES\_QueryAllTasksCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1420

CFE\_ES\_QueryOneCmd, 658

  CommandHeader, 658

  Payload, 658

CFE\_ES\_QueryOneCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1421

CFE\_ES\_QUEUE\_DELETE\_ERR  
  cFE Return Code Defines, 239

CFE\_ES\_RegisterCDS  
  cFE Critical Data Store APIs, 279

CFE\_ES\_RegisterGenCounter  
  cFE Generic Counter APIs, 294

CFE\_ES\_RELOAD\_APP\_CC  
  cfe\_es\_fcncodes.h, 1455

CFE\_ES\_RELOAD\_APP\_DBG\_EID  
  cfe\_es\_eventids.h, 1437

CFE\_ES\_RELOAD\_APP\_ERR1\_EID  
  cfe\_es\_eventids.h, 1438

CFE\_ES\_RELOAD\_APP\_ERR2\_EID  
     cfe\_es\_eventids.h, 1438

CFE\_ES\_RELOAD\_APP\_ERR3\_EID  
     cfe\_es\_eventids.h, 1438

CFE\_ES\_RELOAD\_APP\_ERR4\_EID  
     cfe\_es\_eventids.h, 1438

CFE\_ES\_RELOAD\_APP\_INF\_EID  
     cfe\_es\_eventids.h, 1439

CFE\_ES\_ReloadApp  
     cFE Application Control APIs, 257

CFE\_ES\_ReloadAppCmd, 659  
     CommandHeader, 659  
     Payload, 659

CFE\_ES\_ReloadAppCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1421

CFE\_ES\_RESET\_COUNTERS\_CC  
     cfe\_es\_fcncodes.h, 1456

CFE\_ES\_RESET\_INF\_EID  
     cfe\_es\_eventids.h, 1439

CFE\_ES\_RESET\_PR\_COUNT\_CC  
     cfe\_es\_fcncodes.h, 1457

CFE\_ES\_RESET\_PR\_COUNT\_EID  
     cfe\_es\_eventids.h, 1439

CFE\_ES\_ResetCFE  
     cFE Entry/Exit APIs, 255

CFE\_ES\_ResetCountersCmd, 659  
     CommandHeader, 660

CFE\_ES\_ResetCountersCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1421

CFE\_ES\_ResetPRCountCmd, 660  
     CommandHeader, 660

CFE\_ES\_ResetPRCountCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1421

CFE\_ES\_RESTART\_APP\_CC  
     cfe\_es\_fcncodes.h, 1458

CFE\_ES\_RESTART\_APP\_DBG\_EID  
     cfe\_es\_eventids.h, 1439

CFE\_ES\_RESTART\_APP\_ERR1\_EID  
     cfe\_es\_eventids.h, 1440

CFE\_ES\_RESTART\_APP\_ERR2\_EID  
     cfe\_es\_eventids.h, 1440

CFE\_ES\_RESTART\_APP\_ERR3\_EID  
     cfe\_es\_eventids.h, 1440

CFE\_ES\_RESTART\_APP\_ERR4\_EID  
     cfe\_es\_eventids.h, 1440

CFE\_ES\_RESTART\_APP\_INF\_EID  
     cfe\_es\_eventids.h, 1441

CFE\_ES\_RESTART\_CC  
     cfe\_es\_fcncodes.h, 1459

CFE\_ES\_RestartApp  
     cFE Application Control APIs, 258

CFE\_ES\_RestartAppCmd, 660  
     CommandHeader, 660  
     Payload, 661

CFE\_ES\_RestartAppCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1421

CFE\_ES\_RestartCmd, 661  
     CommandHeader, 661  
     Payload, 661

CFE\_ES\_RestartCmd\_Payload, 661  
     RestartType, 662

CFE\_ES\_RestartCmd\_Payload\_t  
     default\_cfe\_es\_msgdefs.h, 1415

CFE\_ES\_RestartCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1421

CFE\_ES\_RestoreFromCDS  
     cFE Critical Data Store APIs, 280

CFE\_ES\_RST\_ACCESS\_ERR  
     cFE Return Code Defines, 239

CFE\_ES\_RunLoop  
     cFE Application Behavior APIs, 260

CFE\_ES\_RunStatus  
     default\_cfe\_es\_extern\_typedefs.h, 1407

CFE\_ES\_RunStatus\_APP\_ERROR  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_APP\_EXIT  
     default\_cfe\_es\_extern\_typedefs.h, 1407

CFE\_ES\_RunStatus\_APP\_RUN  
     default\_cfe\_es\_extern\_typedefs.h, 1407

CFE\_ES\_RunStatus\_CORE\_APP\_INIT\_ERROR  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_CORE\_APP\_RUNTIME\_ERROR  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_Enum\_t  
     default\_cfe\_es\_extern\_typedefs.h, 1405

CFE\_ES\_RunStatus\_MAX  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_SYS\_DELETE  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_SYS\_EXCEPTION  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_SYS\_RELOAD  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_SYS\_RESTART  
     default\_cfe\_es\_extern\_typedefs.h, 1408

CFE\_ES\_RunStatus\_UNDEFINED  
     default\_cfe\_es\_extern\_typedefs.h, 1407

CFE\_ES\_SEND\_HK\_MID  
     default\_cfe\_es\_msgids.h, 1417

CFE\_ES\_SEND\_MEM\_POOL\_STATS\_CC  
     cfe\_es\_fcncodes.h, 1460

CFE\_ES\_SendHkCmd, 662  
     CommandHeader, 662

CFE\_ES\_SendHkCmd\_t  
     default\_cfe\_es\_msgstruct.h, 1421

CFE\_ES\_SendMemPoolStatsCmd, 662  
     CommandHeader, 663  
     Payload, 663

CFE\_ES\_SendMemPoolStatsCmd\_Payload, 663  
Application, 663  
PoolHandle, 663  
CFE\_ES\_SendMemPoolStatsCmd\_Payload\_t  
default\_cfe\_es\_msgdefs.h, 1415  
CFE\_ES\_SendMemPoolStatsCmd\_t  
default\_cfe\_es\_msgstruct.h, 1421  
CFE\_ES\_SET\_MAX\_PR\_COUNT\_CC  
cfe\_es\_fcncodes.h, 1461  
CFE\_ES\_SET\_MAX\_PR\_COUNT\_EID  
cfe\_es\_eventids.h, 1441  
CFE\_ES\_SET\_PERF\_FILTER\_MASK\_CC  
cfe\_es\_fcncodes.h, 1461  
CFE\_ES\_SET\_PERF\_TRIGGER\_MASK\_CC  
cfe\_es\_fcncodes.h, 1462  
CFE\_ES\_SetGenCount  
cFE Generic Counter APIs, 295  
CFE\_ES\_SetMaxPRCountCmd, 664  
CommandHeader, 664  
Payload, 664  
CFE\_ES\_SetMaxPRCountCmd\_Payload, 664  
MaxPRCount, 664  
CFE\_ES\_SetMaxPRCountCmd\_Payload\_t  
default\_cfe\_es\_msgdefs.h, 1415  
CFE\_ES\_SetMaxPRCountCmd\_t  
default\_cfe\_es\_msgstruct.h, 1421  
CFE\_ES\_SetPerfFilterMaskCmd, 665  
CommandHeader, 665  
Payload, 665  
CFE\_ES\_SetPerfFilterMaskCmd\_Payload, 665  
FilterMask, 666  
FilterMaskNum, 666  
CFE\_ES\_SetPerfFilterMaskCmd\_Payload\_t  
default\_cfe\_es\_msgdefs.h, 1415  
CFE\_ES\_SetPerfFilterMaskCmd\_t  
default\_cfe\_es\_msgstruct.h, 1421  
CFE\_ES\_SetPerfTriggerMaskCmd, 666  
CommandHeader, 666  
Payload, 666  
CFE\_ES\_SetPerfTriggerMaskCmd\_t  
default\_cfe\_es\_msgstruct.h, 1421  
CFE\_ES\_SetPerfTrigMaskCmd\_Payload, 667  
TriggerMask, 667  
TriggerMaskNum, 667  
CFE\_ES\_SetPerfTrigMaskCmd\_Payload\_t  
default\_cfe\_es\_msgdefs.h, 1416  
CFE\_ES\_StackPointer\_t  
cfe\_es\_api\_typedefs.h, 1361  
CFE\_ES\_START\_APP\_CC  
cfe\_es\_fcncodes.h, 1463  
CFE\_ES\_START\_ERR\_EID  
cfe\_es\_eventids.h, 1441  
CFE\_ES\_START\_EXC\_ACTION\_ERR\_EID  
cfe\_es\_eventids.h, 1441  
CFE\_ES\_START\_INF\_EID  
cfe\_es\_eventids.h, 1442  
CFE\_ES\_START\_INVALID\_ENTRY\_POINT\_ERR\_EID  
cfe\_es\_eventids.h, 1442  
CFE\_ES\_START\_INVALID\_FILENAME\_ERR\_EID  
cfe\_es\_eventids.h, 1442  
CFE\_ES\_START\_NULL\_APP\_NAME\_ERR\_EID  
cfe\_es\_eventids.h, 1442  
CFE\_ES\_START\_PERF\_DATA\_CC  
cfe\_es\_fcncodes.h, 1464  
CFE\_ES\_START\_PRIORITY\_ERR\_EID  
cfe\_es\_eventids.h, 1443  
CFE\_ES\_StartApp, 667  
CommandHeader, 667  
Payload, 668  
CFE\_ES\_StartAppCmd\_Payload, 668  
AppEntryPoint, 668  
AppFileName, 668  
Application, 668  
ExceptionAction, 669  
Priority, 669  
StackSize, 669  
CFE\_ES\_StartAppCmd\_Payload\_t  
default\_cfe\_es\_msgdefs.h, 1416  
CFE\_ES\_StartAppCmd\_t  
default\_cfe\_es\_msgstruct.h, 1421  
CFE\_ES\_StartPerfCmd\_Payload, 669  
TriggerMode, 669  
CFE\_ES\_StartPerfCmd\_Payload\_t  
default\_cfe\_es\_msgdefs.h, 1416  
CFE\_ES\_StartPerfDataCmd, 669  
CommandHeader, 670  
Payload, 670  
CFE\_ES\_StartPerfDataCmd\_t  
default\_cfe\_es\_msgstruct.h, 1421  
CFE\_ES\_STATIC\_POOL\_TYPE  
cfe\_es\_api\_typedefs.h, 1359  
CFE\_ES\_StatusToString  
cfe\_error.h, 1353  
CFE\_ES\_STOP\_APP\_CC  
cfe\_es\_fcncodes.h, 1465  
CFE\_ES\_STOP\_DBG\_EID  
cfe\_es\_eventids.h, 1443  
CFE\_ES\_STOP\_ERR1\_EID  
cfe\_es\_eventids.h, 1443  
CFE\_ES\_STOP\_ERR2\_EID  
cfe\_es\_eventids.h, 1443  
CFE\_ES\_STOP\_ERR3\_EID  
cfe\_es\_eventids.h, 1444  
CFE\_ES\_STOP\_INF\_EID  
cfe\_es\_eventids.h, 1444  
CFE\_ES\_STOP\_PERF\_DATA\_CC  
cfe\_es\_fcncodes.h, 1466  
CFE\_ES\_StopAppCmd, 670

CommandHeader, 670  
 Payload, 670  
**CFE\_ES\_StopAppCmd\_t**  
 default\_cfe\_es\_msgstruct.h, 1421  
**CFE\_ES\_StopPerfCmd\_Payload**, 671  
 DataFileName, 671  
**CFE\_ES\_StopPerfCmd\_Payload\_t**  
 default\_cfe\_es\_msgdefs.h, 1416  
**CFE\_ES\_StopPerfDataCmd**, 671  
 CommandHeader, 671  
 Payload, 672  
**CFE\_ES\_StopPerfDataCmd\_t**  
 default\_cfe\_es\_msgstruct.h, 1422  
**CFE\_ES\_SYSLOG1\_INF\_EID**  
 cfe\_es\_eventids.h, 1444  
**CFE\_ES\_SYSLOG2\_EID**  
 cfe\_es\_eventids.h, 1444  
**CFE\_ES\_SYSLOG2\_ERR\_EID**  
 cfe\_es\_eventids.h, 1445  
**CFE\_ES\_SYSLOGMODE\_EID**  
 cfe\_es\_eventids.h, 1445  
**CFE\_ES\_SystemState**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_APPS\_INIT**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_CORE\_READY**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_CORE\_STARTUP**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_EARLY\_INIT**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_Enum\_t**  
 default\_cfe\_es\_extern\_typedefs.h, 1405  
**CFE\_ES\_SystemState\_MAX**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_OPERATIONAL**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_SHUTDOWN**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_SystemState\_UNDEFINED**  
 default\_cfe\_es\_extern\_typedefs.h, 1408  
**CFE\_ES\_TASK\_DELETE\_ERR**  
 cFE Return Code Defines, 239  
**CFE\_ES\_TASK\_STACK\_ALLOCATE**  
 cfe\_es\_api\_typedefs.h, 1360  
**CFE\_ES\_TaskEntryFuncPtr\_t**  
 cfe\_es\_api\_typedefs.h, 1361  
**CFE\_ES\_TASKID\_BASE**  
 cFE Resource ID base values, 393  
**CFE\_ES\_TASKID\_C**  
 cfe\_es\_api\_typedefs.h, 1360  
**CFE\_ES\_TaskId\_t**  
 default\_cfe\_es\_extern\_typedefs.h, 1405  
**CFE\_ES\_TaskID\_ToIndex**  
 cFE Resource ID APIs, 254  
**CFE\_ES\_TASKID\_UNDEFINED**  
 cfe\_es\_api\_typedefs.h, 1360  
**CFE\_ES\_TaskInfo**, 672  
 ApId, 672  
 AppName, 672  
 ExecutionCounter, 673  
 Priority, 673  
 Spare, 673  
 StackSize, 673  
 TaskId, 673  
 TaskName, 673  
**CFE\_ES\_TASKINFO\_EID**  
 cfe\_es\_eventids.h, 1445  
**CFE\_ES\_TASKINFO\_OSCCREATE\_ERR\_EID**  
 cfe\_es\_eventids.h, 1445  
**CFE\_ES\_TaskInfo\_t**  
 default\_cfe\_es\_extern\_typedefs.h, 1406  
**CFE\_ES\_TASKINFO\_WR\_ERR\_EID**  
 cfe\_es\_eventids.h, 1446  
**CFE\_ES\_TASKINFO\_WRHDR\_ERR\_EID**  
 cfe\_es\_eventids.h, 1446  
**CFE\_ES\_TaskPriority\_Atom\_t**  
 default\_cfe\_es\_extern\_typedefs.h, 1406  
**CFE\_ES\_TASKWR\_ERR\_EID**  
 cfe\_es\_eventids.h, 1446  
**CFE\_ES\_TIMER\_DELETE\_ERR**  
 cFE Return Code Defines, 239  
**CFE\_ES\_TLM\_POOL\_STATS\_INFO\_EID**  
 cfe\_es\_eventids.h, 1446  
**cfe\_es\_topicids.h**  
 CFE\_MISSION\_ES\_APP\_TLM\_TOPICID, 1505  
 CFE\_MISSION\_ES\_CMD\_TOPICID, 1505  
 CFE\_MISSION\_ES\_HK\_TLM\_TOPICID, 1505  
 CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID,  
 1506  
 CFE\_MISSION\_ES\_SEND\_HK\_TOPICID, 1506  
 DEFAULT\_CFE\_MISSION\_ES\_APP\_TLM\_TOPICID,  
 1506  
 DEFAULT\_CFE\_MISSION\_ES\_CMD\_TOPICID,  
 1506  
 DEFAULT\_CFE\_MISSION\_ES\_HK\_TLM\_TOPICID,  
 1506  
 DEFAULT\_CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID,  
 1506  
 DEFAULT\_CFE\_MISSION\_ES\_SEND\_HK\_TOPICID,  
 1506  
**CFE\_ES\_USE\_MUTEX**  
 cfe\_es\_api\_typedefs.h, 1360  
**CFE\_ES\_VERSION\_INF\_EID**  
 cfe\_es\_eventids.h, 1447  
**CFE\_ES\_WaitForStartupSync**  
 cFE Application Behavior APIs, 260  
**CFE\_ES\_WaitForSystemState**

cFE Application Behavior APIs, 261  
CFE\_ES\_WRHDR\_ERR\_EID  
  cfe\_es\_eventids.h, 1447  
CFE\_ES\_WRITE\_CFE\_HDR\_ERR\_EID  
  cfe\_es\_eventids.h, 1447  
CFE\_ES\_WRITE\_ER\_LOG\_CC  
  cfe\_es\_fcncodes.h, 1467  
CFE\_ES\_WRITE\_SYS\_LOG\_CC  
  cfe\_es\_fcncodes.h, 1468  
CFE\_ES\_WriteERLogCmd, 673  
  CommandHeader, 674  
  Payload, 674  
CFE\_ES\_WriteERLogCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1422  
CFE\_ES\_WriteSysLogCmd, 674  
  CommandHeader, 674  
  Payload, 674  
CFE\_ES\_WriteSysLogCmd\_t  
  default\_cfe\_es\_msgstruct.h, 1422  
CFE\_ES\_WriteToSysLog  
  cFE Miscellaneous APIs, 276  
CFE\_EVENTS\_SERVICE  
  cfe\_error.h, 1351  
cfe\_evs.h  
  CFE\_EVS\_Send, 1363  
  CFE\_EVS\_SendCrit, 1363  
  CFE\_EVS\_SendDbg, 1363  
  CFE\_EVS\_SendErr, 1363  
  CFE\_EVS\_SendInfo, 1363  
CFE\_EVS\_ADD\_EVENT\_FILTER\_CC  
  cfe\_evs\_fcncodes.h, 1533  
CFE\_EVS\_AddEventFilterCmd, 674  
  CommandHeader, 675  
  Payload, 675  
CFE\_EVS\_AddEventFilterCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1518  
CFE\_EVS\_ADDFILTER\_EID  
  cfe\_evs\_eventids.h, 1522  
cfe\_evs\_api\_typedefs.h  
  CFE\_EVS\_BinFilter\_t, 1365  
  CFE\_EVS\_EVERY\_FOURTH\_ONE, 1364  
  CFE\_EVS\_EVERY\_OTHER\_ONE, 1364  
  CFE\_EVS\_EVERY\_OTHER\_TWO, 1365  
  CFE\_EVS\_FIRST\_16\_STOP, 1365  
  CFE\_EVS\_FIRST\_32\_STOP, 1365  
  CFE\_EVS\_FIRST\_4\_STOP, 1365  
  CFE\_EVS\_FIRST\_64\_STOP, 1365  
  CFE\_EVS\_FIRST\_8\_STOP, 1365  
  CFE\_EVS\_FIRST\_ONE\_STOP, 1365  
  CFE\_EVS\_FIRST\_TWO\_STOP, 1365  
  CFE\_EVS\_NO\_FILTER, 1365  
CFE\_EVS\_APP\_FILTER\_OVERLOAD  
  cFE Return Code Defines, 240  
CFE\_EVS\_APP\_ILLEGAL\_APP\_ID  
  cFE Return Code Defines, 240  
CFE\_EVS\_APP\_NOT\_REGISTERED  
  cFE Return Code Defines, 240  
CFE\_EVS\_APP\_SQUELCHED  
  cFE Return Code Defines, 240  
CFE\_EVS\_AppDataCmd\_Payload, 675  
  AppDataFilename, 675  
CFE\_EVS\_AppDataCmd\_Payload\_t  
  default\_cfe\_evs\_msgdefs.h, 1514  
CFE\_EVS\_AppNameBitMaskCmd\_Payload, 676  
  AppName, 676  
  BitMask, 676  
  Spare, 676  
CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t  
  default\_cfe\_evs\_msgdefs.h, 1514  
CFE\_EVS\_AppNameCmd\_Payload, 676  
  AppName, 677  
CFE\_EVS\_AppNameCmd\_Payload\_t  
  default\_cfe\_evs\_msgdefs.h, 1514  
CFE\_EVS\_AppNameEventIDCmd\_Payload, 677  
  AppName, 677  
  EventID, 677  
CFE\_EVS\_AppNameEventIDCmd\_Payload\_t  
  default\_cfe\_evs\_msgdefs.h, 1515  
CFE\_EVS\_AppNameEventIDMaskCmd\_Payload, 677  
  AppName, 678  
  EventID, 678  
  Mask, 678  
CFE\_EVS\_AppNameEventIDMaskCmd\_Payload\_t  
  default\_cfe\_evs\_msgdefs.h, 1515  
CFE\_EVS\_AppTlmData, 678  
  AppEnableStatus, 679  
  AppID, 679  
  AppMessageSentCounter, 679  
  AppMessageSquelchedCounter, 679  
CFE\_EVS\_AppTlmData\_t  
  default\_cfe\_evs\_msgdefs.h, 1515  
CFE\_EVS\_BinFilter, 679  
  EventID, 680  
  Mask, 680  
CFE\_EVS\_BinFilter\_t  
  cfe\_evs\_api\_typedefs.h, 1365  
CFE\_EVS\_BitMaskCmd\_Payload, 680  
  BitMask, 680  
  Spare, 680  
CFE\_EVS\_BitMaskCmd\_Payload\_t  
  default\_cfe\_evs\_msgdefs.h, 1515  
CFE\_EVS\_CCVAL  
  default\_cfe\_evs\_fcncode\_values.h, 1510  
CFE\_EVS\_CLEAR\_LOG\_CC  
  cfe\_evs\_fcncodes.h, 1534  
CFE\_EVS\_ClearLogCmd, 681  
  CommandHeader, 681  
CFE\_EVS\_ClearLogCmd\_t

default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_CMD\_MID  
    default\_cfe\_evs\_msgids.h, 1516  
CFE\_EVS\_CRITICAL\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1514  
CFE\_EVS\_DEBUG\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1514  
CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC  
    cfe\_evs\_fcncodes.h, 1535  
CFE\_EVS\_DeleteEventFilterCmd, 681  
    CommandHeader, 681  
    Payload, 681  
CFE\_EVS\_DeleteEventFilterCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_DELFILTER\_EID  
    cfe\_evs\_eventids.h, 1523  
CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC  
    cfe\_evs\_fcncodes.h, 1536  
CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC  
    cfe\_evs\_fcncodes.h, 1537  
CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC  
    cfe\_evs\_fcncodes.h, 1538  
CFE\_EVS\_DISABLE\_PORTS\_CC  
    cfe\_evs\_fcncodes.h, 1539  
CFE\_EVS\_DisableAppEventsCmd, 682  
    CommandHeader, 682  
    Payload, 682  
CFE\_EVS\_DisableAppEventsCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_DisableEventTypeCmd, 682  
    CommandHeader, 682  
    Payload, 682  
CFE\_EVS\_DisableEventTypeCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_DisableEventTypeCmd, 683  
    CommandHeader, 683  
    Payload, 683  
CFE\_EVS\_DisableEventTypeCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_DisablePortsCmd, 683  
    CommandHeader, 684  
    Payload, 684  
CFE\_EVS\_DisablePortsCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_DISAPPENTTYPE\_EID  
    cfe\_evs\_eventids.h, 1523  
CFE\_EVS\_DISAPPEVT\_EID  
    cfe\_evs\_eventids.h, 1523  
CFE\_EVS\_DISEVTTYPE\_EID  
    cfe\_evs\_eventids.h, 1523  
CFE\_EVS\_DISPRT\_EID  
    cfe\_evs\_eventids.h, 1524  
CFE\_EVS\_ENAAPPEVT\_EID  
    cfe\_evs\_eventids.h, 1524  
CFE\_EVS\_ENAAPPEVTTYPE\_EID  
    cfe\_evs\_eventids.h, 1524  
CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC  
    cfe\_evs\_fcncodes.h, 1540  
CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC  
    cfe\_evs\_fcncodes.h, 1540  
CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC  
    cfe\_evs\_fcncodes.h, 1541  
CFE\_EVS\_ENABLE\_PORTS\_CC  
    cfe\_evs\_fcncodes.h, 1542  
CFE\_EVS\_EnableAppEventsCmd, 684  
    CommandHeader, 684  
    Payload, 684  
CFE\_EVS\_EnableAppEventsCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_EnableEventTypeCmd, 684  
    CommandHeader, 685  
    Payload, 685  
CFE\_EVS\_EnableEventTypeCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_EnablePortsCmd, 685  
    CommandHeader, 685  
    Payload, 685  
CFE\_EVS\_EnablePortsCmd\_t  
    default\_cfe\_evs\_msgstruct.h, 1519  
CFE\_EVS\_ENAEVTTYPE\_EID  
    cfe\_evs\_eventids.h, 1524  
CFE\_EVS\_ENAPORT\_EID  
    cfe\_evs\_eventids.h, 1525  
CFE\_EVS\_ERR\_APPNOREGS\_EID  
    cfe\_evs\_eventids.h, 1525  
CFE\_EVS\_ERR\_CC\_EID  
    cfe\_evs\_eventids.h, 1525  
CFE\_EVS\_ERR\_CRDATFILE\_EID  
    cfe\_evs\_eventids.h, 1525  
CFE\_EVS\_ERR\_CRLOGFILE\_EID  
    cfe\_evs\_eventids.h, 1526  
CFE\_EVS\_ERR\_EVTIDNOREGS\_EID  
    cfe\_evs\_eventids.h, 1526  
CFE\_EVS\_ERR\_ILLAPPIDRANGE\_EID  
    cfe\_evs\_eventids.h, 1526  
CFE\_EVS\_ERR\_ILLEGALFMTMOD\_EID  
    cfe\_evs\_eventids.h, 1526  
CFE\_EVS\_ERR\_INVALID\_BITMASK\_EID  
    cfe\_evs\_eventids.h, 1527  
CFE\_EVS\_ERR\_LOGMODE\_EID  
    cfe\_evs\_eventids.h, 1527  
CFE\_EVS\_ERR\_MAXREGSFILTER\_EID  
    cfe\_evs\_eventids.h, 1527

CFE\_EVS\_ERR\_MSGID\_EID  
    cfe\_evs\_eventids.h, 1527

CFE\_EVS\_ERR\_NOAPPIDFOUND\_EID  
    cfe\_evs\_eventids.h, 1528

CFE\_EVS\_ERR\_UNREGISTERED\_EVS\_APP  
    cfe\_evs\_eventids.h, 1528

CFE\_EVS\_ERR\_WRDATFILE\_EID  
    cfe\_evs\_eventids.h, 1528

CFE\_EVS\_ERR\_WRLGOFLE\_EID  
    cfe\_evs\_eventids.h, 1528

CFE\_EVS\_ERROR\_BIT  
    default\_cfe\_evs\_msgdefs.h, 1514

CFE\_EVS\_EventFilter  
    default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventFilter\_BINARY  
    default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventFilter\_Enum\_t  
    default\_cfe\_evs\_extern\_typedefs.h, 1507

cfe\_evs\_eventids.h

- CFE\_EVS\_ADDFILTER\_EID, 1522
- CFE\_EVS\_DELFILTER\_EID, 1523
- CFE\_EVS\_DISAPPENTTYPE\_EID, 1523
- CFE\_EVS\_DISAPPEVT\_EID, 1523
- CFE\_EVS\_DISEVTTYPE\_EID, 1523
- CFE\_EVS\_DISPRT\_EID, 1524
- CFE\_EVS\_ENAAPPEVT\_EID, 1524
- CFE\_EVS\_ENAAPPEVTTYPE\_EID, 1524
- CFE\_EVS\_ENAEVTTYPE\_EID, 1524
- CFE\_EVS\_ENAPORT\_EID, 1525
- CFE\_EVS\_ERR\_APPNOREGS\_EID, 1525
- CFE\_EVS\_ERR\_CC\_EID, 1525
- CFE\_EVS\_ERR\_CRDATFILE\_EID, 1525
- CFE\_EVS\_ERR\_CRLOGFILE\_EID, 1526
- CFE\_EVS\_ERR\_EVTIDNOREGS\_EID, 1526
- CFE\_EVS\_ERR\_ILLAPPIDRANGE\_EID, 1526
- CFE\_EVS\_ERR\_ILLEGALFMTMOD\_EID, 1526
- CFE\_EVS\_ERR\_INVALID\_BITMASK\_EID, 1527
- CFE\_EVS\_ERR\_LOGMODE\_EID, 1527
- CFE\_EVS\_ERR\_MAXREGSFILTER\_EID, 1527
- CFE\_EVS\_ERR\_MSGID\_EID, 1527
- CFE\_EVS\_ERR\_NOAPPIDFOUND\_EID, 1528
- CFE\_EVS\_ERR\_UNREGISTERED\_EVS\_APP, 1528
- CFE\_EVS\_ERR\_WRDATFILE\_EID, 1528
- CFE\_EVS\_ERR\_WRLGOFLE\_EID, 1528
- CFE\_EVS\_EVT\_FILTERED\_EID, 1529
- CFE\_EVS\_FILTER\_MAX\_EID, 1529
- CFE\_EVS\_LEN\_ERR\_EID, 1529
- CFE\_EVS\_LOGMODE\_EID, 1529
- CFE\_EVS\_NOOP\_EID, 1530
- CFE\_EVS\_RSTALLFILTER\_EID, 1530
- CFE\_EVS\_RSTCNT\_EID, 1530
- CFE\_EVS\_RSTEVCNT\_EID, 1530
- CFE\_EVS\_RSTFILTER\_EID, 1531

CFE\_EVS\_SETEVTFMTMOD\_EID, 1531

CFE\_EVS\_SETFILTERMSK\_EID, 1531

CFE\_EVS\_SQUELCHED\_ERR\_EID, 1531

CFE\_EVS\_STARTUP\_EID, 1532

CFE\_EVS\_WRDAT\_EID, 1532

CFE\_EVS\_WRITE\_HEADER\_ERR\_EID, 1532

CFE\_EVS\_WRLG\_EID, 1532

CFE\_EVS\_EventOutput

- default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventOutput\_Enum\_t

- default\_cfe\_evs\_extern\_typedefs.h, 1507

CFE\_EVS\_EventOutput\_PORT1

- default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventOutput\_PORT2

- default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventOutput\_PORT3

- default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventOutput\_PORT4

- default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventType

- default\_cfe\_evs\_extern\_typedefs.h, 1508

CFE\_EVS\_EventType\_CRITICAL

- default\_cfe\_evs\_extern\_typedefs.h, 1509

CFE\_EVS\_EventType\_DEBUG

- default\_cfe\_evs\_extern\_typedefs.h, 1509

CFE\_EVS\_EventType\_Enum\_t

- default\_cfe\_evs\_extern\_typedefs.h, 1507

CFE\_EVS\_EventType\_ERROR

- default\_cfe\_evs\_extern\_typedefs.h, 1509

CFE\_EVS\_EventType\_INFORMATION

- default\_cfe\_evs\_extern\_typedefs.h, 1509

CFE\_EVS\_EVERY\_FOURTH\_ONE

- cfe\_evs\_api\_typedefs.h, 1364

CFE\_EVS\_EVERY\_OTHER\_ONE

- cfe\_evs\_api\_typedefs.h, 1364

CFE\_EVS\_EVERY\_OTHER\_TWO

- cfe\_evs\_api\_typedefs.h, 1365

CFE\_EVS\_EVT\_FILTERED\_EID

- cfe\_evs\_eventids.h, 1529

CFE\_EVS\_EVT\_NOT\_REGISTERED

- CFE Return Code Defines, 240

cfe\_evs\_fcncodes.h

- CFE\_EVS\_ADD\_EVENT\_FILTER\_CC, 1533
- CFE\_EVS\_CLEAR\_LOG\_CC, 1534
- CFE\_EVS\_DELETE\_EVENT\_FILTER\_CC, 1535
- CFE\_EVS\_DISABLE\_APP\_EVENT\_TYPE\_CC, 1536
- CFE\_EVS\_DISABLE\_APP\_EVENTS\_CC, 1537
- CFE\_EVS\_DISABLE\_EVENT\_TYPE\_CC, 1538
- CFE\_EVS\_DISABLE\_PORTS\_CC, 1539
- CFE\_EVS\_ENABLE\_APP\_EVENT\_TYPE\_CC, 1540
- CFE\_EVS\_ENABLE\_APP\_EVENTS\_CC, 1540
- CFE\_EVS\_ENABLE\_EVENT\_TYPE\_CC, 1541
- CFE\_EVS\_ENABLE\_PORTS\_CC, 1542

CFE\_EVS\_NOOP\_CC, [1543](#)  
 CFE\_EVS\_RESET\_ALL\_FILTERS\_CC, [1544](#)  
 CFE\_EVS\_RESET\_APP\_COUNTER\_CC, [1544](#)  
 CFE\_EVS\_RESET\_COUNTERS\_CC, [1545](#)  
 CFE\_EVS\_RESET\_FILTER\_CC, [1546](#)  
 CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC,  
     [1547](#)  
 CFE\_EVS\_SET\_FILTER\_CC, [1548](#)  
 CFE\_EVS\_SET\_LOG\_MODE\_CC, [1549](#)  
 CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC, [1550](#)  
 CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC, [1550](#)  
**CFE\_EVS\_FILE\_WRITE\_ERROR**  
     cFE Return Code Defines, [240](#)  
**CFE\_EVS\_FILTER\_MAX\_EID**  
     cfe\_evs\_eventids.h, [1529](#)  
**CFE\_EVS\_FIRST\_16\_STOP**  
     cfe\_evs\_api\_typedefs.h, [1365](#)  
**CFE\_EVS\_FIRST\_32\_STOP**  
     cfe\_evs\_api\_typedefs.h, [1365](#)  
**CFE\_EVS\_FIRST\_4\_STOP**  
     cfe\_evs\_api\_typedefs.h, [1365](#)  
**CFE\_EVS\_FIRST\_64\_STOP**  
     cfe\_evs\_api\_typedefs.h, [1365](#)  
**CFE\_EVS\_FIRST\_8\_STOP**  
     cfe\_evs\_api\_typedefs.h, [1365](#)  
**CFE\_EVS\_FIRST\_ONE\_STOP**  
     cfe\_evs\_api\_typedefs.h, [1365](#)  
**CFE\_EVS\_FIRST\_TWO\_STOP**  
     cfe\_evs\_api\_typedefs.h, [1365](#)  
**CFE\_EVS\_FunctionCode\_**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_ADD\_EVENT\_FILTER**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_CLEAR\_LOG**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_DELETE\_EVENT\_FILTER**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_DISABLE\_APP\_EVENT\_TYPE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_DISABLE\_APP\_EVENTS**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_DISABLE\_EVENT\_TYPE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_DISABLE\_PORTS**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_ENABLE\_APP\_EVENT\_TYPE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_ENABLE\_APP\_EVENTS**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_ENABLE\_EVENT\_TYPE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_ENABLE\_PORTS**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_NOOP**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_RESET\_ALL\_FILTERS**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_RESET\_APP\_COUNTER**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_RESET\_COUNTERS**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_RESET\_FILTER**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_SET\_EVENT\_FORMAT\_MODE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_SET\_FILTER**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_SET\_LOG\_MODE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_WRITE\_APP\_DATA\_FILE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_FunctionCode\_WRITE\_LOG\_DATA\_FILE**  
     default\_cfe\_evs\_fcncode\_values.h, [1510](#)  
**CFE\_EVS\_HK\_TLM\_MID**  
     default\_cfe\_evs\_msgids.h, [1517](#)  
**CFE\_EVS\_HousekeepingTlm**, [686](#)  
     Payload, [687](#)  
     TelemetryHeader, [687](#)  
**CFE\_EVS\_HousekeepingTlm\_Payload**, [687](#)  
     AppData, [688](#)  
     CommandCounter, [688](#)  
     CommandErrorCounter, [688](#)  
     LogEnabled, [688](#)  
     LogFullFlag, [688](#)  
     LogMode, [688](#)  
     LogOverflowCounter, [688](#)  
     MessageFormatMode, [689](#)  
     MessageSendCounter, [689](#)  
     MessageTruncCounter, [689](#)  
     OutputPort, [689](#)  
     Spare1, [689](#)  
     Spare2, [689](#)  
     Spare3, [689](#)  
     UnregisteredAppCounter, [690](#)  
**CFE\_EVS\_HousekeepingTlm\_Payload\_t**  
     default\_cfe\_evs\_msgdefs.h, [1515](#)  
**CFE\_EVS\_HousekeepingTlm\_t**  
     default\_cfe\_evs\_msgstruct.h, [1519](#)  
**CFE\_EVS\_INFORMATION\_BIT**  
     default\_cfe\_evs\_msgdefs.h, [1514](#)  
**cfe\_evs\_interface\_cfg.h**  
     CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH,  
         [1552](#)  
     DEFAULT\_CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH,  
         [1552](#)  
**cfe\_evs\_internal\_cfg.h**  
     CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC,  
         [1553](#)

CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE, CFE\_EVS\_LOGMODE\_EID  
1553 cfe\_evs\_eventids.h, 1529

CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE, 1554 CFE\_EVS\_LogMode\_Enum\_t  
CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE, default\_cfe\_evs\_extern\_typedefs.h, 1508  
1554 CFE\_EVS\_LogMode\_OVERWRITE

CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE, default\_cfe\_evs\_extern\_typedefs.h, 1509  
1554 CFE\_EVS\_LONG\_EVENT\_MSG\_MID  
CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG, default\_cfe\_evs\_msgids.h, 1517  
1554 CFE\_EVS\_LongEventTlm, 690

CFE\_PLATFORM\_EVS\_LOG\_MAX, 1555 Payload, 691

CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST, TelemetryHeader, 691  
1555 CFE\_EVS\_LongEventTlm\_Payload, 691

CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS, Message, 691  
1555 PacketID, 691

CFE\_PLATFORM\_EVS\_PORT\_DEFAULT, 1556 Spare1, 692

CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY, Spare2, 692  
1556 CFE\_EVS\_LongEventTlm\_Payload\_t

CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE, default\_cfe\_evs\_msgdefs.h, 1515  
1556 CFE\_EVS\_LongEventTlm\_t

DEFAULT\_CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SECOND, default\_cfe\_evs\_msgstruct.h, 1519  
1557 CFE\_EVS\_MsgFormat

DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE, default\_cfe\_evs\_extern\_typedefs.h, 1509  
1557 CFE\_EVS\_MsgFormat\_Enum\_t

DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE, default\_cfe\_evs\_extern\_typedefs.h, 1508  
1557 CFE\_EVS\_MsgFormat\_LONG

DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE, default\_cfe\_evs\_extern\_typedefs.h, 1509  
1557 CFE\_EVS\_MsgFormat\_SHORT

DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE, default\_cfe\_evs\_extern\_typedefs.h, 1509  
1557 CFE\_EVS\_NO\_FILTER

DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG, cfe\_evs\_api\_typedefs.h, 1365  
1557 CFE\_EVS\_NOOP\_CC

DEFAULT\_CFE\_PLATFORM\_EVS\_LOG\_MAX, 1557 cfe\_evs\_fcncodes.h, 1543

DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURSTS, CFE\_EVS\_NOOP\_EID  
1557 cfe\_evs\_eventids.h, 1530

DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS, CFE\_EVS\_NoopCmd, 692  
1558 CommandHeader, 692

DEFAULT\_CFE\_PLATFORM\_EVS\_PORT\_DEFAULT, CFE\_EVS\_NoopCmd\_t  
1558 default\_cfe\_evs\_msgstruct.h, 1519

DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY, CFE\_EVS\_NOT\_IMPLEMENTED  
1558 cFE Return Code Defines, 240

DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE, CFE\_EVS\_PacketID, 692  
1558 AppName, 693

CFE\_EVS\_INVALID\_PARAMETER EventID, 693

cFE Return Code Defines, 240 EventType, 693

CFE\_EVS\_LEN\_ERR\_EID ProcessorID, 693

cfe\_evs\_eventids.h, 1529 SpacecraftID, 693

CFE\_EVS\_LogFileCmd\_Payload, 690 CFE\_EVS\_PacketID\_t

LogFilename, 690 default\_cfe\_evs\_msgdefs.h, 1515

CFE\_EVS\_LogFileCmd\_Payload\_t CFE\_EVS\_PORT1\_BIT

default\_cfe\_evs\_msgdefs.h, 1515 default\_cfe\_evs\_msgdefs.h, 1514

CFE\_EVS\_LogMode CFE\_EVS\_PORT2\_BIT

default\_cfe\_evs\_extern\_typedefs.h, 1509 default\_cfe\_evs\_msgdefs.h, 1514

CFE\_EVS\_LogMode\_DISCARD CFE\_EVS\_PORT3\_BIT

default\_cfe\_evs\_extern\_typedefs.h, 1509 default\_cfe\_evs\_msgdefs.h, 1514

CFE\_EVS\_PORT4\_BIT  
     default\_cfe\_evs\_msgdefs.h, 1514

CFE\_EVS\_Register  
     cFE Registration APIs, 296

CFE\_EVS\_RESET\_ALL\_FILTERS\_CC  
     cfe\_evs\_fcncodes.h, 1544

CFE\_EVS\_RESET\_APP\_COUNTER\_CC  
     cfe\_evs\_fcncodes.h, 1544

CFE\_EVS\_RESET\_AREA\_POINTER  
     cFE Return Code Defines, 241

CFE\_EVS\_RESET\_COUNTERS\_CC  
     cfe\_evs\_fcncodes.h, 1545

CFE\_EVS\_RESET\_FILTER\_CC  
     cfe\_evs\_fcncodes.h, 1546

CFE\_EVS\_ResetAllFilters  
     cFE Reset Event Filter APIs, 301

CFE\_EVS\_ResetAllFiltersCmd, 694  
     CommandHeader, 694  
     Payload, 694

CFE\_EVS\_ResetAllFiltersCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1519

CFE\_EVS\_ResetAppCounterCmd, 694  
     CommandHeader, 695  
     Payload, 695

CFE\_EVS\_ResetAppCounterCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1519

CFE\_EVS\_ResetCountersCmd, 695  
     CommandHeader, 695

CFE\_EVS\_ResetCountersCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1519

CFE\_EVS\_ResetFilter  
     cFE Reset Event Filter APIs, 302

CFE\_EVS\_ResetFilterCmd, 695  
     CommandHeader, 696  
     Payload, 696

CFE\_EVS\_ResetFilterCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1520

CFE\_EVS\_RSTALLFILTER\_EID  
     cfe\_evs\_eventids.h, 1530

CFE\_EVS\_RSTCNT\_EID  
     cfe\_evs\_eventids.h, 1530

CFE\_EVS\_RSTEVTCNT\_EID  
     cfe\_evs\_eventids.h, 1530

CFE\_EVS\_RSTFILTER\_EID  
     cfe\_evs\_eventids.h, 1531

CFE\_EVS\_Send  
     cfe\_evs.h, 1363

CFE\_EVS\_SEND\_HK\_MID  
     default\_cfe\_evs\_msgids.h, 1517

CFE\_EVS\_SendCrit  
     cfe\_evs.h, 1363

CFE\_EVS\_SendDbg  
     cfe\_evs.h, 1363

CFE\_EVS\_SendErr

          cfe\_evs.h, 1363

CFE\_EVS\_SendEvent  
     cFE Send Event APIs, 297

CFE\_EVS\_SendEventWithAppID  
     cFE Send Event APIs, 299

CFE\_EVS\_SendHkCmd, 696  
     CommandHeader, 696

CFE\_EVS\_SendHkCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1520

CFE\_EVS\_SendInfo  
     cfe\_evs.h, 1363

CFE\_EVS\_SendTimedEvent  
     cFE Send Event APIs, 300

CFE\_EVS\_SET\_EVENT\_FORMAT\_MODE\_CC  
     cfe\_evs\_fcncodes.h, 1547

CFE\_EVS\_SET\_FILTER\_CC  
     cfe\_evs\_fcncodes.h, 1548

CFE\_EVS\_SET\_LOG\_MODE\_CC  
     cfe\_evs\_fcncodes.h, 1549

CFE\_EVS\_SetEventFormatCode\_Payload, 696  
     MsgFormat, 697  
     Spare, 697

CFE\_EVS\_SetEventFormatMode\_Payload\_t  
     default\_cfe\_evs\_msgdefs.h, 1515

CFE\_EVS\_SetEventFormatModeCmd, 697  
     CommandHeader, 697  
     Payload, 697

CFE\_EVS\_SetEventFormatModeCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1520

CFE\_EVS\_SETEVTFMTMOD\_EID  
     cfe\_evs\_eventids.h, 1531

CFE\_EVS\_SetFilterCmd, 698  
     CommandHeader, 698  
     Payload, 698

CFE\_EVS\_SetFilterCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1520

CFE\_EVS\_SETFILTERMSK\_EID  
     cfe\_evs\_eventids.h, 1531

CFE\_EVS\_SetLogMode\_Payload, 698  
     LogMode, 698  
     Spare, 699

CFE\_EVS\_SetLogMode\_Payload\_t  
     default\_cfe\_evs\_msgdefs.h, 1515

CFE\_EVS\_SetLogModeCmd, 699  
     CommandHeader, 699  
     Payload, 699

CFE\_EVS\_SetLogModeCmd\_t  
     default\_cfe\_evs\_msgstruct.h, 1520

CFE\_EVS\_SHORT\_EVENT\_MSG\_MID  
     default\_cfe\_evs\_msgids.h, 1517

CFE\_EVS\_ShortEventTlm, 699  
     Payload, 700  
     TelemetryHeader, 700

CFE\_EVS\_ShortEventTlm\_Payload, 700

PacketID, 700  
CFE\_EVS\_ShortEventTlm\_Payload\_t  
  default\_cfe\_evs\_msgdefs.h, 1515  
CFE\_EVS\_ShortEventTlm\_t  
  default\_cfe\_evs\_msgstruct.h, 1520  
CFE\_EVS\_SQUELCHED\_ERR\_EID  
  cfe\_evs\_eventids.h, 1531  
CFE\_EVS\_STARTUP\_EID  
  cfe\_evs\_eventids.h, 1532  
cfe\_evs\_topicids.h  
  CFE\_MISSION\_EVS\_CMD\_TOPICID, 1558  
  CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID, 1559  
  CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID,  
    1559  
  CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID, 1559  
  CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID,  
    1559  
  DEFAULT\_CFE\_MISSION\_EVS\_CMD\_TOPICID,  
    1559  
  DEFAULT\_CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID,  
    1559  
  DEFAULT\_CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID,  
    1559  
  DEFAULT\_CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID, CFE\_FS\_BackgroundFileDumpIsPending  
    cFE File Utility APIs, 307  
  DEFAULT\_CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID,  
    1560  
CFE\_EVS\_UNKNOWN\_FILTER  
  cFE Return Code Defines, 241  
CFE\_EVS\_WRDAT\_EID  
  cfe\_evs\_eventids.h, 1532  
CFE\_EVS\_WRITE\_APP\_DATA\_FILE\_CC  
  cfe\_evs\_fcncodes.h, 1550  
CFE\_EVS\_WRITE\_HEADER\_ERR\_EID  
  cfe\_evs\_eventids.h, 1532  
CFE\_EVS\_WRITE\_LOG\_DATA\_FILE\_CC  
  cfe\_evs\_fcncodes.h, 1550  
CFE\_EVS\_WriteAppDataFileCmd, 700  
  CommandHeader, 701  
  Payload, 701  
CFE\_EVS\_WriteAppDataFileCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1520  
CFE\_EVS\_WriteLogDataFileCmd, 701  
  CommandHeader, 701  
  Payload, 701  
CFE\_EVS\_WriteLogDataFileCmd\_t  
  default\_cfe\_evs\_msgstruct.h, 1520  
CFE\_EVS\_WRLOG\_EID  
  cfe\_evs\_eventids.h, 1532  
CFE\_EXECUTIVE\_SERVICE  
  cfe\_error.h, 1351  
CFE\_FILE\_SERVICE  
  cfe\_error.h, 1352  
cfe\_fs\_api\_typedefs.h  
CFE\_FS\_FileCategory\_BINARY\_DATA\_DUMP,  
  1368  
CFE\_FS\_FileCategory\_DYNAMIC\_MODULE, 1368  
CFE\_FS\_FileCategory\_MAX, 1368  
CFE\_FS\_FileCategory\_SCRIPT, 1368  
CFE\_FS\_FileCategory\_t, 1368  
CFE\_FS\_FileCategory\_TEMP, 1368  
CFE\_FS\_FileCategory\_TEXT\_LOG, 1368  
CFE\_FS\_FileCategory\_UNKNOWN, 1368  
CFE\_FS\_FileWriteEvent\_COMPLETE, 1369  
CFE\_FS\_FileWriteEvent\_CREATE\_ERROR, 1369  
CFE\_FS\_FileWriteEvent\_HEADER\_WRITE\_ERROR,  
  1369  
CFE\_FS\_FileWriteEvent\_MAX, 1369  
CFE\_FS\_FileWriteEvent\_RECORD\_WRITE\_ERROR,  
  1369  
CFE\_FS\_FileWriteEvent\_t, 1368  
CFE\_FS\_FileWriteEvent\_UNDEFINED, 1369  
CFE\_FS\_FileWriteGetData\_t, 1367  
CFE\_FS\_FileWriteMetaData\_t, 1368  
CFE\_FS\_FileWriteOnEvent\_t, 1368  
CFE\_FS\_FILE\_CONTENT\_ID  
  cfe\_fs\_interface\_cfg.h, 1563  
  example\_mission\_cfg.h, 1276  
CFE\_FS\_FileCategory\_BINARY\_DATA\_DUMP  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileCategory\_DYNAMIC\_MODULE  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileCategory\_MAX  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileCategory\_SCRIPT  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileCategory\_t  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileCategory\_TEMP  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileCategory\_TEXT\_LOG  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileCategory\_UNKNOWN  
  cfe\_fs\_api\_typedefs.h, 1368  
CFE\_FS\_FileWriteEvent\_COMPLETE  
  cfe\_fs\_api\_typedefs.h, 1369  
CFE\_FS\_FileWriteEvent\_CREATE\_ERROR  
  cfe\_fs\_api\_typedefs.h, 1369  
CFE\_FS\_FileWriteEvent\_HEADER\_WRITE\_ERROR  
  cfe\_fs\_api\_typedefs.h, 1369

CFE\_FS\_FileWriteEvent\_MAX  
     `cfe_fs_api_typedefs.h, 1369`

CFE\_FS\_FileWriteEvent\_RECORD\_WRITE\_ERROR  
     `cfe_fs_api_typedefs.h, 1369`

CFE\_FS\_FileWriteEvent\_t  
     `cfe_fs_api_typedefs.h, 1368`

CFE\_FS\_FileWriteEvent\_UNDEFINED  
     `cfe_fs_api_typedefs.h, 1369`

CFE\_FS\_FileWriteGetData\_t  
     `cfe_fs_api_typedefs.h, 1367`

CFE\_FS\_FileWriteMetaData, 702  
     Description, 702  
     FileName, 702  
     FileSubType, 702  
     GetData, 702  
     IsPending, 702  
     OnEvent, 703

CFE\_FS\_FileWriteMetaData\_t  
     `cfe_fs_api_typedefs.h, 1368`

CFE\_FS\_FileWriteOnEvent\_t  
     `cfe_fs_api_typedefs.h, 1368`

CFE\_FS\_FNAME\_TOO\_LONG  
     cFE Return Code Defines, 241

CFE\_FS\_GetDefaultExtension  
     cFE File Utility APIs, 308

CFE\_FS\_GetDefaultMountPoint  
     cFE File Utility APIs, 309

CFE\_FS\_HDR\_DESC\_MAX\_LEN  
     `cfe_fs_interface_cfg.h, 1563`  
     `example_mission_cfg.h, 1276`

CFE\_FS\_Header, 703  
     ApplicationID, 703  
     ContentType, 703  
     Description, 704  
     Length, 704  
     ProcessorID, 704  
     SpacecraftID, 704  
     SubType, 704  
     TimeSeconds, 704  
     TimeSubSeconds, 704

CFE\_FS\_Header\_t  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_InitHeader  
     cFE File Header Management APIs, 303

`cfe_fs_interface_cfg.h`  
     CFE\_FS\_FILE\_CONTENT\_ID, 1563  
     CFE\_FS\_HDR\_DESC\_MAX\_LEN, 1563  
     DEFAULT\_CFE\_FS\_FILE\_CONTENT\_ID, 1563  
     DEFAULT\_CFE\_FS\_HDR\_DESC\_MAX\_LEN, 1564

CFE\_FS\_INVALID\_PATH  
     cFE Return Code Defines, 241

CFE\_FS\_NOT\_IMPLEMENTED  
     cFE Return Code Defines, 241

CFE\_FS\_ParseInputFileName  
     cFE File Utility APIs, 309  
     `cfe_fs_parseinputfilenameex.h, 1561`

CFE\_FS\_ParseInputFileNameEx  
     cFE File Utility APIs, 310

CFE\_FS\_ReadHeader  
     cFE File Header Management APIs, 303

CFE\_FS\_SetTimestamp  
     cFE File Header Management APIs, 304

CFE\_FS\_SubType  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_Enum\_t  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_ES\_CDS\_REG  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_ES\_ERLOG  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_ES\_PERFDATA  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_ES\_QUERYALL  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_ES\_QUERYALLTASKS  
     `default_cfe_fs_filedef.h, 1562`

CFE\_FS\_SubType\_ES\_SYSLOG  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_EVS\_APPDATA  
     `default_cfe_fs_filedef.h, 1562`

CFE\_FS\_SubType\_EVS\_EVENTLOG  
     `default_cfe_fs_filedef.h, 1562`

CFE\_FS\_SubType\_SB\_MAPDATA  
     `default_cfe_fs_filedef.h, 1562`

CFE\_FS\_SubType\_SB\_PIPEDATA  
     `default_cfe_fs_filedef.h, 1562`

CFE\_FS\_SubType\_SB\_ROUTEDATA  
     `default_cfe_fs_filedef.h, 1562`

CFE\_FS\_SubType\_TBL\_IMG  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_SubType\_TBL\_REG  
     `default_cfe_fs_filedef.h, 1561`

CFE\_FS\_WriteHeader  
     cFE File Header Management APIs, 305

CFE\_GENERIC\_SERVICE  
     `cfe_error.h, 1352`

CFE\_GLOBAL\_BASE\_MIDVAL  
     `default_cfe_core_api_base_mgid_values.h, 1333`

CFE\_GLOBAL\_CMD\_TOPICID\_TO\_MIDV  
     `default_cfe_core_api_mgid_mapping.h, 1334`

CFE\_GLOBAL\_TLM\_TOPICID\_TO\_MIDV  
     `default_cfe_core_api_mgid_mapping.h, 1334`

CFE\_LAST\_OFFICIAL  
     `cfe_version.h, 1400`

CFE\_MAJOR\_VERSION  
     `cfe_version.h, 1400`

CFE\_MAKE\_BIG16  
     `cfe_endian.h, 1345`

CFE\_MAKE\_BIG32

cfe\_endian.h, 1345  
CFE\_MINOR\_VERSION  
  cfe\_version.h, 1400  
CFE\_MISSION\_CF\_CH0\_RX\_TOPICID  
  cf\_topicids.h, 827  
CFE\_MISSION\_CF\_CH0\_TX\_TOPICID  
  cf\_topicids.h, 827  
CFE\_MISSION\_CF\_CH1\_RX\_TOPICID  
  cf\_topicids.h, 827  
CFE\_MISSION\_CF\_CH1\_TX\_TOPICID  
  cf\_topicids.h, 827  
CFE\_MISSION\_CF\_CMD\_TOPICID  
  cf\_topicids.h, 828  
CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID  
  cf\_topicids.h, 828  
CFE\_MISSION\_CF\_HK\_TLM\_TOPICID  
  cf\_topicids.h, 828  
CFE\_MISSION\_CF\_SEND\_HK\_TOPICID  
  cf\_topicids.h, 828  
CFE\_MISSION\_CF\_TIDVAL  
  default\_cf\_topicid\_values.h, 813  
  eds\_cf\_topicid\_values.h, 816  
CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID  
  cf\_topicids.h, 828  
CFE\_MISSION\_CORE\_API\_CFGVAL  
  default\_cfe\_core\_api\_interface\_cfg\_values.h, 1333  
CFE\_MISSION\_ES\_APP\_TLM\_TOPICID  
  cfe\_es\_topicids.h, 1505  
CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN  
  cfe\_es\_interface\_cfg.h, 1470  
  example\_mission\_cfg.h, 1276  
CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH  
  cfe\_es\_interface\_cfg.h, 1470  
  example\_mission\_cfg.h, 1276  
CFE\_MISSION\_ES\_CFGVAL  
  default\_cfe\_es\_interface\_cfg\_values.h, 1410  
CFE\_MISSION\_ES\_CMD\_TOPICID  
  cfe\_es\_topicids.h, 1505  
CFE\_MISSION\_ES\_CRC\_16  
  cfe\_es\_interface\_cfg.h, 1470  
  example\_mission\_cfg.h, 1276  
CFE\_MISSION\_ES\_CRC\_32  
  cfe\_es\_interface\_cfg.h, 1470  
  example\_mission\_cfg.h, 1276  
CFE\_MISSION\_ES\_CRC\_8  
  cfe\_es\_interface\_cfg.h, 1471  
  example\_mission\_cfg.h, 1277  
CFE\_MISSION\_ES\_DEFAULT\_CRC  
  cfe\_es\_interface\_cfg.h, 1471  
  example\_mission\_cfg.h, 1277  
CFE\_MISSION\_ES\_HK\_TLM\_TOPICID  
  cfe\_es\_topicids.h, 1505  
CFE\_MISSION\_ES\_MAIN\_PERF\_ID  
  cfe\_perfids.h, 1268  
CFE\_MISSION\_ES\_MAX\_APPLICATIONS  
  cfe\_es\_interface\_cfg.h, 1471  
  example\_mission\_cfg.h, 1277  
CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID  
  cfe\_es\_topicids.h, 1506  
CFE\_MISSION\_ES\_PERF\_EXIT\_BIT  
  cfe\_perfids.h, 1268  
CFE\_MISSION\_ES\_PERF\_MAX\_IDS  
  cfe\_es\_interface\_cfg.h, 1471  
  example\_mission\_cfg.h, 1277  
CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS  
  cfe\_es\_interface\_cfg.h, 1472  
  example\_mission\_cfg.h, 1277  
CFE\_MISSION\_ES\_SEND\_HK\_TOPICID  
  cfe\_es\_topicids.h, 1506  
CFE\_MISSION\_ES\_TIDVAL  
  default\_cfe\_es\_topicid\_values.h, 1422  
CFE\_MISSION\_EVS\_CFGVAL  
  default\_cfe\_evs\_interface\_cfg\_values.h, 1511  
CFE\_MISSION\_EVS\_CMD\_TOPICID  
  cfe\_evs\_topicids.h, 1558  
CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID  
  cfe\_evs\_topicids.h, 1559  
CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID  
  cfe\_evs\_topicids.h, 1559  
CFE\_MISSION\_EVS\_MAIN\_PERF\_ID  
  cfe\_perfids.h, 1268  
CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH  
  cfe\_evs\_interface\_cfg.h, 1552  
  example\_mission\_cfg.h, 1278  
CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID  
  cfe\_evs\_topicids.h, 1559  
CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID  
  cfe\_evs\_topicids.h, 1559  
CFE\_MISSION\_EVS\_TIDVAL  
  default\_cfe\_evs\_topicid\_values.h, 1521  
CFE\_MISSION\_FS\_CFGVAL  
  default\_cfe\_fs\_interface\_cfg\_values.h, 1562  
CFE\_MISSION\_MAX\_API\_LEN  
  cfe\_core\_api\_interface\_cfg.h, 1342  
  example\_mission\_cfg.h, 1278  
CFE\_MISSION\_MAX\_FILE\_LEN  
  cfe\_core\_api\_interface\_cfg.h, 1343  
  example\_mission\_cfg.h, 1278  
CFE\_MISSION\_MAX\_NUM\_FILES  
  cfe\_core\_api\_interface\_cfg.h, 1343  
  example\_mission\_cfg.h, 1279  
CFE\_MISSION\_MAX\_PATH\_LEN  
  cfe\_core\_api\_interface\_cfg.h, 1343  
  example\_mission\_cfg.h, 1279  
CFE\_MISSION\_REV  
  cfe\_version.h, 1400  
CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID  
  cfe\_sb\_topicids.h, 1623

CFE\_MISSION\_SB\_CFGVAL  
     default\_cfe\_sb\_interface\_cfg\_values.h, 1569

CFE\_MISSION\_SB\_CMD\_TOPICID  
     cfe\_sb\_topicids.h, 1623

CFE\_MISSION\_SB\_HK\_TLM\_TOPICID  
     cfe\_sb\_topicids.h, 1624

CFE\_MISSION\_SB\_MAIN\_PERF\_ID  
     cfe\_perfids.h, 1268

CFE\_MISSION\_SB\_MAX\_PIPES  
     cfe\_sb\_interface\_cfg.h, 1607  
     example\_mission\_cfg.h, 1280

CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE  
     cfe\_sb\_interface\_cfg.h, 1607  
     example\_mission\_cfg.h, 1280

CFE\_MISSION\_SB\_MSG\_LIM\_PERF\_ID  
     cfe\_perfids.h, 1268

CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID  
     cfe\_sb\_topicids.h, 1624

CFE\_MISSION\_SB\_PIPE\_OFLOW\_PERF\_ID  
     cfe\_perfids.h, 1268

CFE\_MISSION\_SB\_SEND\_HK\_TOPICID  
     cfe\_sb\_topicids.h, 1624

CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID  
     cfe\_sb\_topicids.h, 1624

CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT  
     cfe\_sb\_interface\_cfg.h, 1607

CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID  
     cfe\_sb\_topicids.h, 1624

CFE\_MISSION\_SB\_TIDVAL  
     default\_cfe\_sb\_topicid\_values.h, 1577

CFE\_MISSION\_TBL\_CFGVAL  
     default\_cfe\_tbl\_interface\_cfg\_values.h, 1628

CFE\_MISSION\_TBL\_CMD\_TOPICID  
     cfe\_tbl\_topicids.h, 1685

CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID  
     cfe\_tbl\_topicids.h, 1685

CFE\_MISSION\_TBL\_MAIN\_PERF\_ID  
     cfe\_perfids.h, 1269

CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN  
     cfe\_tbl\_interface\_cfg.h, 1675  
     default\_cfe\_tbl\_mission\_cfg.h, 1629  
     example\_mission\_cfg.h, 1280

CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH  
     cfe\_tbl\_interface\_cfg.h, 1675  
     default\_cfe\_tbl\_mission\_cfg.h, 1629  
     example\_mission\_cfg.h, 1281

CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID  
     cfe\_tbl\_topicids.h, 1685

CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID  
     cfe\_tbl\_topicids.h, 1686

CFE\_MISSION\_TBL\_TIDVAL  
     default\_cfe\_tbl\_topicid\_values.h, 1645

CFE\_MISSION\_TIME\_AT\_TONE\_WAS  
     cfe\_time\_interface\_cfg.h, 1729

example\_mission\_cfg.h, 1281

CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE  
     cfe\_time\_interface\_cfg.h, 1729  
     example\_mission\_cfg.h, 1282

CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI  
     cfe\_time\_interface\_cfg.h, 1729  
     example\_mission\_cfg.h, 1282

CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC  
     cfe\_time\_interface\_cfg.h, 1730  
     example\_mission\_cfg.h, 1282

CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE  
     cfe\_time\_interface\_cfg.h, 1730  
     example\_mission\_cfg.h, 1282

CFE\_MISSION\_TIME\_CFGVAL  
     default\_cfe\_time\_interface\_cfg\_values.h, 1692

CFE\_MISSION\_TIME\_CMD\_TOPICID  
     cfe\_time\_topicids.h, 1744

CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID  
     cfe\_time\_topicids.h, 1744

CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS  
     cfe\_time\_interface\_cfg.h, 1730  
     example\_mission\_cfg.h, 1282

CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS  
     cfe\_time\_interface\_cfg.h, 1730  
     example\_mission\_cfg.h, 1282

CFE\_MISSION\_TIME\_DEF\_LEAPS  
     cfe\_time\_interface\_cfg.h, 1730  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_DEF\_MET\_SECS  
     cfe\_time\_interface\_cfg.h, 1730  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_DEF\_MET\_SUBS  
     cfe\_time\_interface\_cfg.h, 1731  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_DEF\_STCF\_SECS  
     cfe\_time\_interface\_cfg.h, 1731  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS  
     cfe\_time\_interface\_cfg.h, 1731  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID  
     cfe\_time\_topicids.h, 1745

CFE\_MISSION\_TIME\_EPOCH\_DAY  
     cfe\_time\_interface\_cfg.h, 1731  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_EPOCH\_HOUR  
     cfe\_time\_interface\_cfg.h, 1731  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_EPOCH\_MICROS  
     cfe\_time\_interface\_cfg.h, 1731  
     example\_mission\_cfg.h, 1283

CFE\_MISSION\_TIME\_EPOCH\_MINUTE  
     cfe\_time\_interface\_cfg.h, 1731  
     example\_mission\_cfg.h, 1284

CFE\_MISSION\_TIME\_EPOCH\_SECOND  
  cfe\_time\_interface\_cfg.h, 1731  
  example\_mission\_cfg.h, 1284  
CFE\_MISSION\_TIME\_EPOCH\_SECONDS  
  cfe\_time\_interface\_cfg.h, 1732  
CFE\_MISSION\_TIME\_EPOCH\_YEAR  
  cfe\_time\_interface\_cfg.h, 1732  
  example\_mission\_cfg.h, 1284  
CFE\_MISSION\_TIME\_FS\_FACTOR  
  cfe\_time\_interface\_cfg.h, 1732  
  example\_mission\_cfg.h, 1284  
CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID  
  cfe\_time\_topicids.h, 1745  
CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID  
  cfe\_perfids.h, 1269  
CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID  
  cfe\_perfids.h, 1269  
CFE\_MISSION\_TIME\_MAIN\_PERF\_ID  
  cfe\_perfids.h, 1269  
CFE\_MISSION\_TIME\_MAX\_ELAPSED  
  cfe\_time\_interface\_cfg.h, 1732  
  example\_mission\_cfg.h, 1284  
CFE\_MISSION\_TIME\_MIN\_ELAPSED  
  cfe\_time\_interface\_cfg.h, 1732  
  example\_mission\_cfg.h, 1284  
CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID  
  cfe\_time\_topicids.h, 1745  
CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID  
  cfe\_time\_topicids.h, 1745  
CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID  
  cfe\_time\_topicids.h, 1745  
CFE\_MISSION\_TIME\_SEDMET\_PERF\_ID  
  cfe\_perfids.h, 1269  
CFE\_MISSION\_TIME\_TIDVAL  
  default\_cfe\_time\_topicid\_values.h, 1701  
CFE\_MISSION\_TIME\_TONE1HZISR\_PERF\_ID  
  cfe\_perfids.h, 1269  
CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID  
  cfe\_perfids.h, 1269  
CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID  
  cfe\_time\_topicids.h, 1745  
cfe\_msg\_api\_typedefs.h  
  CFE\_MSG\_Apld\_t, 1373  
  CFE\_MSG\_BAD\_ARGUMENT, 1372  
  CFE\_MSG\_Checksum\_t, 1373  
  CFE\_MSG\_CommandHeader\_t, 1373  
  CFE\_MSG\_EDSVersion\_t, 1373  
  CFE\_MSG\_Endian, 1374  
  CFE\_MSG\_Endian\_Big, 1374  
  CFE\_MSG\_Endian\_Invalid, 1374  
  CFE\_MSG\_Endian\_Little, 1374  
  CFE\_MSG\_Endian\_t, 1373  
  CFE\_MSG\_FcnCode\_t, 1373  
  CFE\_MSG\_HeaderVersion\_t, 1373  
CFE\_MSG\_Message\_t, 1373  
CFE\_MSG\_NOT\_IMPLEMENTED, 1372  
CFE\_MSG\_PlaybackFlag, 1374  
CFE\_MSG\_PlaybackFlag\_t, 1373  
CFE\_MSG\_PlayFlag\_Invalid, 1375  
CFE\_MSG\_PlayFlag\_Original, 1375  
CFE\_MSG\_PlayFlag\_Playback, 1375  
CFE\_MSG\_SegFlag\_Continue, 1375  
CFE\_MSG\_SegFlag\_First, 1375  
CFE\_MSG\_SegFlag\_Invalid, 1375  
CFE\_MSG\_SegFlag\_Last, 1375  
CFE\_MSG\_SegFlag\_Unsegmented, 1375  
CFE\_MSG\_SegmentationFlag, 1375  
CFE\_MSG\_SegmentationFlag\_t, 1374  
CFE\_MSG\_SequenceCount\_t, 1374  
CFE\_MSG\_Size\_t, 1374  
CFE\_MSG\_Subsystem\_t, 1374  
CFE\_MSG\_System\_t, 1374  
CFE\_MSG\_TelemetryHeader\_t, 1374  
CFE\_MSG\_Type, 1375  
CFE\_MSG\_Type\_Cmd, 1375  
CFE\_MSG\_Type\_Invalid, 1375  
CFE\_MSG\_Type\_t, 1374  
CFE\_MSG\_Type\_Tlm, 1375  
CFE\_MSG\_WRONG\_MSG\_TYPE, 1373  
CFE\_MSG\_Apld\_t  
  cfe\_msg\_api\_typedefs.h, 1373  
CFE\_MSG\_BAD\_ARGUMENT  
  cfe\_msg\_api\_typedefs.h, 1372  
CFE\_MSG\_Checksum\_t  
  cfe\_msg\_api\_typedefs.h, 1373  
CFE\_MSG\_CommandHeader\_t  
  cfe\_msg\_api\_typedefs.h, 1373  
CFE\_MSG\_EDSVersion\_t  
  cfe\_msg\_api\_typedefs.h, 1373  
CFE\_MSG\_Endian  
  cfe\_msg\_api\_typedefs.h, 1374  
CFE\_MSG\_Endian\_Big  
  cfe\_msg\_api\_typedefs.h, 1374  
CFE\_MSG\_Endian\_Invalid  
  cfe\_msg\_api\_typedefs.h, 1374  
CFE\_MSG\_Endian\_Little  
  cfe\_msg\_api\_typedefs.h, 1374  
CFE\_MSG\_Endian\_t  
  cfe\_msg\_api\_typedefs.h, 1373  
CFE\_MSG\_FcnCode\_t  
  cfe\_msg\_api\_typedefs.h, 1373  
CFE\_MSG\_GenerateChecksum  
  cFE Message Secondary Header APIs, 326  
CFE\_MSG\_GetApld  
  cFE Message Primary Header APIs, 312  
CFE\_MSG\_GetEDSVersion  
  cFE Message Extended Header APIs, 321  
CFE\_MSG\_GetEndian

cFE Message Extended Header APIs, 321  
CFE\_MSG\_GetFcnCode  
    cFE Message Secondary Header APIs, 327  
CFE\_MSG\_GetHasSecondaryHeader  
    cFE Message Primary Header APIs, 313  
CFE\_MSG\_GetHeaderVersion  
    cFE Message Primary Header APIs, 313  
CFE\_MSG\_GetMsgId  
    cFE Message Id APIs, 330  
CFE\_MSG\_GetMsgTime  
    cFE Message Secondary Header APIs, 327  
CFE\_MSG\_GetNextSequenceCount  
    cFE Message Primary Header APIs, 314  
CFE\_MSG\_GetPlaybackFlag  
    cFE Message Extended Header APIs, 322  
CFE\_MSG\_GetSegmentationFlag  
    cFE Message Primary Header APIs, 314  
CFE\_MSG\_GetSequenceCount  
    cFE Message Primary Header APIs, 315  
CFE\_MSG.GetSize  
    cFE Message Primary Header APIs, 315  
CFE\_MSG\_GetSubsystem  
    cFE Message Extended Header APIs, 322  
CFE\_MSG\_GetSystem  
    cFE Message Extended Header APIs, 323  
CFE\_MSG.GetType  
    cFE Message Primary Header APIs, 316  
CFE\_MSG.GetTypeFromMsgId  
    cFE Message Id APIs, 331  
CFE\_MSG\_HeaderVersion\_t  
    `cfe_msg_api_typedefs.h`, 1373  
CFE\_MSG\_Init  
    cFE Generic Message APIs, 311  
CFE\_MSG\_Message\_t  
    `cfe_msg_api_typedefs.h`, 1373  
CFE\_MSG\_NOT\_IMPLEMENTED  
    `cfe_msg_api_typedefs.h`, 1372  
CFE\_MSG\_OriginationAction  
    cFE Message Integrity APIs, 332  
CFE\_MSG\_PlaybackFlag  
    `cfe_msg_api_typedefs.h`, 1374  
CFE\_MSG\_PlaybackFlag\_t  
    `cfe_msg_api_typedefs.h`, 1373  
CFE\_MSG\_PlayFlag\_Invalid  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_PlayFlag\_Original  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_PlayFlag\_Playback  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_SegFlag\_Continue  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_SegFlag\_First  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_SegFlag\_Invalid  
    `cfe_msg_api_typedefs.h`, 1375  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_SegFlag\_Last  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_SegFlag\_Unsegmented  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_SegmentationFlag  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_SegmentationFlag\_t  
    `cfe_msg_api_typedefs.h`, 1374  
CFE\_MSG\_SequenceCount\_t  
    `cfe_msg_api_typedefs.h`, 1374  
CFE\_MSG\_SetApId  
    cFE Message Primary Header APIs, 316  
CFE\_MSG\_SetEDSVersion  
    cFE Message Extended Header APIs, 323  
CFE\_MSG\_SetEndian  
    cFE Message Extended Header APIs, 324  
CFE\_MSG\_SetFcnCode  
    cFE Message Secondary Header APIs, 328  
CFE\_MSG\_SetHasSecondaryHeader  
    cFE Message Primary Header APIs, 317  
CFE\_MSG\_SetHeaderVersion  
    cFE Message Primary Header APIs, 317  
CFE\_MSG\_SetMsgId  
    cFE Message Id APIs, 331  
CFE\_MSG\_SetMsgTime  
    cFE Message Secondary Header APIs, 329  
CFE\_MSG\_SetPlaybackFlag  
    cFE Message Extended Header APIs, 324  
CFE\_MSG\_SetSegmentationFlag  
    cFE Message Primary Header APIs, 318  
CFE\_MSG\_SetSequenceCount  
    cFE Message Primary Header APIs, 318  
CFE\_MSG\_SetSize  
    cFE Message Primary Header APIs, 319  
CFE\_MSG\_SetSubsystem  
    cFE Message Extended Header APIs, 325  
CFE\_MSG\_SetSystem  
    cFE Message Extended Header APIs, 325  
CFE\_MSG\_SetType  
    cFE Message Primary Header APIs, 319  
CFE\_MSG\_Size\_t  
    `cfe_msg_api_typedefs.h`, 1374  
CFE\_MSG\_Subsystem\_t  
    `cfe_msg_api_typedefs.h`, 1374  
CFE\_MSG\_System\_t  
    `cfe_msg_api_typedefs.h`, 1374  
CFE\_MSG\_TelemetryHeader\_t  
    `cfe_msg_api_typedefs.h`, 1374  
CFE\_MSG\_Type  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_Type\_Cmd  
    `cfe_msg_api_typedefs.h`, 1375  
CFE\_MSG\_Type\_Invalid

cfe\_msg\_api\_typedefs.h, 1375  
CFE\_MSG\_Type\_t  
  cfe\_msg\_api\_typedefs.h, 1374  
CFE\_MSG\_Type\_Tlm  
  cfe\_msg\_api\_typedefs.h, 1375  
CFE\_MSG\_ValidateChecksum  
  cFE Message Secondary Header APIs, 329  
CFE\_MSG\_VerificationAction  
  cFE Message Integrity APIs, 333  
CFE\_MSG\_WRONG\_MSG\_TYPE  
  cfe\_msg\_api\_typedefs.h, 1373  
cfe\_perfids.h  
  CFE\_MISSION\_ES\_MAIN\_PERF\_ID, 1268  
  CFE\_MISSION\_ES\_PERF\_EXIT\_BIT, 1268  
  CFE\_MISSION\_EVS\_MAIN\_PERF\_ID, 1268  
  CFE\_MISSION\_SB\_MAIN\_PERF\_ID, 1268  
  CFE\_MISSION\_SB\_MSG\_LIM\_PERF\_ID, 1268  
  CFE\_MISSION\_SB\_PIPE\_OFLOW\_PERF\_ID, 1268  
  CFE\_MISSION\_TBL\_MAIN\_PERF\_ID, 1269  
  CFE\_MISSION\_TIME\_LOCAL1HZISR\_PERF\_ID,  
    1269  
  CFE\_MISSION\_TIME\_LOCAL1HZTASK\_PERF\_ID,  
    1269  
  CFE\_MISSION\_TIME\_MAIN\_PERF\_ID, 1269  
  CFE\_MISSION\_TIME\_SENDMET\_PERF\_ID, 1269  
  CFE\_MISSION\_TIME\_TONE1HZISR\_PERF\_ID,  
    1269  
  CFE\_MISSION\_TIME\_TONE1HZTASK\_PERF\_ID,  
    1269  
CFE\_PLATFORM\_BASE\_MIDVAL  
  default\_cfe\_core\_api\_basemsgid\_values.h, 1333  
CFE\_PLATFORM\_CF\_CMD\_MIDVAL  
  default\_cf\_msgid\_values.h, 807  
CFE\_PLATFORM\_CF\_TLM\_MIDVAL  
  default\_cf\_msgid\_values.h, 807  
CFE\_PLATFORM\_CMD\_TOPICID\_TO\_MIDV  
  default\_cfe\_core\_apimsgid\_mapping.h, 1334  
CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC  
  example\_platform\_cfg.h, 1289  
CFE\_PLATFORM\_ENDIAN  
  example\_platform\_cfg.h, 1289  
CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT  
  cfe\_es\_internal\_cfg.h, 1478  
  example\_platform\_cfg.h, 1289  
CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE  
  cfe\_es\_internal\_cfg.h, 1478  
  example\_platform\_cfg.h, 1290  
CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE  
  cfe\_es\_internal\_cfg.h, 1479  
  example\_platform\_cfg.h, 1290  
CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES  
  cfe\_es\_internal\_cfg.h, 1479  
  example\_platform\_cfg.h, 1290  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01  
  cfe\_es\_internal\_cfg.h, 1479  
  example\_platform\_cfg.h, 1291  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02  
  cfe\_es\_internal\_cfg.h, 1479  
  example\_platform\_cfg.h, 1291  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03  
  cfe\_es\_internal\_cfg.h, 1479  
  example\_platform\_cfg.h, 1291  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04  
  cfe\_es\_internal\_cfg.h, 1479  
  example\_platform\_cfg.h, 1291  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1291  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1291  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14  
  cfe\_es\_internal\_cfg.h, 1480  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15  
  cfe\_es\_internal\_cfg.h, 1481  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16  
  cfe\_es\_internal\_cfg.h, 1481  
  example\_platform\_cfg.h, 1292  
CFE\_PLATFORM\_ES\_CDS\_SIZE  
  cfe\_es\_internal\_cfg.h, 1481  
  example\_platform\_cfg.h, 1293  
CFE\_PLATFORM\_ES\_CFGVAL  
  default\_cfe\_es\_internal\_cfg\_values.h, 1410  
CFE\_PLATFORM\_ES\_CMD\_MIDVAL  
  default\_cfe\_es\_msgid\_values.h, 1417

CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE  
  cfe\_es\_internal\_cfg.h, 1481  
  example\_platform\_cfg.h, 1293

CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE  
  cfe\_es\_internal\_cfg.h, 1481  
  example\_platform\_cfg.h, 1293

CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE  
  cfe\_es\_internal\_cfg.h, 1482  
  example\_platform\_cfg.h, 1293

CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME  
  cfe\_es\_internal\_cfg.h, 1482  
  example\_platform\_cfg.h, 1294

CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE  
  cfe\_es\_internal\_cfg.h, 1482  
  example\_platform\_cfg.h, 1294

CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE  
  cfe\_es\_internal\_cfg.h, 1483  
  example\_platform\_cfg.h, 1294

CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE  
  cfe\_es\_internal\_cfg.h, 1483  
  example\_platform\_cfg.h, 1295

CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE  
  cfe\_es\_internal\_cfg.h, 1483  
  example\_platform\_cfg.h, 1295

CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE  
  cfe\_es\_internal\_cfg.h, 1484  
  example\_platform\_cfg.h, 1295

CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES  
  cfe\_es\_internal\_cfg.h, 1484  
  example\_platform\_cfg.h, 1296

CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE  
  cfe\_es\_internal\_cfg.h, 1484  
  example\_platform\_cfg.h, 1296

CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS  
  cfe\_es\_internal\_cfg.h, 1485  
  example\_platform\_cfg.h, 1296

CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE  
  cfe\_es\_internal\_cfg.h, 1485  
  example\_platform\_cfg.h, 1297

CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS  
  cfe\_es\_internal\_cfg.h, 1485  
  example\_platform\_cfg.h, 1297

CFE\_PLATFORM\_ES\_MAX\_LIBRARIES  
  cfe\_es\_internal\_cfg.h, 1485  
  example\_platform\_cfg.h, 1297

CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS  
  cfe\_es\_internal\_cfg.h, 1486  
  example\_platform\_cfg.h, 1297

CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS  
  cfe\_es\_internal\_cfg.h, 1486  
  example\_platform\_cfg.h, 1298

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01  
  cfe\_es\_internal\_cfg.h, 1486  
  example\_platform\_cfg.h, 1298

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02  
  cfe\_es\_internal\_cfg.h, 1487  
  example\_platform\_cfg.h, 1298

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03  
  cfe\_es\_internal\_cfg.h, 1487  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04  
  cfe\_es\_internal\_cfg.h, 1487  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05  
  cfe\_es\_internal\_cfg.h, 1487  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06  
  cfe\_es\_internal\_cfg.h, 1487  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07  
  cfe\_es\_internal\_cfg.h, 1487  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08  
  cfe\_es\_internal\_cfg.h, 1487  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1299

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1300

CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1300

CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN  
  cfe\_es\_internal\_cfg.h, 1488  
  example\_platform\_cfg.h, 1300

CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING  
  cfe\_es\_internal\_cfg.h, 1489  
  example\_platform\_cfg.h, 1300

CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE  
  cfe\_es\_internal\_cfg.h, 1489  
  example\_platform\_cfg.h, 1300

- CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE  
  cfe\_es\_internal\_cfg.h, 1489  
  example\_platform\_cfg.h, 1301
- CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY  
  cfe\_es\_internal\_cfg.h, 1489  
  example\_platform\_cfg.h, 1301
- CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY  
  cfe\_es\_internal\_cfg.h, 1490  
  example\_platform\_cfg.h, 1301
- CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE  
  cfe\_es\_internal\_cfg.h, 1490  
  example\_platform\_cfg.h, 1301
- CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE  
  cfe\_es\_internal\_cfg.h, 1490  
  example\_platform\_cfg.h, 1302
- CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS  
  cfe\_es\_internal\_cfg.h, 1491  
  example\_platform\_cfg.h, 1302
- CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL  
  cfe\_es\_internal\_cfg.h, 1491  
  example\_platform\_cfg.h, 1302
- CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT  
  cfe\_es\_internal\_cfg.h, 1491  
  example\_platform\_cfg.h, 1302
- CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE  
  cfe\_es\_internal\_cfg.h, 1491  
  example\_platform\_cfg.h, 1303
- CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL  
  cfe\_es\_internal\_cfg.h, 1492  
  example\_platform\_cfg.h, 1303
- CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT  
  cfe\_es\_internal\_cfg.h, 1492  
  example\_platform\_cfg.h, 1303
- CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE  
  cfe\_es\_internal\_cfg.h, 1492  
  example\_platform\_cfg.h, 1303
- CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS  
  cfe\_es\_internal\_cfg.h, 1492  
  example\_platform\_cfg.h, 1303
- CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING  
  cfe\_es\_internal\_cfg.h, 1493  
  example\_platform\_cfg.h, 1304
- CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS  
  cfe\_es\_internal\_cfg.h, 1493  
  example\_platform\_cfg.h, 1304
- CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVEDCFE\_PLATFORM\_ES\_MAX\_APP\_EVENT\_BURST  
  cfe\_es\_internal\_cfg.h, 1493  
  example\_platform\_cfg.h, 1304
- CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE  
  cfe\_es\_internal\_cfg.h, 1494  
  example\_platform\_cfg.h, 1305
- CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY  
  cfe\_es\_internal\_cfg.h, 1494  
  example\_platform\_cfg.h, 1305
- CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE  
  cfe\_es\_internal\_cfg.h, 1494  
  example\_platform\_cfg.h, 1305
- CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC  
  cfe\_es\_internal\_cfg.h, 1495  
  example\_platform\_cfg.h, 1306
- CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC  
  cfe\_es\_internal\_cfg.h, 1495  
  example\_platform\_cfg.h, 1306
- CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE  
  cfe\_es\_internal\_cfg.h, 1495  
  example\_platform\_cfg.h, 1306
- CFE\_PLATFORM\_ES\_TLM\_MIDVAL  
  default\_cfe\_es\_msgid\_values.h, 1417
- CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE  
  cfe\_es\_internal\_cfg.h, 1496  
  example\_platform\_cfg.h, 1307
- CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE  
  cfe\_es\_internal\_cfg.h, 1496  
  example\_platform\_cfg.h, 1307
- CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC  
  cfe\_evs\_internal\_cfg.h, 1553  
  example\_platform\_cfg.h, 1307
- CFE\_PLATFORM\_EVS\_CFGVAL  
  default\_cfe\_evs\_internal\_cfg\_values.h, 1511
- CFE\_PLATFORM\_EVS\_CMD\_MIDVAL  
  default\_cfe\_evs\_msgid\_values.h, 1516
- CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE  
  cfe\_evs\_internal\_cfg.h, 1553  
  example\_platform\_cfg.h, 1308
- CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE  
  cfe\_evs\_internal\_cfg.h, 1554  
  example\_platform\_cfg.h, 1308
- CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE  
  cfe\_evs\_internal\_cfg.h, 1554  
  example\_platform\_cfg.h, 1308
- CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE  
  cfe\_evs\_internal\_cfg.h, 1554  
  example\_platform\_cfg.h, 1309
- CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG  
  cfe\_evs\_internal\_cfg.h, 1554  
  example\_platform\_cfg.h, 1309
- CFE\_PLATFORM\_EVS\_LOG\_MAX  
  cfe\_evs\_internal\_cfg.h, 1555  
  example\_platform\_cfg.h, 1309
- CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST  
  cfe\_evs\_internal\_cfg.h, 1555  
  example\_platform\_cfg.h, 1310
- CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS  
  cfe\_evs\_internal\_cfg.h, 1555  
  example\_platform\_cfg.h, 1310
- CFE\_PLATFORM\_EVS\_PORT\_DEFAULT  
  cfe\_evs\_internal\_cfg.h, 1556  
  example\_platform\_cfg.h, 1310

CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY  
     cfe\_evs\_internal\_cfg.h, 1556  
     example\_platform\_cfg.h, 1310

CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE  
     cfe\_evs\_internal\_cfg.h, 1556  
     example\_platform\_cfg.h, 1311

CFE\_PLATFORM\_EVS\_TLM\_MIDVAL  
     default\_cfe\_evs\_msgid\_values.h, 1516

CFE\_PLATFORM\_SB\_BUFS\_MEMORY\_BYTES  
     cfe\_sb\_internal\_cfg.h, 1610  
     example\_platform\_cfg.h, 1311

CFE\_PLATFORM\_SB\_CFGVAL  
     default\_cfe\_sb\_internal\_cfg\_values.h, 1570

CFE\_PLATFORM\_SB\_CMD\_MIDVAL  
     default\_cfe\_sb\_msgid\_values.h, 1573

CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME  
     cfe\_sb\_internal\_cfg.h, 1611  
     example\_platform\_cfg.h, 1311

CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT  
     cfe\_sb\_internal\_cfg.h, 1611  
     example\_platform\_cfg.h, 1312

CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME  
     cfe\_sb\_internal\_cfg.h, 1611  
     example\_platform\_cfg.h, 1312

CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME  
     cfe\_sb\_internal\_cfg.h, 1612  
     example\_platform\_cfg.h, 1312

CFE\_PLATFORM\_SB\_FILTER\_MASK1  
     cfe\_sb\_internal\_cfg.h, 1612  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK2  
     cfe\_sb\_internal\_cfg.h, 1612  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK3  
     cfe\_sb\_internal\_cfg.h, 1612  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK4  
     cfe\_sb\_internal\_cfg.h, 1612  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK5  
     cfe\_sb\_internal\_cfg.h, 1612  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK6  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK7  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK8  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTERED\_EVENT1  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1313

CFE\_PLATFORM\_SB\_FILTERED\_EVENT2  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_FILTERED\_EVENT3  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_FILTERED\_EVENT4  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_FILTERED\_EVENT5  
     cfe\_sb\_internal\_cfg.h, 1613  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_FILTERED\_EVENT6  
     cfe\_sb\_internal\_cfg.h, 1614  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_FILTERED\_EVENT7  
     cfe\_sb\_internal\_cfg.h, 1614  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_FILTERED\_EVENT8  
     cfe\_sb\_internal\_cfg.h, 1614  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID  
     cfe\_sb\_internal\_cfg.h, 1614  
     example\_platform\_cfg.h, 1314

CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE  
     cfe\_sb\_internal\_cfg.h, 1614  
     example\_platform\_cfg.h, 1315

CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT  
     cfe\_sb\_internal\_cfg.h, 1614  
     example\_platform\_cfg.h, 1315

CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS  
     cfe\_sb\_internal\_cfg.h, 1615  
     example\_platform\_cfg.h, 1315

CFE\_PLATFORM\_SB\_MAX\_PIPES  
     cfe\_sb\_internal\_cfg.h, 1615  
     example\_platform\_cfg.h, 1315

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01  
     cfe\_sb\_internal\_cfg.h, 1615  
     example\_platform\_cfg.h, 1316

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02  
     cfe\_sb\_internal\_cfg.h, 1616  
     example\_platform\_cfg.h, 1316

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03  
     cfe\_sb\_internal\_cfg.h, 1616  
     example\_platform\_cfg.h, 1316

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04  
     cfe\_sb\_internal\_cfg.h, 1616  
     example\_platform\_cfg.h, 1316

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05  
     cfe\_sb\_internal\_cfg.h, 1616  
     example\_platform\_cfg.h, 1316

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06  
     cfe\_sb\_internal\_cfg.h, 1616  
     example\_platform\_cfg.h, 1316

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07  
  cfe\_sb\_internal\_cfg.h, 1616  
  example\_platform\_cfg.h, 1316

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08  
  cfe\_sb\_internal\_cfg.h, 1616  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09  
  cfe\_sb\_internal\_cfg.h, 1616  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10  
  cfe\_sb\_internal\_cfg.h, 1617  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11  
  cfe\_sb\_internal\_cfg.h, 1617  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12  
  cfe\_sb\_internal\_cfg.h, 1617  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13  
  cfe\_sb\_internal\_cfg.h, 1617  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14  
  cfe\_sb\_internal\_cfg.h, 1617  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15  
  cfe\_sb\_internal\_cfg.h, 1617  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16  
  cfe\_sb\_internal\_cfg.h, 1617  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS  
  cfe\_sb\_internal\_cfg.h, 1617

CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY  
  cfe\_sb\_internal\_cfg.h, 1618  
  example\_platform\_cfg.h, 1317

CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE  
  cfe\_sb\_internal\_cfg.h, 1618  
  example\_platform\_cfg.h, 1318

CFE\_PLATFORM\_SB\_TLM\_MIDVAL  
  default\_cfe\_sb\_msgid\_values.h, 1574

CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES  
  cfe\_tbl\_internal\_cfg.h, 1677  
  default\_cfe\_tbl\_platform\_cfg.h, 1638  
  example\_platform\_cfg.h, 1318

CFE\_PLATFORM\_TBL\_CFGVAL  
  default\_cfe\_tbl\_internal\_cfg\_values.h, 1629

CFE\_PLATFORM\_TBL\_CMD\_MIDVAL  
  default\_cfe\_tbl\_msgid\_values.h, 1633

CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE  
  cfe\_tbl\_internal\_cfg.h, 1678  
  default\_cfe\_tbl\_platform\_cfg.h, 1638  
  example\_platform\_cfg.h, 1318

CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES  
  cfe\_tbl\_internal\_cfg.h, 1678

  default\_cfe\_tbl\_platform\_cfg.h, 1638  
  example\_platform\_cfg.h, 1319

CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE  
  cfe\_tbl\_internal\_cfg.h, 1678  
  default\_cfe\_tbl\_platform\_cfg.h, 1638  
  example\_platform\_cfg.h, 1319

CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES  
  cfe\_tbl\_internal\_cfg.h, 1678  
  default\_cfe\_tbl\_platform\_cfg.h, 1639  
  example\_platform\_cfg.h, 1319

CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES  
  cfe\_tbl\_internal\_cfg.h, 1679  
  default\_cfe\_tbl\_platform\_cfg.h, 1639  
  example\_platform\_cfg.h, 1320

CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS  
  cfe\_tbl\_internal\_cfg.h, 1679  
  default\_cfe\_tbl\_platform\_cfg.h, 1639  
  example\_platform\_cfg.h, 1320

CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS  
  cfe\_tbl\_internal\_cfg.h, 1679  
  default\_cfe\_tbl\_platform\_cfg.h, 1640  
  example\_platform\_cfg.h, 1320

CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE  
  cfe\_tbl\_internal\_cfg.h, 1680  
  default\_cfe\_tbl\_platform\_cfg.h, 1640  
  example\_platform\_cfg.h, 1321

CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY  
  cfe\_tbl\_internal\_cfg.h, 1680  
  default\_cfe\_tbl\_platform\_cfg.h, 1640  
  example\_platform\_cfg.h, 1321

CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE  
  cfe\_tbl\_internal\_cfg.h, 1680  
  default\_cfe\_tbl\_platform\_cfg.h, 1641  
  example\_platform\_cfg.h, 1321

CFE\_PLATFORM\_TBL\_TLM\_MIDVAL  
  default\_cfe\_tbl\_msgid\_values.h, 1633

CFE\_PLATFORM\_TBL\_U32FROM4CHARS  
  cfe\_tbl\_internal\_cfg.h, 1681  
  default\_cfe\_tbl\_platform\_cfg.h, 1641  
  example\_platform\_cfg.h, 1322

CFE\_PLATFORM\_TBL\_VALID\_PRID\_1  
  cfe\_tbl\_internal\_cfg.h, 1681  
  default\_cfe\_tbl\_platform\_cfg.h, 1641  
  example\_platform\_cfg.h, 1322

CFE\_PLATFORM\_TBL\_VALID\_PRID\_2  
  cfe\_tbl\_internal\_cfg.h, 1681  
  default\_cfe\_tbl\_platform\_cfg.h, 1641  
  example\_platform\_cfg.h, 1322

CFE\_PLATFORM\_TBL\_VALID\_PRID\_3  
  cfe\_tbl\_internal\_cfg.h, 1681  
  default\_cfe\_tbl\_platform\_cfg.h, 1642  
  example\_platform\_cfg.h, 1322

CFE\_PLATFORM\_TBL\_VALID\_PRID\_4  
  cfe\_tbl\_internal\_cfg.h, 1681

default\_cfe\_tbl\_platform\_cfg.h, 1642  
 example\_platform\_cfg.h, 1322  
**CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT**  
 cfe\_tbl\_internal\_cfg.h, 1682  
 default\_cfe\_tbl\_platform\_cfg.h, 1642  
 example\_platform\_cfg.h, 1322  
**CFE\_PLATFORM\_TBL\_VALID\_SCID\_1**  
 cfe\_tbl\_internal\_cfg.h, 1682  
 default\_cfe\_tbl\_platform\_cfg.h, 1642  
 example\_platform\_cfg.h, 1323  
**CFE\_PLATFORM\_TBL\_VALID\_SCID\_2**  
 cfe\_tbl\_internal\_cfg.h, 1682  
 default\_cfe\_tbl\_platform\_cfg.h, 1642  
 example\_platform\_cfg.h, 1323  
**CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT**  
 cfe\_tbl\_internal\_cfg.h, 1682  
 default\_cfe\_tbl\_platform\_cfg.h, 1642  
 example\_platform\_cfg.h, 1323  
**CFE\_PLATFORM\_TIME\_CFG\_CLIENT**  
 cfe\_time\_internal\_cfg.h, 1736  
 example\_platform\_cfg.h, 1323  
**CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY**  
 cfe\_time\_internal\_cfg.h, 1736  
 example\_platform\_cfg.h, 1324  
**CFE\_PLATFORM\_TIME\_CFG\_SERVER**  
 cfe\_time\_internal\_cfg.h, 1737  
 example\_platform\_cfg.h, 1324  
**CFE\_PLATFORM\_TIME\_CFG\_SIGNAL**  
 cfe\_time\_internal\_cfg.h, 1737  
 example\_platform\_cfg.h, 1324  
**CFE\_PLATFORM\_TIME\_CFG\_SOURCE**  
 cfe\_time\_internal\_cfg.h, 1737  
 example\_platform\_cfg.h, 1324  
**CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS**  
 cfe\_time\_internal\_cfg.h, 1738  
 example\_platform\_cfg.h, 1325  
**CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET**  
 cfe\_time\_internal\_cfg.h, 1738  
 example\_platform\_cfg.h, 1325  
**CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME**  
 cfe\_time\_internal\_cfg.h, 1738  
 example\_platform\_cfg.h, 1325  
**CFE\_PLATFORM\_TIME\_CFG\_START\_FLY**  
 cfe\_time\_internal\_cfg.h, 1738  
 example\_platform\_cfg.h, 1325  
**CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT**  
 cfe\_time\_internal\_cfg.h, 1739  
 example\_platform\_cfg.h, 1326  
**CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL**  
 cfe\_time\_internal\_cfg.h, 1739  
 example\_platform\_cfg.h, 1326  
**CFE\_PLATFORM\_TIME\_CFGVAL**  
 default\_cfe\_time\_internal\_cfg\_values.h, 1693  
**CFE\_PLATFORM\_TIME\_CMD\_MIDVAL**  
 default\_cfe\_time\_msgid\_values.h, 1696  
**CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL**  
 default\_cfe\_time\_msgid\_values.h, 1696  
**CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS**  
 cfe\_time\_internal\_cfg.h, 1739  
 example\_platform\_cfg.h, 1326  
**CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS**  
 cfe\_time\_internal\_cfg.h, 1740  
 example\_platform\_cfg.h, 1327  
**CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS**  
 cfe\_time\_internal\_cfg.h, 1740  
 example\_platform\_cfg.h, 1327  
**CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS**  
 cfe\_time\_internal\_cfg.h, 1740  
 example\_platform\_cfg.h, 1327  
**CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY**  
 cfe\_time\_internal\_cfg.h, 1740  
 example\_platform\_cfg.h, 1327  
**CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE**  
 cfe\_time\_internal\_cfg.h, 1740  
 example\_platform\_cfg.h, 1327  
**CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY**  
 cfe\_time\_internal\_cfg.h, 1740  
 example\_platform\_cfg.h, 1327  
**CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE**  
 cfe\_time\_internal\_cfg.h, 1741  
 example\_platform\_cfg.h, 1328  
**CFE\_PLATFORM\_TIME\_TLM\_MIDVAL**  
 default\_cfe\_time\_msgid\_values.h, 1697  
**CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY**  
 cfe\_time\_internal\_cfg.h, 1741  
 example\_platform\_cfg.h, 1328  
**CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE**  
 cfe\_time\_internal\_cfg.h, 1741  
 example\_platform\_cfg.h, 1328  
**CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV**  
 default\_cfe\_core\_api\_msgid\_mapping.h, 1335  
**cfe\_psp.h**  
 CFE\_PSP\_Main, 1790  
 PSP\_DEBUG, 1790  
 PSP\_DEBUG\_LEV, 1790  
**CFE\_PSP\_AttachExceptions**  
 cfe\_psp\_exception\_api.h, 1801  
**CFE\_PSP\_BE16toH**  
 cfe\_psp\_endian.h, 1796  
**CFE\_PSP\_BE32toH**  
 cfe\_psp\_endian.h, 1796  
**CFE\_PSP\_BE64toH**  
 cfe\_psp\_endian.h, 1797  
**cfe\_psp\_cache\_api.h**  
 CFE\_PSP\_FlushCaches, 1790  
**cfe\_psp\_cds\_api.h**  
 CFE\_PSP\_GetCDSSize, 1792  
 CFE\_PSP\_ReadFromCDS, 1792

CFE\_PSP\_WriteToCDS, 1793  
cfe\_psp\_eepromaccess\_api.h  
    CFE\_PSP\_EepromPowerDown, 1793  
    CFE\_PSP\_EepromPowerUp, 1794  
    CFE\_PSP\_EepromWrite16, 1794  
    CFE\_PSP\_EepromWrite32, 1794  
    CFE\_PSP\_EepromWrite8, 1795  
    CFE\_PSP\_EepromWriteDisable, 1795  
    CFE\_PSP\_EepromWriteEnable, 1796  
CFE\_PSP\_EepromPowerDown  
    cfe\_psp\_eepromaccess\_api.h, 1793  
CFE\_PSP\_EepromPowerUp  
    cfe\_psp\_eepromaccess\_api.h, 1794  
CFE\_PSP\_EepromWrite16  
    cfe\_psp\_eepromaccess\_api.h, 1794  
CFE\_PSP\_EepromWrite32  
    cfe\_psp\_eepromaccess\_api.h, 1794  
CFE\_PSP\_EepromWrite8  
    cfe\_psp\_eepromaccess\_api.h, 1795  
CFE\_PSP\_EepromWriteDisable  
    cfe\_psp\_eepromaccess\_api.h, 1795  
CFE\_PSP\_EepromWriteEnable  
    cfe\_psp\_eepromaccess\_api.h, 1796  
cfe\_psp\_endian.h  
    CFE\_PSP\_BE16toH, 1796  
    CFE\_PSP\_BE32toH, 1796  
    CFE\_PSP\_BE64toH, 1797  
    CFE\_PSP\_HtoBE16, 1797  
    CFE\_PSP\_HtoBE32, 1797  
    CFE\_PSP\_HtoBE64, 1797  
    CFE\_PSP\_HtoLE16, 1797  
    CFE\_PSP\_HtoLE32, 1797  
    CFE\_PSP\_HtoLE64, 1797  
    CFE\_PSP\_IsBigEndian, 1797  
    CFE\_PSP\_IsLittleEndian, 1797  
    CFE\_PSP\_LE16toH, 1797  
    CFE\_PSP\_LE32toH, 1797  
    CFE\_PSP\_LE64toH, 1797  
CFE\_PSP\_ERROR  
    cfe\_psp\_error.h, 1798  
cfe\_psp\_error.h  
    CFE\_PSP\_ERROR, 1798  
    CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED,  
        1798  
    CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED, 1799  
    CFE\_PSP\_ERROR\_TIMEOUT, 1799  
    CFE\_PSP\_INVALID\_INT\_NUM, 1799  
    CFE\_PSP\_INVALID\_MEM\_ADDR, 1799  
    CFE\_PSP\_INVALID\_MEM\_ATTR, 1799  
    CFE\_PSP\_INVALID\_MEM\_RANGE, 1799  
    CFE\_PSP\_INVALID\_MEM\_SIZE, 1799  
    CFE\_PSP\_INVALID\_MEM\_TYPE, 1799  
    CFE\_PSP\_INVALID\_MEM\_WORDSIZE, 1799  
    CFE\_PSP\_INVALID\_MODULE\_ID, 1799  
CFE\_PSP\_INVALID\_MODULE\_NAME, 1799  
CFE\_PSP\_INVALID\_POINTER, 1799  
CFE\_PSP\_NO\_EXCEPTION\_DATA, 1799  
CFE\_PSP\_STATUS\_C, 1800  
CFE\_PSP\_STATUS\_STRING\_LENGTH, 1800  
CFE\_PSP\_Status\_t, 1800  
CFE\_PSP\_StatusString\_t, 1800  
CFE\_PSP\_StatusToString, 1800  
CFE\_PSP\_SUCCESS, 1800  
CFE\_PSP\_ERROR\_ADDRESS\_MISALIGNED  
    cfe\_psp\_error.h, 1798  
CFE\_PSP\_ERROR\_NOT\_IMPLEMENTED  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_ERROR\_TIMEOUT  
    cfe\_psp\_error.h, 1799  
cfe\_psp\_exception\_api.h  
    CFE\_PSP\_AttachExceptions, 1801  
    CFE\_PSP\_Exception\_CopyContext, 1801  
    CFE\_PSP\_Exception\_GetCount, 1802  
    CFE\_PSP\_Exception\_GetSummary, 1802  
    CFE\_PSP\_SetDefaultExceptionEnvironment, 1802  
CFE\_PSP\_Exception\_CopyContext  
    cfe\_psp\_exception\_api.h, 1801  
CFE\_PSP\_Exception\_GetCount  
    cfe\_psp\_exception\_api.h, 1802  
CFE\_PSP\_Exception\_GetSummary  
    cfe\_psp\_exception\_api.h, 1802  
CFE\_PSP\_FlushCaches  
    cfe\_psp\_cache\_api.h, 1790  
CFE\_PSP\_Get\_Timebase  
    cfe\_psp\_timertick\_api.h, 1817  
CFE\_PSP\_GetBuildNumber  
    cfe\_psp\_version\_api.h, 1819  
CFE\_PSP\_GetCDSSize  
    cfe\_psp\_cds\_api.h, 1792  
CFE\_PSP\_GetCFETextSegmentInfo  
    cfe\_psp\_memrange\_api.h, 1809  
CFE\_PSP\_GetKernelTextSegmentInfo  
    cfe\_psp\_memrange\_api.h, 1809  
CFE\_PSP\_GetProcessorId  
    cfe\_psp\_id\_api.h, 1803  
CFE\_PSP\_GetProcessorName  
    cfe\_psp\_id\_api.h, 1803  
CFE\_PSP\_GetResetArea  
    cfe\_psp\_memrange\_api.h, 1810  
CFE\_PSP\_GetRestartType  
    cfe\_psp\_watchdog\_api.h, 1823  
CFE\_PSP\_GetSpacecraftId  
    cfe\_psp\_id\_api.h, 1803  
CFE\_PSP\_GetTime  
    cfe\_psp\_timertick\_api.h, 1817  
CFE\_PSP\_GetTimerLow32Rollover  
    cfe\_psp\_timertick\_api.h, 1818  
CFE\_PSP\_GetTimerTicksPerSecond

cfe\_psp\_timetick\_api.h, 1818  
CFE\_PSP\_GetUserReservedArea  
    cfe\_psp\_memrange\_api.h, 1810  
CFE\_PSP\_GetVersionCodeName  
    cfe\_psp\_version\_api.h, 1819  
CFE\_PSP\_GetVersionNumber  
    cfe\_psp\_version\_api.h, 1819  
CFE\_PSP\_GetVersionString  
    cfe\_psp\_version\_api.h, 1819  
CFE\_PSP\_GetVolatileDiskMem  
    cfe\_psp\_memrange\_api.h, 1810  
CFE\_PSP\_HtoBE16  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_HtoBE32  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_HtoBE64  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_HtoLE16  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_HtoLE32  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_HtoLE64  
    cfe\_psp\_endian.h, 1797  
cfe\_psp\_id\_api.h  
    CFE\_PSP\_GetProcessorId, 1803  
    CFE\_PSP\_GetProcessorName, 1803  
    CFE\_PSP\_GetSpacecraftId, 1803  
CFE\_PSP\_InitSSR  
    cfe\_psp\_ssr\_api.h, 1816  
CFE\_PSP\_INVALID\_INT\_NUM  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MEM\_ADDR  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MEM\_ATTR  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MEM\_RANGE  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MEM\_SIZE  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MEM\_TYPE  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MEM\_WORDSIZE  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MODULE\_ID  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_MODULE\_NAME  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_INVALID\_POINTER  
    cfe\_psp\_error.h, 1799  
CFE\_PSP\_IsBigEndian  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_IsLittleEndian  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_LE16toH  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_LE32toH  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_LE64toH  
    cfe\_psp\_endian.h, 1797  
CFE\_PSP\_Main  
    cfe\_psp.h, 1790  
CFE\_PSP\_MEM\_ANY  
    cfe\_psp\_memrange\_api.h, 1808  
CFE\_PSP\_MEM\_ATTR\_READ  
    cfe\_psp\_memrange\_api.h, 1808  
CFE\_PSP\_MEM\_ATTR\_READWRITE  
    cfe\_psp\_memrange\_api.h, 1808  
CFE\_PSP\_MEM\_ATTR\_WRITE  
    cfe\_psp\_memrange\_api.h, 1808  
CFE\_PSP\_MEM\_EEPROM  
    cfe\_psp\_memrange\_api.h, 1808  
CFE\_PSP\_MEM\_INVALID  
    cfe\_psp\_memrange\_api.h, 1808  
CFE\_PSP\_MEM\_RAM  
    cfe\_psp\_memrange\_api.h, 1808  
CFE\_PSP\_MEM\_SIZE\_BYTE  
    cfe\_psp\_memrange\_api.h, 1809  
CFE\_PSP\_MEM\_SIZE\_DWORD  
    cfe\_psp\_memrange\_api.h, 1809  
CFE\_PSP\_MEM\_SIZE\_WORD  
    cfe\_psp\_memrange\_api.h, 1809  
cfe\_psp\_memaccess\_api.h  
    CFE\_PSP\_MemCpy, 1804  
    CFE\_PSP\_MemRead16, 1804  
    CFE\_PSP\_MemRead32, 1805  
    CFE\_PSP\_MemRead8, 1805  
    CFE\_PSP\_MemSet, 1806  
    CFE\_PSP\_MemWrite16, 1806  
    CFE\_PSP\_MemWrite32, 1806  
    CFE\_PSP\_MemWrite8, 1807  
CFE\_PSP\_MemCpy  
    cfe\_psp\_memaccess\_api.h, 1804  
cfe\_psp\_memrange\_api.h  
    CFE\_PSP\_GetCFETextSegmentInfo, 1809  
    CFE\_PSP\_GetKernelTextSegmentInfo, 1809  
    CFE\_PSP\_GetResetArea, 1810  
    CFE\_PSP.GetUserReservedArea, 1810  
    CFE\_PSP\_GetVolatileDiskMem, 1810  
    CFE\_PSP\_MEM\_ANY, 1808  
    CFE\_PSP\_MEM\_ATTR\_READ, 1808  
    CFE\_PSP\_MEM\_ATTR\_READWRITE, 1808  
    CFE\_PSP\_MEM\_ATTR\_WRITE, 1808  
    CFE\_PSP\_MEM\_EEPROM, 1808  
    CFE\_PSP\_MEM\_INVALID, 1808  
    CFE\_PSP\_MEM\_RAM, 1808  
    CFE\_PSP\_MEM\_SIZE\_BYTE, 1809  
    CFE\_PSP\_MEM\_SIZE\_DWORD, 1809  
    CFE\_PSP\_MEM\_SIZE\_WORD, 1809

CFE\_PSP\_MemRangeGet, 1810  
CFE\_PSP\_MemRanges, 1811  
CFE\_PSP\_MemRangeSet, 1811  
CFE\_PSP\_MemValidateRange, 1812  
CFE\_PSP\_MemRangeGet  
  cfe\_psp\_memrange\_api.h, 1810  
CFE\_PSP\_MemRanges  
  cfe\_psp\_memrange\_api.h, 1811  
CFE\_PSP\_MemRangeSet  
  cfe\_psp\_memrange\_api.h, 1811  
CFE\_PSP\_MemRead16  
  cfe\_psp\_memaccess\_api.h, 1804  
CFE\_PSP\_MemRead32  
  cfe\_psp\_memaccess\_api.h, 1805  
CFE\_PSP\_MemRead8  
  cfe\_psp\_memaccess\_api.h, 1805  
CFE\_PSP\_MemSet  
  cfe\_psp\_memaccess\_api.h, 1806  
CFE\_PSP\_MemValidateRange  
  cfe\_psp\_memrange\_api.h, 1812  
CFE\_PSP\_MemWrite16  
  cfe\_psp\_memaccess\_api.h, 1806  
CFE\_PSP\_MemWrite32  
  cfe\_psp\_memaccess\_api.h, 1806  
CFE\_PSP\_MemWrite8  
  cfe\_psp\_memaccess\_api.h, 1807  
CFE\_PSP\_NO\_EXCEPTION\_DATA  
  cfe\_psp\_error.h, 1799  
CFE\_PSP\_Panic  
  cfe\_psp\_watchdog\_api.h, 1823  
CFE\_PSP\_PANIC\_CORE\_APP  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_PANIC\_GENERAL\_FAILURE  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_PANIC\_MEMORY\_ALLOC  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_PANIC\_NONVOL\_DISK  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_PANIC\_STARTUP  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_PANIC\_STARTUP\_SEM  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_PANIC\_VOLATILE\_DISK  
  cfe\_psp\_watchdog\_api.h, 1821  
cfe\_psp\_port\_api.h  
  CFE\_PSP\_PortRead16, 1813  
  CFE\_PSP\_PortRead32, 1814  
  CFE\_PSP\_PortRead8, 1814  
  CFE\_PSP\_PortWrite16, 1814  
  CFE\_PSP\_PortWrite32, 1815  
  CFE\_PSP\_PortWrite8, 1815  
CFE\_PSP\_PortRead16  
  cfe\_psp\_port\_api.h, 1813  
CFE\_PSP\_PortRead32  
  cfe\_psp\_port\_api.h, 1814  
  cfe\_psp\_port\_api.h, 1814  
CFE\_PSP\_PortRead8  
  cfe\_psp\_port\_api.h, 1814  
CFE\_PSP\_PortWrite16  
  cfe\_psp\_port\_api.h, 1814  
  cfe\_psp\_port\_api.h, 1814  
CFE\_PSP\_PortWrite32  
  cfe\_psp\_port\_api.h, 1815  
CFE\_PSP\_PortWrite8  
  cfe\_psp\_port\_api.h, 1815  
CFE\_PSP\_ReadFromCDS  
  cfe\_psp\_cds\_api.h, 1792  
CFE\_PSP\_Restart  
  cfe\_psp\_watchdog\_api.h, 1823  
CFE\_PSP\_RST\_SUBTYPE\_BANKSWITCH\_RESET  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION  
  cfe\_psp\_watchdog\_api.h, 1821  
CFE\_PSP\_RST\_SUBTYPE\_HW\_SPECIAL\_COMMAND  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_SUBTYPE\_HW\_WATCHDOG  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_SUBTYPE\_HWDEBUG\_RESET  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_SUBTYPE\_MAX  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_SUBTYPE\_POWER\_CYCLE  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_SUBTYPE\_PUSH\_BUTTON  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_SUBTYPE\_RESET\_COMMAND  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_TYPE\_MAX  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_TYPE\_POWERON  
  cfe\_psp\_watchdog\_api.h, 1822  
CFE\_PSP\_RST\_TYPE\_PROCESSOR  
  cfe\_psp\_watchdog\_api.h, 1823  
CFE\_PSP\_SetDefaultExceptionEnvironment  
  cfe\_psp\_exception\_api.h, 1802  
CFE\_PSP\_SOFT\_TIMEBASE\_NAME  
  cfe\_psp\_timertick\_api.h, 1817  
cfe\_psp\_ssr\_api.h  
  CFE\_PSP\_InitSSR, 1816  
CFE\_PSP\_STATUS\_C  
  cfe\_psp\_error.h, 1800  
CFE\_PSP\_STATUS\_STRING\_LENGTH  
  cfe\_psp\_error.h, 1800  
CFE\_PSP\_Status\_t  
  cfe\_psp\_error.h, 1800  
CFE\_PSP\_StatusString\_t  
  cfe\_psp\_error.h, 1800  
CFE\_PSP\_StatusToString

cfe\_psp\_error.h, 1800  
 CFE\_PSP\_SUCCESS  
     cfe\_psp\_error.h, 1800  
 cfe\_psp\_timetick\_api.h  
     CFE\_PSP\_Get\_Timebase, 1817  
     CFE\_PSP\_GetTime, 1817  
     CFE\_PSP\_GetTimerLow32Rollover, 1818  
     CFE\_PSP\_GetTimerTicksPerSecond, 1818  
     CFE\_PSP\_SOFT\_TIMEBASE\_NAME, 1817  
 cfe\_psp\_version\_api.h  
     CFE\_PSP\_GetBuildNumber, 1819  
     CFE\_PSP\_GetVersionCodeName, 1819  
     CFE\_PSP\_GetVersionNumber, 1819  
     CFE\_PSP\_GetVersionString, 1819  
 cfe\_psp\_watchdog\_api.h  
     CFE\_PSP\_GetRestartType, 1823  
     CFE\_PSP\_Panic, 1823  
     CFE\_PSP\_PANIC\_CORE\_APP, 1821  
     CFE\_PSP\_PANIC\_GENERAL\_FAILURE, 1821  
     CFE\_PSP\_PANIC\_MEMORY\_ALLOC, 1821  
     CFE\_PSP\_PANIC\_NONVOL\_DISK, 1821  
     CFE\_PSP\_PANIC\_STARTUP, 1821  
     CFE\_PSP\_PANIC\_STARTUP\_SEM, 1821  
     CFE\_PSP\_PANIC\_VOLATILE\_DISK, 1821  
     CFE\_PSP\_Restart, 1823  
     CFE\_PSP\_RST\_SUBTYPE\_BANKSWITCH\_RESET, 1821  
     CFE\_PSP\_RST\_SUBTYPE\_EXCEPTION, 1821  
     CFE\_PSP\_RST\_SUBTYPE\_HW\_SPECIAL\_COMMAND, 1822  
     CFE\_PSP\_RST\_SUBTYPE\_HW\_WATCHDOG, 1822  
     CFE\_PSP\_RST\_SUBTYPE\_HWDEBUG\_RESET, 1822  
     CFE\_PSP\_RST\_SUBTYPE\_MAX, 1822  
     CFE\_PSP\_RST\_SUBTYPE\_POWER\_CYCLE, 1822  
     CFE\_PSP\_RST\_SUBTYPE\_PUSH\_BUTTON, 1822  
     CFE\_PSP\_RST\_SUBTYPE\_RESET\_COMMAND, 1822  
     CFE\_PSP\_RST\_SUBTYPE\_UNDEFINED\_RESET, 1822  
     CFE\_PSP\_RST\_TYPE\_MAX, 1822  
     CFE\_PSP\_RST\_TYPE\_POWERON, 1822  
     CFE\_PSP\_RST\_TYPE\_PROCESSOR, 1823  
     CFE\_PSP\_WatchdogDisable, 1824  
     CFE\_PSP\_WatchdogEnable, 1824  
     CFE\_PSP\_WatchdogGet, 1824  
     CFE\_PSP\_WatchdogInit, 1824  
     CFE\_PSP\_WatchdogService, 1824  
     CFE\_PSP\_WatchdogSet, 1824  
 CFE\_PSP\_WatchdogDisable  
     cfe\_psp\_watchdog\_api.h, 1824  
 CFE\_PSP\_WatchdogEnable  
     cfe\_psp\_watchdog\_api.h, 1824  
 CFE\_PSP\_WatchdogGet  
     cfe\_psp\_watchdog\_api.h, 1824  
 CFE\_PSP\_WatchdogInit  
     cfe\_psp\_watchdog\_api.h, 1824  
 CFE\_PSP\_WatchdogService  
     cfe\_psp\_watchdog\_api.h, 1824  
 CFE\_PSP\_WatchdogSet  
     cfe\_psp\_watchdog\_api.h, 1824  
 CFE\_PSP\_WriteToCDS  
     cfe\_psp\_cds\_api.h, 1793  
 cfe\_resourceid.h  
     CFE\_Resourceld\_CheckFunc\_t, 1377  
     CFE\_Resourceld\_Equal, 1378  
     CFE\_Resourceld\_FindNext, 1378  
     CFE\_Resourceld\_FindNextEx, 1379  
     CFE\_Resourceld\_FromInteger, 1379  
     CFE\_Resourceld\_GetBase, 1380  
     CFE\_Resourceld\_GetSerial, 1380  
     CFE\_Resourceld\_IncrementFunc\_t, 1377  
     CFE\_Resourceld\_IsDefined, 1380  
     CFE\_RESOURCEID\_TEST\_DEFINED, 1376  
     CFE\_RESOURCEID\_TEST\_EQUAL, 1377  
     CFE\_RESOURCEID\_TO ULONG, 1377  
     CFE\_Resourceld\_ToIndex, 1381  
     CFE\_Resourceld\_ToInteger, 1381  
     cfe\_resourceid\_api\_typedefs.h  
         CFE\_RESOURCEID\_RESERVED, 1383  
         CFE\_RESOURCEID\_UNDEFINED, 1383  
     cfe\_resourceid\_basevalue.h  
         CFE\_RESOURCEID\_MAKE\_BASE, 1566  
         CFE\_RESOURCEID\_MAX, 1566  
         CFE\_RESOURCEID\_SHIFT, 1566  
     CFE\_Resourceld\_CheckFunc\_t  
         cfe\_resourceid.h, 1377  
     CFE\_RESOURCEID\_CONFIGID\_BASE\_OFFSET  
         cFE Resource ID base values, 393  
     CFE\_Resourceld\_Equal  
         cfe\_resourceid.h, 1378  
     CFE\_RESOURCEID\_ES\_APPID\_BASE\_OFFSET  
         cFE Resource ID base values, 393  
     CFE\_RESOURCEID\_ES\_CDSBLOCKID\_BASE\_OFFSET  
         cFE Resource ID base values, 393  
     CFE\_RESOURCEID\_ES\_COUNTID\_BASE\_OFFSET  
         cFE Resource ID base values, 393  
     CFE\_RESOURCEID\_ES\_LIBID\_BASE\_OFFSET  
         cFE Resource ID base values, 393  
     CFE\_RESOURCEID\_ES\_POOLID\_BASE\_OFFSET  
         cFE Resource ID base values, 393  
     CFE\_RESOURCEID\_ES\_TASKID\_BASE\_OFFSET  
         cFE Resource ID base values, 393  
     CFE\_Resourceld\_FindNext  
         cfe\_resourceid.h, 1378  
     CFE\_Resourceld\_FindNextEx  
         cfe\_resourceid.h, 1379

CFE\_Resourceld\_FromInteger  
  cfe\_resourceid.h, 1379

CFE\_Resourceld\_GetBase  
  cfe\_resourceid.h, 1380

CFE\_Resourceld\_GetSerial  
  cfe\_resourceid.h, 1380

CFE\_Resourceld\_IncrementFunc\_t  
  cfe\_resourceid.h, 1377

CFE\_Resourceld\_IsDefined  
  cfe\_resourceid.h, 1380

CFE\_RESOURCEID\_MAKE\_BASE  
  cfe\_resourceid\_basevalue.h, 1566

CFE\_RESOURCEID\_MAX  
  cfe\_resourceid\_basevalue.h, 1566

CFE\_RESOURCEID\_RESERVED  
  cfe\_resourceid\_api\_typedefs.h, 1383

CFE\_RESOURCEID\_SB\_PIPEID\_RESOURCE\_BASE\_OFFSET  
  cFE Resource ID base values, 393

CFE\_RESOURCEID\_SHIFT  
  cfe\_resourceid\_basevalue.h, 1566

CFE\_RESOURCEID\_TBL\_ACCESSID\_BASE\_OFFSET  
  cFE Resource ID base values, 393

CFE\_RESOURCEID\_TBL\_DUMPCTRLID\_BASE\_OFFSET  
  cFE Resource ID base values, 393

CFE\_RESOURCEID\_TBL\_LOADBUFFID\_BASE\_OFFSET  
  cFE Resource ID base values, 393

CFE\_RESOURCEID\_TBL\_REGID\_BASE\_OFFSET  
  cFE Resource ID base values, 393

CFE\_RESOURCEID\_TBL\_VALRESULTID\_BASE\_OFFSET  
  cFE Resource ID base values, 393

CFE\_RESOURCEID\_TEST\_DEFINED  
  cfe\_resourceid.h, 1376

CFE\_RESOURCEID\_TEST\_EQUAL  
  cfe\_resourceid.h, 1377

CFE\_RESOURCEID\_TO ULONG  
  cfe\_resourceid.h, 1377

CFE\_Resourceld\_ToIndex  
  cfe\_resourceid.h, 1381

CFE\_Resourceld\_ToInteger  
  cfe\_resourceid.h, 1381

CFE\_RESOURCEID\_UNDEFINED  
  cfe\_resourceid\_api\_typedefs.h, 1383

CFE\_REVISION  
  cfe\_version.h, 1400

cfe\_sb.h

- CFE\_BIT, 1385
- CFE\_CLR, 1385
- CFE\_SET, 1385
- CFE\_TST, 1385

CFE\_SB\_AllocateMessageBuffer  
  cFE Zero Copy APIs, 346

CFE\_SB\_ALLSUBS\_TLM\_MID  
  default\_cfe\_sb\_msgids.h, 1574

CFE\_SB\_AllSubscriptionsTlm, 704

Payload, 705

TelemetryHeader, 705

CFE\_SB\_AllSubscriptionsTlm\_Payload, 705

- Entries, 705
- Entry, 705
- PktSegment, 706
- TotalSegments, 706

CFE\_SB\_AllSubscriptionsTlm\_Payload\_t  
  default\_cfe\_sb\_msgdefs.h, 1572

CFE\_SB\_AllSubscriptionsTlm\_t  
  default\_cfe\_sb\_msgstruct.h, 1576

cfe\_sb\_api\_typedefs.h

- CFE\_SB\_Buffer\_t, 1389
- CFE\_SB\_DEFAULT\_QOS, 1387
- CFE\_SB\_INVALID\_MSG\_ID, 1387
- CFE\_SB\_INVALID\_PIPE, 1387
- CFE\_SB\_MSGID\_C, 1387
- CFE\_SB\_MSGID\_RESERVED, 1387
- CFE\_SB\_MSGID\_UNWRAP\_VALUE, 1387
- CFE\_SB\_MSGID\_WRAP\_VALUE, 1388
- CFE\_SB\_PEND\_FOREVER, 1388
- CFE\_SB\_PIPEID\_C, 1388
- CFE\_SB\_POLL, 1388
- CFE\_SB\_SUBSCRIPTION, 1388
- CFE\_SB\_UNSUBSCRIPTION, 1388
- CFE\_SB\_BAD\_ARGUMENT
- cFE Return Code Defines, 241
- CFE\_SB\_BAD\_CMD\_CODE\_EID
- cfe\_sb\_eventids.h, 1580
- CFE\_SB\_BAD\_MSGID\_EID
- cfe\_sb\_eventids.h, 1580
- CFE\_SB\_BAD\_PIPEID\_EID
- cfe\_sb\_eventids.h, 1580
- CFE\_SB\_BUF\_ALOC\_ERR
- cFE Return Code Defines, 241
- CFE\_SB\_BUFFER\_INVALID
- cFE Return Code Defines, 242
- CFE\_SB\_Buffer\_t
- cfe\_sb\_api\_typedefs.h, 1389
- CFE\_SB\_CCVAL
- default\_cfe\_sb\_fcncode\_values.h, 1568
- CFE\_SB\_CMD0\_RCVD\_EID
- cfe\_sb\_eventids.h, 1581
- CFE\_SB\_CMD1\_RCVD\_EID
- cfe\_sb\_eventids.h, 1581
- CFE\_SB\_CMD\_MID
- default\_cfe\_sb\_msgids.h, 1574
- CFE\_SB\_CmdTopicIdToMsgId
- cFE Message ID APIs, 352
- CFE\_SB\_CR\_PIPE\_BAD\_ARG\_EID
- cfe\_sb\_eventids.h, 1581
- CFE\_SB\_CR\_PIPE\_ERR\_EID
- cfe\_sb\_eventids.h, 1581
- CFE\_SB\_CR\_PIPE\_NAME\_TAKEN\_EID

cfe\_sb\_eventids.h, 1582  
 CFE\_SB\_CR\_PIPE\_NO\_FREE\_EID  
     cfe\_sb\_eventids.h, 1582  
 CFE\_SB\_CreatePipe  
     cFE Pipe Management APIs, 334  
 CFE\_SB\_DEFAULT\_QOS  
     cfe\_sb\_api\_typedefs.h, 1387  
 CFE\_SB\_DEL\_PIPE\_ERR1\_EID  
     cfe\_sb\_eventids.h, 1582  
 CFE\_SB\_DEL\_PIPE\_ERR2\_EID  
     cfe\_sb\_eventids.h, 1582  
 CFE\_SB\_DeletePipe  
     cFE Pipe Management APIs, 335  
 CFE\_SB\_DEST\_BLK\_ERR\_EID  
     cfe\_sb\_eventids.h, 1583  
 CFE\_SB\_DISABLE\_ROUTE\_CC  
     cfe\_sb\_fcncodes.h, 1597  
 CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC  
     cfe\_sb\_fcncodes.h, 1598  
 CFE\_SB\_DisableRouteCmd, 706  
     CommandHeader, 706  
     Payload, 706  
 CFE\_SB\_DisableRouteCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1576  
 CFE\_SB\_DisableSubReportingCmd, 706  
     CommandHeader, 707  
 CFE\_SB\_DisableSubReportingCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1576  
 CFE\_SB\_DSBL RTE1\_EID  
     cfe\_sb\_eventids.h, 1583  
 CFE\_SB\_DSBL RTE2\_EID  
     cfe\_sb\_eventids.h, 1583  
 CFE\_SB\_DSBL RTE3\_EID  
     cfe\_sb\_eventids.h, 1583  
 CFE\_SB\_DUP\_SUBSCRIP\_EID  
     cfe\_sb\_eventids.h, 1584  
 CFE\_SB\_ENABLE\_ROUTE\_CC  
     cfe\_sb\_fcncodes.h, 1599  
 CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC  
     cfe\_sb\_fcncodes.h, 1599  
 CFE\_SB\_EnableRouteCmd, 707  
     CommandHeader, 707  
     Payload, 707  
 CFE\_SB\_EnableRouteCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1576  
 CFE\_SB\_EnableSubReportingCmd, 707  
     CommandHeader, 708  
 CFE\_SB\_EnableSubReportingCmd\_t  
     default\_cfe\_sb\_msgstruct.h, 1576  
 CFE\_SB\_ENBL RTE1\_EID  
     cfe\_sb\_eventids.h, 1584  
 CFE\_SB\_ENBL RTE2\_EID  
     cfe\_sb\_eventids.h, 1584  
 CFE\_SB\_ENBL RTE3\_EID  
     cfe\_sb\_eventids.h, 1584  
 cfe\_sb\_eventids.h, 1584  
 CFE\_SB\_BAD\_CMD\_CODE\_EID, 1580  
 CFE\_SB\_BAD\_MSGID\_EID, 1580  
 CFE\_SB\_BAD\_PIPEID\_EID, 1580  
 CFE\_SB\_CMD0\_RCVD\_EID, 1581  
 CFE\_SB\_CMD1\_RCVD\_EID, 1581  
 CFE\_SB\_CR\_PIPE\_BAD\_ARG\_EID, 1581  
 CFE\_SB\_CR\_PIPE\_ERR\_EID, 1581  
 CFE\_SB\_CR\_PIPE\_NAME\_TAKEN\_EID, 1582  
 CFE\_SB\_CR\_PIPE\_NO\_FREE\_EID, 1582  
 CFE\_SB\_DEL\_PIPE\_ERR1\_EID, 1582  
 CFE\_SB\_DEL\_PIPE\_ERR2\_EID, 1582  
 CFE\_SB\_DEST\_BLK\_ERR\_EID, 1583  
 CFE\_SB\_DSBL RTE1\_EID, 1583  
 CFE\_SB\_DSBL RTE2\_EID, 1583  
 CFE\_SB\_DSBL RTE3\_EID, 1583  
 CFE\_SB\_DUP\_SUBSCRIP\_EID, 1584  
 CFE\_SB\_ENBL RTE1\_EID, 1584  
 CFE\_SB\_ENBL RTE2\_EID, 1584  
 CFE\_SB\_ENBL RTE3\_EID, 1584  
 CFE\_SB\_FILEWRITE\_ERR\_EID, 1585  
 CFE\_SB\_FULL\_SUB\_PKT\_EID, 1585  
 CFE\_SB\_GET\_BUF\_ERR\_EID, 1585  
 CFE\_SB\_GETPIPEIDBYNAME\_EID, 1585  
 CFE\_SB\_GETPIPEIDBYNAME\_NAME\_ERR\_EID, 1586  
 CFE\_SB\_GETPIPEIDBYNAME\_NULL\_ERR\_EID, 1586  
 CFE\_SB\_GETPIPENAME\_EID, 1586  
 CFE\_SB\_GETPIPENAME\_ID\_ERR\_EID, 1586  
 CFE\_SB\_GETPIPENAME\_NULL\_PTR\_EID, 1587  
 CFE\_SB\_GETPIPEOPTS\_EID, 1587  
 CFE\_SB\_GETPIPEOPTS\_ID\_ERR\_EID, 1587  
 CFE\_SB\_GETPIPEOPTS\_PTR\_ERR\_EID, 1587  
 CFE\_SB\_HASHCOLLISION\_EID, 1588  
 CFE\_SB\_INIT\_EID, 1588  
 CFE\_SB\_LEN\_ERR\_EID, 1588  
 CFE\_SB\_MAX\_DESTS\_MET\_EID, 1588  
 CFE\_SB\_MAX\_MSGS\_MET\_EID, 1589  
 CFE\_SB\_MAX\_PIPES\_MET\_EID, 1589  
 CFE\_SB\_MSG\_TOO\_BIG\_EID, 1589  
 CFE\_SB\_MSGID\_LIM\_ERR\_EID, 1589  
 CFE\_SB\_PART\_SUB\_PKT\_EID, 1590  
 CFE\_SB\_PIPE\_ADDED\_EID, 1590  
 CFE\_SB\_PIPE\_DELETED\_EID, 1590  
 CFE\_SB\_Q\_FULL\_ERR\_EID, 1590  
 CFE\_SB\_Q\_RD\_ERR\_EID, 1591  
 CFE\_SB\_Q\_WR\_ERR\_EID, 1591  
 CFE\_SB\_RCV\_BAD\_ARG\_EID, 1591  
 CFE\_SB\_RCV\_MESSAGE\_INTEGRITY\_FAIL\_EID, 1591  
 CFE\_SB\_SEND\_BAD\_ARG\_EID, 1592  
 CFE\_SB\_SEND\_INV\_MSGID\_EID, 1592

CFE\_SB\_SEND\_MESSAGE\_INTEGRITY\_FAIL\_EID, 1592  
CFE\_SB\_SEND\_NO\_SUBS\_EID, 1592  
CFE\_SB\_SETPPIPEOPTS\_EID, 1593  
CFE\_SB\_SETPPIPEOPTS\_ID\_ERR\_EID, 1593  
CFE\_SB\_SETPPIPEOPTS\_OWNER\_ERR\_EID, 1593  
CFE\_SB SND RTG\_EID, 1593  
CFE\_SB SND RTG\_ERR1\_EID, 1594  
CFE\_SB SND STATS\_EID, 1594  
CFE\_SB SUB ARG\_ERR\_EID, 1594  
CFE\_SB SUB INV CALLER\_EID, 1594  
CFE\_SB SUB INV PIPE\_EID, 1595  
CFE\_SB SUBSCRIPTION RCVD\_EID, 1595  
CFE\_SB SUBSCRIPTION REMOVED\_EID, 1595  
CFE\_SB SUBSCRIPTION RPT\_EID, 1595  
CFE\_SB UNSUB ARG\_ERR\_EID, 1596  
CFE\_SB UNSUB INV CALLER\_EID, 1596  
CFE\_SB UNSUB INV PIPE\_EID, 1596  
CFE\_SB UNSUB NO SUBS\_EID, 1596  
cfe\_sb\_fcncodes.h  
    CFE\_SB\_DISABLE\_ROUTE\_CC, 1597  
    CFE\_SB\_DISABLE\_SUB\_REPORTING\_CC, 1598  
    CFE\_SB\_ENABLE\_ROUTE\_CC, 1599  
    CFE\_SB\_ENABLE\_SUB\_REPORTING\_CC, 1599  
    CFE\_SB\_NOOP\_CC, 1600  
    CFE\_SB\_RESET\_COUNTERS\_CC, 1601  
    CFE\_SB\_SEND\_PREV\_SUBS\_CC, 1602  
    CFE\_SB\_SEND\_SB\_STATS\_CC, 1602  
    CFE\_SB\_WRITE\_MAP\_INFO\_CC, 1603  
    CFE\_SB\_WRITE\_PIPE\_INFO\_CC, 1604  
    CFE\_SB\_WRITE\_ROUTING\_INFO\_CC, 1605  
CFE\_SB\_FILEWRITE\_ERR\_EID  
    cfe\_sb\_eventids.h, 1585  
CFE\_SB\_FULL\_SUB\_PKT\_EID  
    cfe\_sb\_eventids.h, 1585  
CFE\_SB\_FunctionCode\_  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_DISABLE\_ROUTE  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_DISABLE\_SUB\_REPORTING  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_ENABLE\_ROUTE  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_ENABLE\_SUB\_REPORTING  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_NOOP  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_RESET\_COUNTERS  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_SEND\_PREV\_SUBS  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_SEND\_SB\_STATS  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_WRITE\_MAP\_INFO  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
    cfe\_sb\_eventids.h, 1569  
CFE\_SB\_FunctionCode\_WRITE\_PIPE\_INFO  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_FunctionCode\_WRITE\_ROUTING\_INFO  
    default\_cfe\_sb\_fcncode\_values.h, 1569  
CFE\_SB\_GET\_BUF\_ERR\_EID  
    cfe\_sb\_eventids.h, 1585  
CFE\_SB\_GetPipeIdByName  
    cFE Pipe Management APIs, 336  
CFE\_SB\_GETPIPEIDBYNAME\_EID  
    cfe\_sb\_eventids.h, 1585  
CFE\_SB\_GETPIPEIDBYNAME\_NAME\_ERR\_EID  
    cfe\_sb\_eventids.h, 1586  
CFE\_SB\_GETPIPEIDBYNAME\_NULL\_ERR\_EID  
    cfe\_sb\_eventids.h, 1586  
CFE\_SB\_GetPipeName  
    cFE Pipe Management APIs, 336  
CFE\_SB\_GETPIPENAME\_EID  
    cfe\_sb\_eventids.h, 1586  
CFE\_SB\_GETPIPENAME\_ID\_ERR\_EID  
    cfe\_sb\_eventids.h, 1586  
CFE\_SB\_GETPIPENAME\_NULL\_PTR\_EID  
    cfe\_sb\_eventids.h, 1587  
CFE\_SB\_GetPipeOpts  
    cFE Pipe Management APIs, 337  
CFE\_SB\_GETPIPEOPTS\_EID  
    cfe\_sb\_eventids.h, 1587  
CFE\_SB\_GETPIPEOPTS\_ID\_ERR\_EID  
    cfe\_sb\_eventids.h, 1587  
CFE\_SB\_GETPIPEOPTS\_PTR\_ERR\_EID  
    cfe\_sb\_eventids.h, 1587  
CFE\_SB\_GetUserData  
    cFE Message Characteristics APIs, 348  
CFE\_SB\_GetUserDataLength  
    cFE Message Characteristics APIs, 349  
CFE\_SB\_GlobalCmdTopicIdToMsgId  
    cFE Message ID APIs, 353  
CFE\_SB\_GlobalTlmTopicIdToMsgId  
    cFE Message ID APIs, 353  
CFE\_SB\_HASHCOLLISION\_EID  
    cfe\_sb\_eventids.h, 1588  
CFE\_SB\_HK\_TLM\_MID  
    default\_cfe\_sb\_msgids.h, 1574  
CFE\_SB\_HousekeepingTlm, 708  
    Payload, 708  
    TelemetryHeader, 708  
CFE\_SB\_HousekeepingTlm\_Payload, 708  
    CommandCounter, 709  
    CommandErrorCounter, 709  
    CreatePipeErrorCounter, 710  
    DuplicateSubscriptionsCounter, 710  
    GetPipeIdByNameErrorCounter, 710  
    InternalErrorCounter, 710  
    MemInUse, 710

MemPoolHandle, 710  
MsgLimitErrorCounter, 710  
MsgReceiveErrorCounter, 711  
MsgSendErrorCounter, 711  
NoSubscribersCounter, 711  
PipeOptsErrorCounter, 711  
PipeOverflowErrorCounter, 711  
Spare2Align, 711  
SubscribeErrorCounter, 711  
UnmarkedMem, 712  
CFE\_SB\_HousekeepingTlm\_Payload\_t  
  default\_cfe\_sb\_msgdefs.h, 1572  
CFE\_SB\_HousekeepingTlm\_t  
  default\_cfe\_sb\_msgstruct.h, 1576  
CFE\_SB\_INIT\_EID  
  cfe\_sb\_eventids.h, 1588  
cfe\_sb\_interface\_cfg.h  
  CFE\_MISSION\_SB\_MAX\_PIPES, 1607  
  CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE, 1607  
  CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT,  
    1607  
  DEFAULT\_CFE\_MISSION\_SB\_MAX\_PIPES, 1607  
  DEFAULT\_CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE,  
    1608  
  DEFAULT\_CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT,  
    1608  
cfe\_sb\_internal\_cfg.h  
  CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES,  
    1610  
  CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME,  
    1611  
  CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT, 1611  
  CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME,  
    1611  
  CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME,  
    1612  
  CFE\_PLATFORM\_SB\_FILTER\_MASK1, 1612  
  CFE\_PLATFORM\_SB\_FILTER\_MASK2, 1612  
  CFE\_PLATFORM\_SB\_FILTER\_MASK3, 1612  
  CFE\_PLATFORM\_SB\_FILTER\_MASK4, 1612  
  CFE\_PLATFORM\_SB\_FILTER\_MASK5, 1612  
  CFE\_PLATFORM\_SB\_FILTER\_MASK6, 1613  
  CFE\_PLATFORM\_SB\_FILTER\_MASK7, 1613  
  CFE\_PLATFORM\_SB\_FILTER\_MASK8, 1613  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT1, 1613  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT2, 1613  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT3, 1613  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT4, 1613  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT5, 1613  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT6, 1614  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT7, 1614  
  CFE\_PLATFORM\_SB\_FILTERED\_EVENT8, 1614  
  CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID,  
    1614  
CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE, 1614  
CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT,  
  1614  
CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS, 1615  
CFE\_PLATFORM\_SB\_MAX\_PIPES, 1615  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01,  
  1615  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09,  
  1616  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10,  
  1617  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11,  
  1617  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12,  
  1617  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13,  
  1617  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14,  
  1617  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15,  
  1617  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16,  
  1617  
CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS,  
  1617  
CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY,  
  1618  
CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE,  
  1618  
DEFAULT\_CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES,  
  1618  
DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME,  
  1618  
DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT,  
  1618  
DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME,  
  1619  
DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME,  
  1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK1,

1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK2,  
1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK3,  
1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK4,  
1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK5,  
1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK6,  
1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK7,  
1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK8,  
1619  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT1,  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT2,  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT3,  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT4,  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT5,  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT6,CFE\_SB\_INTERNAL\_ERR  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT7,CFE\_SB\_INVALID\_MSG\_ID  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT8,CFE\_SB\_INVALID\_PIPE  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID,CFE\_SB\_IsValidMsgId  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE,CFE\_SB\_LEN\_ERR\_EID  
1620  
DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PIPE,CFE\_SB\_LocalCmdTopicIdToMsgId  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS,CFE\_SB\_LocalTlmTopicIdToMsgId  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_PIPES,CFE\_SB\_MAX\_DESTS\_MET  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_0,CFE\_SB\_MAX\_DESTS\_MET\_EID  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_0,CFE\_SB\_MAX\_MSGS\_MET  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_0,CFE\_SB\_MAX\_MSGS\_MET\_EID  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_0,CFE\_SB\_MAX\_PIPES\_MET  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_0,CFE\_SB\_MAX\_PIPES\_MET\_EID  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_0,CFE\_SB\_MessageStringGet  
1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_0,CFE\_SB\_MessageStringSet

cFE Message Characteristics APIs, 350  
**CFE\_SB\_Msg**, 712  
  LongDouble, 712  
  LongInt, 712  
  Msg, 712  
**CFE\_SB\_MSG\_TOO\_BIG**  
  cFE Return Code Defines, 242  
**CFE\_SB\_MSG\_TOO\_BIG\_EID**  
  cfe\_sb\_eventids.h, 1589  
**CFE\_SB\_MsgId\_Atom\_t**  
  default\_cfe\_sb\_extern\_typedefs.h, 1567  
**CFE\_SB\_MSGID\_C**  
  cfe\_sb\_api\_typedefs.h, 1387  
**CFE\_SB\_MsgId\_Equal**  
  cFE Message ID APIs, 355  
**CFE\_SB\_MSGID\_LIM\_ERR\_EID**  
  cfe\_sb\_eventids.h, 1589  
**CFE\_SB\_MSGID\_RESERVED**  
  cfe\_sb\_api\_typedefs.h, 1387  
**CFE\_SB\_MsgId\_t**, 713  
  Value, 713  
**CFE\_SB\_MSGID\_UNWRAP\_VALUE**  
  cfe\_sb\_api\_typedefs.h, 1387  
**CFE\_SB\_MSGID\_WRAP\_VALUE**  
  cfe\_sb\_api\_typedefs.h, 1388  
**CFE\_SB\_MsgIdToValue**  
  cFE Message ID APIs, 356  
**CFE\_SB\_MsgMapFileEntry**, 713  
  Index, 714  
  MsgId, 714  
**CFE\_SB\_MsgMapFileEntry\_t**  
  default\_cfe\_sb\_msgdefs.h, 1572  
**CFE\_SB\_NO\_MESSAGE**  
  cFE Return Code Defines, 242  
**CFE\_SB\_NOOP\_CC**  
  cfe\_sb\_fcncodes.h, 1600  
**CFE\_SB\_NoopCmd**, 714  
  CommandHeader, 714  
**CFE\_SB\_NoopCmd\_t**  
  default\_cfe\_sb\_msgstruct.h, 1576  
**CFE\_SB\_NOT\_IMPLEMENTED**  
  cFE Return Code Defines, 243  
**CFE\_SB\_ONESUB\_TLM\_MID**  
  default\_cfe\_sb\_msgids.h, 1574  
**CFE\_SB\_PART\_SUB\_PKT\_EID**  
  cfe\_sb\_eventids.h, 1590  
**CFE\_SB\_PEND\_FOREVER**  
  cfe\_sb\_api\_typedefs.h, 1388  
**CFE\_SB\_PIPE\_ADDED\_EID**  
  cfe\_sb\_eventids.h, 1590  
**CFE\_SB\_PIPE\_CR\_ERR**  
  cFE Return Code Defines, 243  
**CFE\_SB\_PIPE\_DELETED\_EID**  
  cfe\_sb\_eventids.h, 1590  
**CFE\_SB\_PIPE\_RD\_ERR**  
  cFE Return Code Defines, 243  
**CFE\_SB\_PipeDepthStats**, 714  
  CurrentQueueDepth, 715  
  MaxQueueDepth, 715  
  PeakQueueDepth, 715  
  Pipeld, 715  
  Spare, 715  
**CFE\_SB\_PipeDepthStats\_t**  
  default\_cfe\_sb\_msgdefs.h, 1572  
**CFE\_SB\_PIPEID\_BASE**  
  cFE Resource ID base values, 393  
**CFE\_SB\_PIPEID\_C**  
  cfe\_sb\_api\_typedefs.h, 1388  
**CFE\_SB\_Pipeld\_t**  
  default\_cfe\_sb\_extern\_typedefs.h, 1567  
**CFE\_SB\_Pipeld\_ToIndex**  
  cFE Pipe Management APIs, 338  
**CFE\_SB\_PipeInfoEntry**, 716  
  AppId, 716  
  AppName, 716  
  CurrentQueueDepth, 716  
  MaxQueueDepth, 716  
  Opts, 716  
  PeakQueueDepth, 717  
  Pipeld, 717  
  PipeName, 717  
  SendErrors, 717  
  Spare, 717  
**CFE\_SB\_PipeInfoEntry\_t**  
  default\_cfe\_sb\_msgdefs.h, 1572  
**CFE\_SB PIPEOPTS IGNOREMINE**  
  cFE SB Pipe options, 357  
**CFE\_SB\_POLL**  
  cfe\_sb\_api\_typedefs.h, 1388  
**CFE\_SB\_Q\_FULL\_ERR\_EID**  
  cfe\_sb\_eventids.h, 1590  
**CFE\_SB\_Q\_RD\_ERR\_EID**  
  cfe\_sb\_eventids.h, 1591  
**CFE\_SB\_Q\_WR\_ERR\_EID**  
  cfe\_sb\_eventids.h, 1591  
**CFE\_SB\_Qos\_t**, 717  
  Priority, 717  
  Reliability, 718  
**CFE\_SB\_QosPriority**  
  default\_cfe\_sb\_extern\_typedefs.h, 1568  
**CFE\_SB\_QosPriority\_Enum\_t**  
  default\_cfe\_sb\_extern\_typedefs.h, 1567  
**CFE\_SB\_QosPriority\_HIGH**  
  default\_cfe\_sb\_extern\_typedefs.h, 1568  
**CFE\_SB\_QosPriority\_LOW**  
  default\_cfe\_sb\_extern\_typedefs.h, 1568  
**CFE\_SB\_QosReliability**  
  default\_cfe\_sb\_extern\_typedefs.h, 1568

CFE\_SB\_QosReliability\_Enum\_t  
  default\_cfe\_sb\_extern\_typedefs.h, 1567

CFE\_SB\_QosReliability\_HIGH  
  default\_cfe\_sb\_extern\_typedefs.h, 1568

CFE\_SB\_QosReliability\_LOW  
  default\_cfe\_sb\_extern\_typedefs.h, 1568

CFE\_SB\_RCV\_BAD\_ARG\_EID  
  cfe\_sb\_eventids.h, 1591

CFE\_SB\_RCV\_MESSAGE\_INTEGRITY\_FAIL\_EID  
  cfe\_sb\_eventids.h, 1591

CFE\_SB\_ReceiveBuffer  
  cFE Send/Receive Message APIs, 343

CFE\_SB\_ReleaseMessageBuffer  
  cFE Zero Copy APIs, 346

CFE\_SB\_RESET\_COUNTERS\_CC  
  cfe\_sb\_fncodes.h, 1601

CFE\_SB\_ResetCountersCmd, 718  
  CommandHeader, 718

CFE\_SB\_ResetCountersCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1576

CFE\_SB\_RouteCmd\_Payload, 718  
  MsgId, 719  
  Pipe, 719  
  Spare, 719

CFE\_SB\_RouteCmd\_Payload\_t  
  default\_cfe\_sb\_msgdefs.h, 1572

CFE\_SB\_Routeld\_Atom\_t  
  default\_cfe\_sb\_extern\_typedefs.h, 1567

CFE\_SB\_RoutingFileEntry, 719  
  AppName, 719  
  MsgCnt, 720  
  MsgId, 720  
  Pipeld, 720  
  PipeName, 720  
  State, 720

CFE\_SB\_RoutingFileEntry\_t  
  default\_cfe\_sb\_msgdefs.h, 1572

CFE\_SB\_SEND\_BAD\_ARG\_EID  
  cfe\_sb\_eventids.h, 1592

CFE\_SB\_SEND\_HK\_MID  
  default\_cfe\_sb\_msgids.h, 1574

CFE\_SB\_SEND\_INV\_MSGID\_EID  
  cfe\_sb\_eventids.h, 1592

CFE\_SB\_SEND\_MESSAGE\_INTEGRITY\_FAIL\_EID  
  cfe\_sb\_eventids.h, 1592

CFE\_SB\_SEND\_NO\_SUBS\_EID  
  cfe\_sb\_eventids.h, 1592

CFE\_SB\_SEND\_PREV\_SUBS\_CC  
  cfe\_sb\_fncodes.h, 1602

CFE\_SB\_SEND\_SB\_STATS\_CC  
  cfe\_sb\_fncodes.h, 1602

CFE\_SB\_SendHkCmd, 720  
  CommandHeader, 720

CFE\_SB\_SendHkCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1576

CFE\_SB\_SendPrevSubsCmd, 721  
  CommandHeader, 721

CFE\_SB\_SendPrevSubsCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1576

CFE\_SB\_SendSbStatsCmd, 721  
  CommandHeader, 721

CFE\_SB\_SendSbStatsCmd\_t  
  default\_cfe\_sb\_msgstruct.h, 1576

CFE\_SB\_SetPipeOpts  
  cFE Pipe Management APIs, 338

CFE\_SB\_SETPIPEOPTS\_EID  
  cfe\_sb\_eventids.h, 1593

CFE\_SB\_SETPIPEOPTS\_ID\_ERR\_EID  
  cfe\_sb\_eventids.h, 1593

CFE\_SB\_SETPIPEOPTS\_OWNER\_ERR\_EID  
  cfe\_sb\_eventids.h, 1593

CFE\_SB\_SetUserDataLength  
  cFE Message Characteristics APIs, 351

CFE\_SB\_SingleSubscriptionTlm, 721  
  Payload, 722  
  TelemetryHeader, 722

CFE\_SB\_SingleSubscriptionTlm\_Payload, 722  
  MsgId, 722  
  Pipe, 723  
  Qos, 723  
  SubType, 723

CFE\_SB\_SingleSubscriptionTlm\_Payload\_t  
  default\_cfe\_sb\_msgdefs.h, 1573

CFE\_SB\_SingleSubscriptionTlm\_t  
  default\_cfe\_sb\_msgstruct.h, 1576

CFE\_SB SND RTG EID  
  cfe\_sb\_eventids.h, 1593

CFE\_SB SND RTG ERR1 EID  
  cfe\_sb\_eventids.h, 1594

CFE\_SB SND STATS EID  
  cfe\_sb\_eventids.h, 1594

CFE\_SB STATS TLM MID  
  default\_cfe\_sb\_msgids.h, 1575

CFE\_SB\_StatsTlm, 723  
  Payload, 723  
  TelemetryHeader, 723

CFE\_SB\_StatsTlm\_Payload, 724  
  MaxMemAllowed, 724  
  MaxMsgIdsAllowed, 725  
  MaxPipeDepthAllowed, 725  
  MaxPipesAllowed, 725  
  MaxSubscriptionsAllowed, 725  
  MemInUse, 725  
  MsgIdsInUse, 725  
  PeakMemInUse, 725  
  PeakMsgIdsInUse, 726  
  PeakPipesInUse, 726  
  PeakSBBuffersInUse, 726

PeakSubscriptionsInUse, [726](#)  
 PipeDepthStats, [726](#)  
 PipesInUse, [726](#)  
 SBBuffersInUse, [726](#)  
 SubscriptionsInUse, [727](#)  
**CFE\_SB\_StatsTlm\_Payload\_t**  
 default\_cfe\_sb\_msgdefs.h, [1573](#)  
**CFE\_SB\_StatsTlm\_t**  
 default\_cfe\_sb\_msgstruct.h, [1576](#)  
**CFE\_SB\_SUB\_ARG\_ERR\_EID**  
 cfe\_sb\_eventids.h, [1594](#)  
**CFE\_SB\_SUB\_INV\_CALLER\_EID**  
 cfe\_sb\_eventids.h, [1594](#)  
**CFE\_SB\_SUB\_INV\_PIPE\_EID**  
 cfe\_sb\_eventids.h, [1595](#)  
**CFE\_SB\_SUB\_RPT\_CTRL\_MID**  
 default\_cfe\_sb\_msgids.h, [1575](#)  
**CFE\_SB\_SubEntries**, [727](#)  
 MsgId, [727](#)  
 Pipe, [727](#)  
 Qos, [727](#)  
**CFE\_SB\_SubEntries\_t**  
 default\_cfe\_sb\_msgdefs.h, [1573](#)  
**CFE\_SB\_Subscribe**  
 cFE Message Subscription Control APIs, [339](#)  
**CFE\_SB\_SubscribeEx**  
 cFE Message Subscription Control APIs, [340](#)  
**CFE\_SB\_SubscribeLocal**  
 cFE Message Subscription Control APIs, [341](#)  
**CFE\_SB\_SUBSCRIPTION**  
 cfe\_sb\_api\_typedefs.h, [1388](#)  
**CFE\_SB\_SUBSCRIPTION\_RCVD\_EID**  
 cfe\_sb\_eventids.h, [1595](#)  
**CFE\_SB\_SUBSCRIPTION\_REMOVED\_EID**  
 cfe\_sb\_eventids.h, [1595](#)  
**CFE\_SB\_SUBSCRIPTION\_RPT\_EID**  
 cfe\_sb\_eventids.h, [1595](#)  
**CFE\_SB\_TIME\_OUT**  
 cFE Return Code Defines, [243](#)  
**CFE\_SB\_TimeStampMsg**  
 cFE Message Characteristics APIs, [351](#)  
**CFE\_SB\_TlmTopicIdToMsgId**  
 cFE Message ID APIs, [356](#)  
**cfe\_sb\_topicids.h**  
 CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID, [1623](#)  
 CFE\_MISSION\_SB\_CMD\_TOPICID, [1623](#)  
 CFE\_MISSION\_SB\_HK\_TLM\_TOPICID, [1624](#)  
 CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID, [1624](#)  
 CFE\_MISSION\_SB\_SEND\_HK\_TOPICID, [1624](#)  
 CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID, [1624](#)  
 CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID, [1624](#)  
 DEFAULT\_CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID, [1624](#)  
 DEFAULT\_CFE\_MISSION\_SB\_CMD\_TOPICID, [1624](#)  
 DEFAULT\_CFE\_MISSION\_SB\_HK\_TLM\_TOPICID, [1625](#)  
 DEFAULT\_CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID, [1625](#)  
 DEFAULT\_CFE\_MISSION\_SB\_SEND\_HK\_TOPICID, [1625](#)  
 DEFAULT\_CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID, [1625](#)  
 DEFAULT\_CFE\_MISSION\_SB\_SUB\_RPT\_CTRL\_TOPICID, [1625](#)  
**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, [1596](#)  
**CFE\_SB\_UNSUB\_INV\_CALLER\_EID**  
 cfe\_sb\_eventids.h, [1596](#)  
**CFE\_SB\_UNSUB\_INV\_PIPE\_EID**  
 cfe\_sb\_eventids.h, [1596](#)  
**CFE\_SB\_UNSUB\_NO\_SUBS\_EID**  
 cfe\_sb\_eventids.h, [1596](#)  
**CFE\_SB\_Unsubscribe**  
 cFE Message Subscription Control APIs, [342](#)  
**CFE\_SB\_UnsubscribeLocal**  
 cFE Message Subscription Control APIs, [342](#)  
**CFE\_SB\_UNSUBSCRIPTION**  
 cfe\_sb\_api\_typedefs.h, [1388](#)  
**CFE\_SB\_ValueToMsgId**  
 cFE Message ID APIs, [356](#)  
**CFE\_SB\_WRITE\_MAP\_INFO\_CC**  
 cfe\_sb\_fcncodes.h, [1603](#)  
**CFE\_SB\_WRITE\_PIPE\_INFO\_CC**  
 cfe\_sb\_fcncodes.h, [1604](#)  
**CFE\_SB\_WRITE\_ROUTING\_INFO\_CC**  
 cfe\_sb\_fcncodes.h, [1605](#)  
**CFE\_SB\_WriteFileInfoCmd\_Payload**, [728](#)  
 Filename, [728](#)  
**CFE\_SB\_WriteFileInfoCmd\_Payload\_t**  
 default\_cfe\_sb\_msgdefs.h, [1573](#)  
**CFE\_SB\_WriteMapInfoCmd**, [728](#)  
 CommandHeader, [728](#)  
 Payload, [729](#)  
**CFE\_SB\_WriteMapInfoCmd\_t**  
 default\_cfe\_sb\_msgstruct.h, [1577](#)  
**CFE\_SB\_WritePipeInfoCmd**, [729](#)  
 CommandHeader, [729](#)  
 Payload, [729](#)  
**CFE\_SB\_WritePipeInfoCmd\_t**  
 default\_cfe\_sb\_msgstruct.h, [1577](#)

CFE\_SB\_WriteRoutingInfoCmd, 729  
    CommandHeader, 730  
    Payload, 730  
CFE\_SB\_WriteRoutingInfoCmd\_t  
    default\_cfe\_sb\_msgstruct.h, 1577  
CFE\_SB\_WRONG\_MSG\_TYPE  
    cFE Return Code Defines, 243  
CFE\_SERVICE\_BITMASK  
    cfe\_error.h, 1352  
CFE\_SET  
    cfe\_sb.h, 1385  
CFE\_SEVERITY\_BITMASK  
    cfe\_error.h, 1352  
CFE\_SEVERITY\_ERROR  
    cfe\_error.h, 1352  
CFE\_SEVERITY\_INFO  
    cfe\_error.h, 1352  
CFE\_SEVERITY\_SUCCESS  
    cfe\_error.h, 1352  
CFE\_SOFTWARE\_BUS\_SERVICE  
    cfe\_error.h, 1352  
CFE\_SRC\_VERSION  
    cfe\_version.h, 1401  
CFE\_STATUS\_BAD\_COMMAND\_CODE  
    cFE Return Code Defines, 243  
CFE\_STATUS\_C  
    cfe\_error.h, 1352  
CFE\_STATUS\_EXTERNAL\_RESOURCE\_FAIL  
    cFE Return Code Defines, 243  
CFE\_STATUS\_INCORRECT\_STATE  
    cFE Return Code Defines, 244  
CFE\_STATUS\_NO\_COUNTER\_INCREMENT  
    cFE Return Code Defines, 244  
CFE\_STATUS\_NOT\_IMPLEMENTED  
    cFE Return Code Defines, 244  
CFE\_STATUS\_RANGE\_ERROR  
    cFE Return Code Defines, 244  
CFE\_STATUS\_REQUEST\_ALREADY\_PENDING  
    cFE Return Code Defines, 244  
CFE\_STATUS\_STRING\_LENGTH  
    cfe\_error.h, 1352  
CFE\_Status\_t  
    cfe\_error.h, 1353  
CFE\_STATUS\_UNKNOWN\_MSG\_ID  
    cFE Return Code Defines, 244  
CFE\_STATUS\_VALIDATION\_FAILURE  
    cFE Return Code Defines, 244  
CFE\_STATUS\_WRONG\_MSG\_LENGTH  
    cFE Return Code Defines, 244  
CFE\_StatusString\_t  
    cfe\_error.h, 1353  
CFE\_STR  
    cfe\_version.h, 1401  
CFE\_STR\_HELPER  
    cfe\_version.h, 1401  
    cfe\_version.h, 1401  
CFE\_SUCCESS  
    cFE Return Code Defines, 245  
CFE\_TABLE\_SERVICE  
    cfe\_error.h, 1353  
cfe\_tbl.h  
    CFE\_TBL\_HandleFromID, 1390  
    CFE\_TBL\_HandleID\_AsInt, 1390  
    CFE\_TBL\_HandleID\_IsDefined, 1390  
    CFE\_TBL\_HandleID\_IsEqual, 1390  
    CFE\_TBL\_HandleToID, 1390  
CFE\_TBL\_ABORT\_LOAD\_CC  
    cfe\_tbl\_fcncodes.h, 1666  
CFE\_TBL\_AbortLoadCmd, 730  
    CommandHeader, 730  
    Payload, 730  
CFE\_TBL\_AbortLoadCmd\_Payload, 731  
    TableName, 731  
CFE\_TBL\_AbortLoadCmd\_Payload\_t  
    default\_cfe\_tbl\_msgdefs.h, 1632  
CFE\_TBL\_AbortLoadCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1635  
CFE\_TBL\_ACTIVATE\_CC  
    cfe\_tbl\_fcncodes.h, 1666  
CFE\_TBL\_ACTIVATE\_DUMP\_ONLY\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1648  
CFE\_TBL\_ACTIVATE\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1648  
CFE\_TBL\_ActivateCmd, 731  
    CommandHeader, 731  
    Payload, 731  
CFE\_TBL\_ActivateCmd\_Payload, 732  
    TableName, 732  
CFE\_TBL\_ActivateCmd\_Payload\_t  
    default\_cfe\_tbl\_msgdefs.h, 1632  
CFE\_TBL\_ActivateCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1635  
cfe\_tbl\_api\_typedefs.h  
    CFE\_TBL\_BAD\_TABLE\_HANDLE, 1392  
    CFE\_TBL\_CallbackFuncPtr\_t, 1393  
    CFE\_TBL\_HANDLE\_EQ, 1392  
    CFE\_TBL\_HANDLE\_INT, 1392  
    CFE\_TBL\_HANDLE\_IS\_VALID, 1393  
    CFE\_TBL\_Handle\_t, 1393  
    CFE\_TBL\_HANDLEID\_C, 1393  
    CFE\_TBL\_HANDLEID\_UNDEFINED, 1393  
    CFE\_TBL\_Info\_t, 1393  
    CFE\_TBL\_MAX\_FULL\_NAME\_LEN, 1393  
    CFE\_TBL\_REGID\_C, 1393  
    CFE\_TBL\_REGID\_UNDEFINED, 1393  
    CFE\_TBL\_SRC\_ADDRESS, 1394  
    CFE\_TBL\_SRC\_FILE, 1394  
    CFE\_TBL\_SrcEnum, 1394  
    CFE\_TBL\_SrcEnum\_t, 1393

CFE\_TBL\_ASSUMED\_VALID\_INF\_EID  
     cfe\_tbl\_eventids.h, 1648

CFE\_TBL\_BAD\_ARGUMENT  
     cFE Return Code Defines, 245

CFE\_TBL\_BAD\_TABLE\_HANDLE  
     cfe\_tbl\_api\_typedefs.h, 1392

CFE\_TBL\_BufferSelect  
     default\_cfe\_tbl\_extern\_typedefs.h, 1627

CFE\_TBL\_BufferSelect\_ACTIVE  
     default\_cfe\_tbl\_extern\_typedefs.h, 1627

CFE\_TBL\_BufferSelect\_Enum\_t  
     default\_cfe\_tbl\_extern\_typedefs.h, 1626

CFE\_TBL\_BufferSelect\_INACTIVE  
     default\_cfe\_tbl\_extern\_typedefs.h, 1627

CFE\_TBL\_CallbackFuncPtr\_t  
     cfe\_tbl\_api\_typedefs.h, 1393

CFE\_TBL\_CC1\_ERR\_EID  
     cfe\_tbl\_eventids.h, 1648

CFE\_TBL\_CCVAL  
     default\_cfe\_tbl\_fnccode\_values.h, 1627

CFE\_TBL\_CDS\_DELETE\_ERR\_EID  
     cfe\_tbl\_eventids.h, 1649

CFE\_TBL\_CDS\_DELETED\_INFO\_EID  
     cfe\_tbl\_eventids.h, 1649

CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID  
     cfe\_tbl\_eventids.h, 1649

CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID  
     cfe\_tbl\_eventids.h, 1649

CFE\_TBL\_CMD\_MID  
     default\_cfe\_tbl\_msgids.h, 1634

CFE\_TBL\_CODEC\_ERROR\_ERR\_EID  
     cfe\_tbl\_eventids.h, 1650

CFE\_TBL\_CombinedFileHdr, 732  
     Std, 732  
     Tbl, 733

CFE\_TBL\_CombinedFileHdr\_t  
     default\_cfe\_tbl\_extern\_typedefs.h, 1626

CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID  
     cfe\_tbl\_eventids.h, 1650

CFE\_TBL\_DelCDSCmd\_Payload, 733  
     TableName, 733

CFE\_TBL\_DelCDSCmd\_Payload\_t  
     default\_cfe\_tbl\_msgdefs.h, 1632

CFE\_TBL\_DELETE\_CDS\_CC  
     cfe\_tbl\_fnccodes.h, 1667

CFE\_TBL\_DeleteCDSCmd, 733  
     CommandHeader, 734  
     Payload, 734

CFE\_TBL\_DeleteCDSCmd\_t  
     default\_cfe\_tbl\_msgstruct.h, 1635

CFE\_TBL\_DUMP\_CC  
     cfe\_tbl\_fnccodes.h, 1668

CFE\_TBL\_DUMP\_PENDING\_ERR\_EID  
     cfe\_tbl\_eventids.h, 1650

CFE\_TBL\_DUMP\_REGISTRY\_CC  
     cfe\_tbl\_fnccodes.h, 1669

CFE\_TBL\_DumpCmd, 734  
     CommandHeader, 734  
     Payload, 734

CFE\_TBL\_DumpCmd\_Payload, 734  
     ActiveTableFlag, 735  
     DumpFilename, 735  
     TableName, 735

CFE\_TBL\_DumpCmd\_Payload\_t  
     default\_cfe\_tbl\_msgdefs.h, 1632

CFE\_TBL\_DumpCmd\_t  
     default\_cfe\_tbl\_msgstruct.h, 1635

CFE\_TBL\_DUMPCTRLID\_BASE  
     cFE Resource ID base values, 393

CFE\_TBL\_DumpRegistryCmd, 735  
     CommandHeader, 736  
     Payload, 736

CFE\_TBL\_DumpRegistryCmd\_Payload, 736  
     DumpFilename, 736

CFE\_TBL\_DumpRegistryCmd\_Payload\_t  
     default\_cfe\_tbl\_msgdefs.h, 1632

CFE\_TBL\_DumpRegistryCmd\_t  
     default\_cfe\_tbl\_msgstruct.h, 1635

CFE\_TBL\_DumpToBuffer  
     cFE Manage Table Content APIs, 362

CFE\_TBL\_ERR\_ACCESS  
     cFE Return Code Defines, 245

CFE\_TBL\_ERR\_BAD\_CONTENT\_ID  
     cFE Return Code Defines, 245

CFE\_TBL\_ERR\_BAD\_PROCESSOR\_ID  
     cFE Return Code Defines, 245

CFE\_TBL\_ERR\_BAD\_SPACECRAFT\_ID  
     cFE Return Code Defines, 245

CFE\_TBL\_ERR\_BAD\_SUBTYPE\_ID  
     cFE Return Code Defines, 245

CFE\_TBL\_ERR\_DUMP\_ONLY  
     cFE Return Code Defines, 245

CFE\_TBL\_ERR\_DUPLICATE\_DIFF\_SIZE  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_DUPLICATE\_NOT OWNED  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_FILE\_FOR\_WRONG\_TABLE  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_FILE\_SIZE\_INCONSISTENT  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_FILE\_TOO\_LARGE  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_FILENAME\_TOO\_LONG  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_HANDLES\_FULL  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_ILLEGAL\_SRC\_TYPE  
     cFE Return Code Defines, 246

CFE\_TBL\_ERR\_INVALID\_HANDLE  
  cFE Return Code Defines, 247

CFE\_TBL\_ERR\_INVALID\_NAME  
  cFE Return Code Defines, 247

CFE\_TBL\_ERR\_INVALID\_OPTIONS  
  cFE Return Code Defines, 247

CFE\_TBL\_ERR\_INVALID\_SIZE  
  cFE Return Code Defines, 247

CFE\_TBL\_ERR\_LOAD\_IN\_PROGRESS  
  cFE Return Code Defines, 247

CFE\_TBL\_ERR\_LOAD\_INCOMPLETE  
  cFE Return Code Defines, 247

CFE\_TBL\_ERR\_NEVER\_LOADED  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_NO\_ACCESS  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_NO\_BUFFER\_AVAIL  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_NO\_STD\_HEADER  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_NO\_TBL\_HEADER  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_PARTIAL\_LOAD  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_REGISTRY\_FULL  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_SHORT\_FILE  
  cFE Return Code Defines, 248

CFE\_TBL\_ERR\_UNREGISTERED  
  cFE Return Code Defines, 249

cfe\_tbl\_eventids.h

- CFE\_TBL\_ACTIVATE\_DUMP\_ONLY\_ERR\_EID, 1648
- CFE\_TBL\_ACTIVATE\_ERR\_EID, 1648
- CFE\_TBL\_ASSUMED\_VALID\_INF\_EID, 1648
- CFE\_TBL\_CC1\_ERR\_EID, 1648
- CFE\_TBL\_CDS\_DELETE\_ERR\_EID, 1649
- CFE\_TBL\_CDS\_DELETED\_INFO\_EID, 1649
- CFE\_TBL\_CDS\_NOT\_FOUND\_ERR\_EID, 1649
- CFE\_TBL\_CDS\_OWNER\_ACTIVE\_ERR\_EID, 1649
- CFE\_TBL\_CODEC\_ERROR\_ERR\_EID, 1650
- CFE\_TBL\_CREATING\_DUMP\_FILE\_ERR\_EID, 1650
- CFE\_TBL\_DUMP\_PENDING\_ERR\_EID, 1650
- CFE\_TBL\_FAIL\_HK\_SEND\_ERR\_EID, 1650
- CFE\_TBL\_FAIL\_NOTIFY\_SEND\_ERR\_EID, 1651
- CFE\_TBL\_FILE\_ACCESS\_ERR\_EID, 1651
- CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID, 1651
- CFE\_TBL\_FILE\_LOADED\_INF\_EID, 1651
- CFE\_TBL\_FILE\_STD\_HDR\_ERR\_EID, 1652
- CFE\_TBL\_FILE\_SUBTYPE\_ERR\_EID, 1652
- CFE\_TBL\_FILE\_TBL\_HDR\_ERR\_EID, 1652
- CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID, 1652
- CFE\_TBL\_FILE\_TYPE\_ERR\_EID, 1653

CFE\_TBL\_HANDLE\_ACCESS\_ERR\_EID, 1653

CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID, 1653

CFE\_TBL\_IN\_REGISTRY\_ERR\_EID, 1653

CFE\_TBL\_INIT\_INF\_EID, 1654

CFE\_TBL\_LEN\_ERR\_EID, 1654

CFE\_TBL\_LOAD\_ABORT\_ERR\_EID, 1654

CFE\_TBL\_LOAD\_ABORT\_INF\_EID, 1654

CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID, 1655

CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID, 1655

CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID, 1655

CFE\_TBL\_LOAD\_PEND\_REQ\_INF\_EID, 1655

CFE\_TBL\_LOAD\_SUCCESS\_INF\_EID, 1656

CFE\_TBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID, 1656

CFE\_TBL\_LOAD\_TYPE\_ERR\_EID, 1656

CFE\_TBL\_LOAD\_VAL\_ERR\_EID, 1656

CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID, 1657

CFE\_TBL\_MID\_ERR\_EID, 1657

CFE\_TBL\_NO\_INACTIVE\_BUFFER\_ERR\_EID, 1657

CFE\_TBL\_NO\_SUCH\_TABLE\_ERR\_EID, 1657

CFE\_TBL\_NO\_WORK\_BUFFERS\_ERR\_EID, 1658

CFE\_TBL\_NOOP\_INF\_EID, 1658

CFE\_TBL\_NOT\_CRITICAL\_TBL\_ERR\_EID, 1658

CFE\_TBL\_NOT\_IN\_CRIT\_REG\_ERR\_EID, 1658

CFE\_TBL\_OVERWRITE\_DUMP\_INF\_EID, 1659

CFE\_TBL\_OVERWRITE\_REG\_DUMP\_INF\_EID, 1659

CFE\_TBL\_PARTIAL\_LOAD\_ERR\_EID, 1659

CFE\_TBL\_PROCESSOR\_ID\_ERR\_EID, 1659

CFE\_TBL\_REGISTER\_ERR\_EID, 1660

CFE\_TBL\_RESET\_INF\_EID, 1660

CFE\_TBL\_SHARE\_ERR\_EID, 1660

CFE\_TBL\_SPACECRAFT\_ID\_ERR\_EID, 1660

CFE\_TBL\_TLM\_REG\_CMD\_INF\_EID, 1661

CFE\_TBL\_TOO\_MANY\_DUMPS\_ERR\_EID, 1661

CFE\_TBL\_TOO\_MANY\_VALIDATIONS\_ERR\_EID, 1661

CFE\_TBL\_UNREGISTER\_ERR\_EID, 1661

CFE\_TBL\_UNVALIDATED\_ERR\_EID, 1662

CFE\_TBL\_UPDATE\_ERR\_EID, 1662

CFE\_TBL\_UPDATE\_SUCCESS\_INF\_EID, 1662

CFE\_TBL\_VAL\_REQ\_MADE\_INF\_EID, 1662

CFE\_TBL\_VALIDATION\_ERR\_EID, 1663

CFE\_TBL\_VALIDATION\_INF\_EID, 1663

CFE\_TBL\_WRITE\_CFE\_HDR\_ERR\_EID, 1663

CFE\_TBL\_WRITE\_DUMP\_INF\_EID, 1663

CFE\_TBL\_WRITE\_REG\_DUMP\_INF\_EID, 1664

CFE\_TBL\_WRITE\_TBL\_HDR\_ERR\_EID, 1664

CFE\_TBL\_WRITE\_TBL\_IMG\_ERR\_EID, 1664

CFE\_TBL\_WRITE\_TBL\_REG\_ERR\_EID, 1664

CFE\_TBL\_ZERO\_LENGTH\_LOAD\_ERR\_EID, [1665](#)  
 CFE\_TBL\_FAIL\_HK\_SEND\_ERR\_EID  
 cfe\_tbl\_eventids.h, [1650](#)  
 CFE\_TBL\_FAIL\_NOTIFY\_SEND\_ERR\_EID  
 cfe\_tbl\_eventids.h, [1651](#)  
 cfe\_tbl\_fcncodes.h  
     CFE\_TBL\_ABORT\_LOAD\_CC, [1666](#)  
     CFE\_TBL\_ACTIVATE\_CC, [1666](#)  
     CFE\_TBL\_DELETE\_CDS\_CC, [1667](#)  
     CFE\_TBL\_DUMP\_CC, [1668](#)  
     CFE\_TBL\_DUMP\_REGISTRY\_CC, [1669](#)  
     CFE\_TBL\_LOAD\_CC, [1670](#)  
     CFE\_TBL\_NOOP\_CC, [1671](#)  
     CFE\_TBL\_RESET\_COUNTERS\_CC, [1672](#)  
     CFE\_TBL\_SEND\_REGISTRY\_CC, [1672](#)  
     CFE\_TBL\_VALIDATE\_CC, [1673](#)  
 CFE\_TBL\_FILE\_ACCESS\_ERR\_EID  
 cfe\_tbl\_eventids.h, [1651](#)  
 CFE\_TBL\_File\_Hdr, [736](#)  
     NumBytes, [737](#)  
     Offset, [737](#)  
     Reserved, [737](#)  
     TableName, [737](#)  
 CFE\_TBL\_File\_Hdr\_t  
     default\_cfe\_tbl\_extern\_typedefs.h, [1626](#)  
 CFE\_TBL\_FILE\_INCOMPLETE\_ERR\_EID  
     cfe\_tbl\_eventids.h, [1651](#)  
 CFE\_TBL\_FILE\_LOADED\_INF\_EID  
     cfe\_tbl\_eventids.h, [1651](#)  
 CFE\_TBL\_FILE\_STD\_HDR\_ERR\_EID  
     cfe\_tbl\_eventids.h, [1652](#)  
 CFE\_TBL\_FILE\_SUBTYPE\_ERR\_EID  
     cfe\_tbl\_eventids.h, [1652](#)  
 CFE\_TBL\_FILE\_TBL\_HDR\_ERR\_EID  
     cfe\_tbl\_eventids.h, [1652](#)  
 CFE\_TBL\_FILE\_TOO\_BIG\_ERR\_EID  
     cfe\_tbl\_eventids.h, [1652](#)  
 CFE\_TBL\_FILE\_TYPE\_ERR\_EID  
     cfe\_tbl\_eventids.h, [1653](#)  
 CFE\_TBL\_FILEDEF  
     cfe\_tbl\_filedef.h, [1395](#)  
 CFE\_TBL\_FileDef, [737](#)  
     Description, [738](#)  
     ObjectName, [738](#)  
     ObjectSize, [738](#)  
     TableName, [738](#)  
     TgtFilename, [739](#)  
 cfe\_tbl\_filedef.h  
     CFE\_TBL\_FILEDEF, [1395](#)  
     CFE\_TBL\_FileDef\_t, [1395](#)  
 CFE\_TBL\_FileDef\_t  
     cfe\_tbl\_filedef.h, [1395](#)  
 CFE\_TBL\_FunctionCode\_  
     default\_cfe\_tbl\_fcncode\_values.h, [1627](#)  
         CFE\_TBL\_FunctionCode\_ABORT\_LOAD  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_ACTIVATE  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_DELETE\_CDS  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_DUMP  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_DUMP\_REGISTRY  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_LOAD  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_NOOP  
             default\_cfe\_tbl\_fcncode\_values.h, [1627](#)  
         CFE\_TBL\_FunctionCode\_RESET\_COUNTERS  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_SEND\_REGISTRY  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
         CFE\_TBL\_FunctionCode\_VALIDATE  
             default\_cfe\_tbl\_fcncode\_values.h, [1628](#)  
 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, [372](#)  
 CFE\_TBL\_GetStatus  
     cFE Get Table Information APIs, [373](#)  
 CFE\_TBL\_HANDLE\_ACCESS\_ERR\_EID  
     cfe\_tbl\_eventids.h, [1653](#)  
 CFE\_TBL\_HANDLE\_BASE  
     cFE Resource ID base values, [393](#)  
 CFE\_TBL\_HANDLE\_EQ  
     cfe\_tbl\_api\_typedefs.h, [1392](#)  
 CFE\_TBL\_HANDLE\_INT  
     cfe\_tbl\_api\_typedefs.h, [1392](#)  
 CFE\_TBL\_HANDLE\_IS\_VALID  
     cfe\_tbl\_api\_typedefs.h, [1393](#)  
 CFE\_TBL\_Handle\_t  
     cfe\_tbl\_api\_typedefs.h, [1393](#)  
 CFE\_TBL\_HandleFromID  
     cfe\_tbl.h, [1390](#)  
 CFE\_TBL\_HandleID\_AsInt  
     cfe\_tbl.h, [1390](#)  
 CFE\_TBL\_HANDLEID\_C  
     cfe\_tbl\_api\_typedefs.h, [1393](#)  
 CFE\_TBL\_HandleID\_IsDefined  
     cfe\_tbl.h, [1390](#)  
 CFE\_TBL\_HandleID\_IsEqual  
     cfe\_tbl.h, [1390](#)  
 CFE\_TBL\_HandleId\_t  
     default\_cfe\_tbl\_extern\_typedefs.h, [1626](#)  
 CFE\_TBL\_HANDLEID\_UNDEFINED  
     cfe\_tbl\_api\_typedefs.h, [1393](#)

CFE\_TBL\_HandleTolD  
  cfe\_tbl.h, 1390

CFE\_TBL\_HK\_TLM\_MID  
  default\_cfe\_tbl\_msgids.h, 1634

CFE\_TBL\_HousekeepingTlm, 739  
  Payload, 739  
  TelemetryHeader, 739

CFE\_TBL\_HousekeepingTlm\_Payload, 739  
  ActiveBuffer, 741  
  ByteAlignPad1, 741  
  CommandCounter, 741  
  CommandErrorCounter, 741  
  FailedValCounter, 741  
  LastFileDumped, 741  
  LastFileLoaded, 741  
  LastTableLoaded, 741  
  LastUpdatedTable, 742  
  LastUpdateTime, 742  
  LastValCrc, 742  
  LastValStatus, 742  
  LastValTableName, 742  
  MemPoolHandle, 742  
  NumFreeSharedBufs, 742  
  NumLoadPending, 743  
  NumTables, 743  
  NumValRequests, 743  
  SuccessValCounter, 743  
  ValidationCounter, 743

CFE\_TBL\_HousekeepingTlm\_Payload\_t  
  default\_cfe\_tbl\_msgdefs.h, 1632

CFE\_TBL\_HousekeepingTlm\_t  
  default\_cfe\_tbl\_msgstruct.h, 1635

CFE\_TBL\_ILLEGAL\_BUFF\_PARAM\_ERR\_EID  
  cfe\_tbl\_eventids.h, 1653

CFE\_TBL\_IN\_REGISTRY\_ERR\_EID  
  cfe\_tbl\_eventids.h, 1653

CFE\_TBL\_Info, 743  
  Crc, 744  
  Critical, 744  
  DoubleBuffered, 744  
  DumpOnly, 744  
  FileTime, 745  
  LastFileLoaded, 745  
  NumUsers, 745  
  Size, 745  
  TableLoadedOnce, 745  
  TimeOfLastUpdate, 745  
  UserDefAddr, 745

CFE\_TBL\_INFO\_DUMP\_PENDING  
  cFE Return Code Defines, 249

CFE\_TBL\_INFO\_NO\_DUMP\_PENDING  
  cFE Return Code Defines, 249

CFE\_TBL\_INFO\_NO\_UPDATE\_PENDING  
  cFE Return Code Defines, 249

CFE\_TBL\_INFO\_NO\_VALIDATION\_PENDING  
  cFE Return Code Defines, 249

CFE\_TBL\_INFO\_RECOVERED\_TBL  
  cFE Return Code Defines, 249

CFE\_TBL\_Info\_t  
  cfe\_tbl\_api\_typedefs.h, 1393

CFE\_TBL\_INFO\_TABLE\_LOCKED  
  cFE Return Code Defines, 249

CFE\_TBL\_INFO\_UPDATE\_PENDING  
  cFE Return Code Defines, 249

CFE\_TBL\_INFO\_UPDATED  
  cFE Return Code Defines, 250

CFE\_TBL\_INFO\_VALIDATION\_PENDING  
  cFE Return Code Defines, 250

CFE\_TBL\_INIT\_INF\_EID  
  cfe\_tbl\_eventids.h, 1654

cfe\_tbl\_interface\_cfg.h  
  CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN,  
    1675  
  CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, 1675  
  DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN,  
    1676  
  DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH,  
    1676

cfe\_tbl\_internal\_cfg.h  
  CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES,  
    1677  
  CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE,  
    1678  
  CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES,  
    1678  
  CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE,  
    1678  
  CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES,  
    1678  
  CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES, 1679  
  CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIFICATIONS,  
    1679  
  CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS,  
    1679  
  CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE,  
    1680  
  CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY,  
    1680  
  CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE,  
    1680  
  CFE\_PLATFORM\_TBL\_U32FROM4CHARS, 1681  
  CFE\_PLATFORM\_TBL\_VALID\_PRID\_1, 1681  
  CFE\_PLATFORM\_TBL\_VALID\_PRID\_2, 1681  
  CFE\_PLATFORM\_TBL\_VALID\_PRID\_3, 1681  
  CFE\_PLATFORM\_TBL\_VALID\_PRID\_4, 1681  
  CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT, 1682  
  CFE\_PLATFORM\_TBL\_VALID\_SCID\_1, 1682  
  CFE\_PLATFORM\_TBL\_VALID\_SCID\_2, 1682

CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT, 1682                    cfe\_tbl\_eventids.h, 1655  
 DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYCIES\_TBL\_LOAD\_PEND\_REQ\_INF\_EID  
     1683    cfe\_tbl\_eventids.h, 1655  
 DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMPCFIE\_TBL\_LOAD\_SUCCESS\_INF\_EID  
     1683    cfe\_tbl\_eventids.h, 1656  
 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLESSTBL\_LOAD\_TBLNAME\_MISMATCH\_ERR\_EID  
     1683    cfe\_tbl\_eventids.h, 1656  
 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE\_TBL\_LOAD\_TYPE\_ERR\_EID  
     1683    cfe\_tbl\_eventids.h, 1656  
 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLESSE\_TBL\_LOAD\_VAL\_ERR\_EID  
     1683    cfe\_tbl\_eventids.h, 1656  
 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES\$CFE\_TBL\_LOADBUFFID\_BASE  
     1683    cFE Resource ID base values, 393  
 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDACIONSTBL\_LoadCmd, 745  
     1683    CommandHeader, 746  
 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_PAYLOADS, 746  
     1683    CFE\_TBL\_LoadCmd\_Payload, 746  
 DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZELOADFilename, 746  
     1683    CFE\_TBL\_LoadCmd\_Payload\_t  
 DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITYddefault\_cfe\_tbl\_msgdefs.h, 1632  
     1684    CFE\_TBL\_LoadCmd\_t  
 DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZEfault\_cfe\_tbl\_msgstruct.h, 1636  
     1684    CFE\_TBL\_LOADING\_A\_DUMP\_ONLY\_ERR\_EID  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1,                    cfe\_tbl\_eventids.h, 1657  
     1684    CFE\_TBL\_Manage  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2,                    cFE Manage Table Content APIs, 365  
     1684    CFE\_TBL\_MAX\_FULL\_NAME\_LEN  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3,                    cfe\_tbl\_api\_typedefs.h, 1393  
     1684    CFE\_TBL\_MESSAGE\_ERROR  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4,                    cFE Return Code Defines, 250  
     1684    CFE\_TBL\_MID\_ERR\_EID  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT,                cfe\_tbl\_eventids.h, 1657  
     1684    CFE\_TBL\_Modified  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1,                    cFE Manage Table Content APIs, 365  
     1684    CFE\_TBL\_NO\_INACTIVE\_BUFFER\_ERR\_EID  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_2,                    cfe\_tbl\_eventids.h, 1657  
     1684    CFE\_TBL\_NO SUCH\_TABLE\_ERR\_EID  
 DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT,                cfe\_tbl\_eventids.h, 1657  
     1684    CFE\_TBL\_NO\_WORK\_BUFFERS\_ERR\_EID  
 CFE\_TBL\_LEN\_ERR\_EID    cfe\_tbl\_eventids.h, 1658  
 CFE\_TBL\_Load    cfe\_tbl\_fcncodes.h, 1671  
 cFE Manage Table Content APIs, 363  
 CFE\_TBL\_LOAD\_ABORT\_ERR\_EID                                cfe\_tbl\_eventids.h, 1654  
 CFE\_TBL\_LOAD\_ABORT\_INF\_EID                                cfe\_tbl\_eventids.h, 1654  
 CFE\_TBL\_LOAD\_CC    cfe\_tbl\_fcncodes.h, 1670  
 CFE\_TBL\_LOAD\_EXCEEDS\_SIZE\_ERR\_EID                        cfe\_tbl\_eventids.h, 1655  
 CFE\_TBL\_LOAD\_FILENAME\_LONG\_ERR\_EID                        cfe\_tbl\_eventids.h, 1655  
 CFE\_TBL\_LOAD\_IN\_PROGRESS\_ERR\_EID                         cfe\_tbl\_eventids.h, 1658  
 CFE\_TBL\_NOOP\_CC    cfe\_tbl\_fcncodes.h, 1671  
 CFE\_TBL\_NOOP\_INF\_EID                                        cfe\_tbl\_eventids.h, 1658  
 CFE\_TBL\_NoopCmd, 746                                        CommandHeader, 747  
 CFE\_TBL\_NoopCmd\_t    default\_cfe\_tbl\_msgstruct.h, 1636  
 CFE\_TBL\_NOT\_CRITICAL\_TBL\_ERR\_EID                         cfe\_tbl\_eventids.h, 1658  
 CFE\_TBL\_NOT\_IMPLEMENTED                                    cFE Return Code Defines, 250  
 CFE\_TBL\_NOT\_IN\_CRIT\_REG\_ERR\_EID                         cfe\_tbl\_eventids.h, 1658

CFE\_TBL\_NotifyByMessage  
    cFE Get Table Information APIs, 374

CFE\_TBL\_NotifyCmd, 747  
    CommandHeader, 747  
    Payload, 747

CFE\_TBL\_NotifyCmd\_Payload, 747  
    Parameter, 748

CFE\_TBL\_NotifyCmd\_Payload\_t  
    default\_cfe\_tbl\_msgdefs.h, 1632

CFE\_TBL\_NotifyCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1636

CFE\_TBL\_OPT\_BUFFER\_MSK  
    cFE Table Type Defines, 375

CFE\_TBL\_OPT\_CRITICAL  
    cFE Table Type Defines, 375

CFE\_TBL\_OPT\_CRITICAL\_MSK  
    cFE Table Type Defines, 375

CFE\_TBL\_OPT\_DBL\_BUFFER  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_DEFAULT  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_DUMP\_ONLY  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_LD\_DMP\_MSK  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_LOAD\_DUMP  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_NOT\_CRITICAL  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_NOT\_USR\_DEF  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_SNGL\_BUFFER  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_USR\_DEF\_ADDR  
    cFE Table Type Defines, 376

CFE\_TBL\_OPT\_USR\_DEF\_MSK  
    cFE Table Type Defines, 376

CFE\_TBL\_OVERWRITE\_DUMP\_INF\_EID  
    cfe\_tbl\_eventids.h, 1659

CFE\_TBL\_OVERWRITE\_REG\_DUMP\_INF\_EID  
    cfe\_tbl\_eventids.h, 1659

CFE\_TBL\_PARTIAL\_LOAD\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1659

CFE\_TBL\_PROCESSOR\_ID\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1659

CFE\_TBL\_REG\_TLM\_MID  
    default\_cfe\_tbl\_msgids.h, 1634

CFE\_TBL\_REGID\_BASE  
    cFE Resource ID base values, 393

CFE\_TBL\_REGID\_C  
    cfe\_tbl\_api\_typedefs.h, 1393

CFE\_TBL\_RegId\_t  
    default\_cfe\_tbl\_extern\_typedefs.h, 1626

CFE\_TBL\_REGID\_UNDEFINED

    cfe\_tbl\_api\_typedefs.h, 1393

CFE\_TBL\_Register  
    cFE Registration APIs, 358

CFE\_TBL\_REGISTER\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1660

CFE\_TBL\_ReleaseAddress  
    cFE Access Table Content APIs, 370

CFE\_TBL\_ReleaseAddresses  
    cFE Access Table Content APIs, 371

CFE\_TBL\_RESET\_COUNTERS\_CC  
    cfe\_tbl\_fcncodes.h, 1672

CFE\_TBL\_RESET\_INF\_EID  
    cfe\_tbl\_eventids.h, 1660

CFE\_TBL\_ResetCountersCmd, 748  
    CommandHeader, 748

CFE\_TBL\_ResetCountersCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1636

CFE\_TBL\_SEND\_HK\_MID  
    default\_cfe\_tbl\_msgids.h, 1634

CFE\_TBL\_SEND\_REGISTRY\_CC  
    cfe\_tbl\_fcncodes.h, 1672

CFE\_TBL\_SendHkCmd, 748  
    CommandHeader, 749

CFE\_TBL\_SendHkCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1636

CFE\_TBL\_SendRegistryCmd, 749  
    CommandHeader, 749  
    Payload, 749

CFE\_TBL\_SendRegistryCmd\_Payload, 750  
    TableName, 750

CFE\_TBL\_SendRegistryCmd\_Payload\_t  
    default\_cfe\_tbl\_msgdefs.h, 1632

CFE\_TBL\_SendRegistryCmd\_t  
    default\_cfe\_tbl\_msgstruct.h, 1636

CFE\_TBL\_Share  
    cFE Registration APIs, 360

CFE\_TBL\_SHARE\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1660

CFE\_TBL\_SPACECRAFT\_ID\_ERR\_EID  
    cfe\_tbl\_eventids.h, 1660

CFE\_TBL\_SRC\_ADDRESS  
    cfe\_tbl\_api\_typedefs.h, 1394

CFE\_TBL\_SRC\_FILE  
    cfe\_tbl\_api\_typedefs.h, 1394

CFE\_TBL\_SrcEnum  
    cfe\_tbl\_api\_typedefs.h, 1394

CFE\_TBL\_SrcEnum\_t  
    cfe\_tbl\_api\_typedefs.h, 1393

CFE\_TBL\_TableRegistryTlm, 750  
    Payload, 750  
    TelemetryHeader, 750

CFE\_TBL\_TableRegistryTlm\_t  
    default\_cfe\_tbl\_msgstruct.h, 1636

CFE\_TBL\_TblRegPacket\_Payload, 751

ActiveBufferAddr, [752](#)  
 ByteAlign4, [752](#)  
 Crc, [752](#)  
 Critical, [752](#)  
 DoubleBuffered, [752](#)  
 DumpOnly, [752](#)  
 FileTime, [752](#)  
 InactiveBufferAddr, [752](#)  
 LastFileLoaded, [753](#)  
 LoadPending, [753](#)  
 Name, [753](#)  
 OwnerAppName, [753](#)  
 Size, [753](#)  
 TableLoadedOnce, [753](#)  
 TimeOfLastUpdate, [753](#)  
 ValidationFuncPtr, [754](#)  
**CFE\_TBL\_TblRegPacket\_Payload\_t**  
 default\_cfe\_tbl\_msgdefs.h, [1632](#)  
**CFE\_TBL\_TLM\_REG\_CMD\_INF\_EID**  
 cfe\_tbl\_eventids.h, [1661](#)  
**CFE\_TBL\_TOO\_MANY\_DUMPSS\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1661](#)  
**CFE\_TBL\_TOO\_MANY\_VALIDATIONS\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1661](#)  
**cfe\_tbl\_topicids.h**  
 CFE\_MISSION\_TBL\_CMD\_TOPICID, [1685](#)  
 CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID, [1685](#)  
 CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID, [1685](#)  
 CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID, [1686](#)  
 DEFAULT\_CFE\_MISSION\_TBL\_CMD\_TOPICID,  
     [1686](#)  
 DEFAULT\_CFE\_MISSION\_TBL\_HK\_TLM\_TOPICID,  
     [1686](#)  
 DEFAULT\_CFE\_MISSION\_TBL\_REG\_TLM\_TOPICID, [1686](#)  
 DEFAULT\_CFE\_MISSION\_TBL\_SEND\_HK\_TOPICID, [1686](#)  
**CFE\_TBL\_Unregister**  
 cFE Registration APIs, [361](#)  
**CFE\_TBL\_UNREGISTER\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1661](#)  
**CFE\_TBL\_UNVALIDATED\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1662](#)  
**CFE\_TBL\_Update**  
 cFE Manage Table Content APIs, [366](#)  
**CFE\_TBL\_UPDATE\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1662](#)  
**CFE\_TBL\_UPDATE\_SUCCESS\_INF\_EID**  
 cfe\_tbl\_eventids.h, [1662](#)  
**CFE\_TBL\_VAL\_REQ\_MADE\_INF\_EID**  
 cfe\_tbl\_eventids.h, [1662](#)  
**CFE\_TBL\_Validate**  
 cFE Manage Table Content APIs, [367](#)  
**CFE\_TBL\_VALIDATE\_CC**  
 cfe\_tbl\_fcncodes.h, [1673](#)  
**CFE\_TBL\_ValidateCmd**, [754](#)  
 CommandHeader, [754](#)  
 Payload, [754](#)  
**CFE\_TBL\_ValidateCmd\_Payload**, [754](#)  
 ActiveTableFlag, [755](#)  
 TableName, [755](#)  
**CFE\_TBL\_ValidateCmd\_Payload\_t**  
 default\_cfe\_tbl\_msgdefs.h, [1633](#)  
**CFE\_TBL\_ValidateCmd\_t**  
 default\_cfe\_tbl\_msgstruct.h, [1636](#)  
**CFE\_TBL\_VALIDATION\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1663](#)  
**CFE\_TBL\_VALIDATION\_INF\_EID**  
 cfe\_tbl\_eventids.h, [1663](#)  
**CFE\_TBL\_VALRESULTID\_BASE**  
 cFE Resource ID base values, [393](#)  
**CFE\_TBL\_WARN\_DUPLICATE**  
 cFE Return Code Defines, [250](#)  
**CFE\_TBL\_WARN\_NOT\_CRITICAL**  
 cFE Return Code Defines, [250](#)  
**CFE\_TBL\_WARN\_PARTIAL\_LOAD**  
 cFE Return Code Defines, [250](#)  
**CFE\_TBL\_WARN\_SHORT\_FILE**  
 cFE Return Code Defines, [251](#)  
**CFE\_TBL\_WRITE\_CFE\_HDR\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1663](#)  
**CFE\_TBL\_WRITE\_DUMP\_INF\_EID**  
 cfe\_tbl\_eventids.h, [1663](#)  
**CFE\_TBL\_WRITE\_REG\_DUMP\_INF\_EID**  
 cfe\_tbl\_eventids.h, [1664](#)  
**CFE\_TBL\_WRITE\_TBL\_HDR\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1664](#)  
**CFE\_TBL\_WRITE\_TBL\_IMG\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1664](#)  
**CFE\_TBL\_WRITE\_TBL\_REG\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1664](#)  
**CFE\_TBL\_ZERO\_LENGTH\_LOAD\_ERR\_EID**  
 cfe\_tbl\_eventids.h, [1665](#)  
**cfe\_time.h**  
 CFE\_TIME\_Copy, [1397](#)  
**CFE\_TIME\_1HZ\_CMD\_MID**  
 default\_cfe\_time\_msgids.h, [1697](#)  
**CFE\_TIME\_A\_GT\_B**  
 cfe\_time\_api\_typedefs.h, [1399](#)  
**CFE\_TIME\_A\_LT\_B**  
 cfe\_time\_api\_typedefs.h, [1399](#)  
**CFE\_TIME\_Add**  
 cFE Time Arithmetic APIs, [382](#)  
**CFE\_TIME\_ADD\_ADJUST\_CC**  
 cfe\_time\_fcncodes.h, [1713](#)  
**CFE\_TIME\_ADD\_DELAY\_CC**  
 cfe\_time\_fcncodes.h, [1714](#)  
**CFE\_TIME\_ADD\_ONE\_HZ\_ADJUSTMENT\_CC**

cfe\_time\_fcncodes.h, 1714  
    CFE\_TIME\_AddAdjustCmd, 755  
        CommandHeader, 755  
        Payload, 756  
    CFE\_TIME\_AddAdjustCmd\_t  
        default\_cfe\_time\_msgstruct.h, 1699  
    CFE\_TIME\_AddDelayCmd, 756  
        CommandHeader, 756  
        Payload, 756  
    CFE\_TIME\_AddDelayCmd\_t  
        default\_cfe\_time\_msgstruct.h, 1699  
    CFE\_TIME\_AddOneHzAdjustmentCmd, 756  
        CommandHeader, 757  
        Payload, 757  
    CFE\_TIME\_AddOneHzAdjustmentCmd\_t  
        default\_cfe\_time\_msgstruct.h, 1700  
    CFE\_TIME\_AdjustDirection  
        default\_cfe\_time\_extern\_typedefs.h, 1689  
    CFE\_TIME\_AdjustDirection\_ADD  
        default\_cfe\_time\_extern\_typedefs.h, 1689  
    CFE\_TIME\_AdjustDirection\_Enum\_t  
        default\_cfe\_time\_extern\_typedefs.h, 1687  
    CFE\_TIME\_AdjustDirection\_SUBTRACT  
        default\_cfe\_time\_extern\_typedefs.h, 1689  
    cfe\_time\_api\_typedefs.h  
        CFE\_TIME\_A\_GT\_B, 1399  
        CFE\_TIME\_A\_LT\_B, 1399  
        CFE\_TIME\_Compare, 1398  
        CFE\_TIME\_Compare\_t, 1398  
        CFE\_TIME\_EQUAL, 1399  
        CFE\_TIME\_PRINTED\_STRING\_SIZE, 1398  
        CFE\_TIME\_SynchCallbackPtr\_t, 1398  
        CFE\_TIME\_ZERO\_VALUE, 1398  
    CFE\_TIME\_BAD\_ARGUMENT  
        cFE Return Code Defines, 251  
    CFE\_TIME\_CALLBACK\_NOT\_REGISTERED  
        cFE Return Code Defines, 251  
    CFE\_TIME\_CC\_ERR\_EID  
        cfe\_time\_eventids.h, 1703  
    CFE\_TIME\_CCVAL  
        default\_cfe\_time\_fcncode\_values.h, 1691  
    CFE\_TIME\_ClockState  
        default\_cfe\_time\_extern\_typedefs.h, 1689  
    CFE\_TIME\_ClockState\_Enum\_t  
        default\_cfe\_time\_extern\_typedefs.h, 1687  
    CFE\_TIME\_ClockState\_FLYWHEEL  
        default\_cfe\_time\_extern\_typedefs.h, 1689  
    CFE\_TIME\_ClockState\_INVALID  
        default\_cfe\_time\_extern\_typedefs.h, 1689  
    CFE\_TIME\_ClockState\_VALID  
        default\_cfe\_time\_extern\_typedefs.h, 1689  
    CFE\_TIME\_CMD\_MID  
        default\_cfe\_time\_msgids.h, 1697  
    CFE\_TIME\_Compare  
        cFE Time Arithmetic APIs, 383  
        cfe\_time\_api\_typedefs.h, 1398  
    CFE\_TIME\_Compare\_t  
        cfe\_time\_api\_typedefs.h, 1398  
    CFE\_TIME\_Copy  
        cfe\_time.h, 1397  
    CFE\_TIME\_DATA\_CMD\_MID  
        default\_cfe\_time\_msgids.h, 1697  
    CFE\_TIME\_DELAY\_CFG\_EID  
        cfe\_time\_eventids.h, 1703  
    CFE\_TIME\_DELAY\_EID  
        cfe\_time\_eventids.h, 1704  
    CFE\_TIME\_DELAY\_ERR\_EID  
        cfe\_time\_eventids.h, 1704  
    CFE\_TIME\_DELTA\_CFG\_EID  
        cfe\_time\_eventids.h, 1704  
    CFE\_TIME\_DELTA\_EID  
        cfe\_time\_eventids.h, 1704  
    CFE\_TIME\_DELTA\_ERR\_EID  
        cfe\_time\_eventids.h, 1705  
    CFE\_TIME\_DIAG\_EID  
        cfe\_time\_eventids.h, 1705  
    CFE\_TIME\_DIAG\_TLM\_MID  
        default\_cfe\_time\_msgids.h, 1697  
    CFE\_TIME\_DiagnosticTlm, 757  
        Payload, 757  
        TelemetryHeader, 757  
    CFE\_TIME\_DiagnosticTlm\_Payload, 757  
        AtToneDelay, 759  
        AtToneLatch, 760  
        AtToneLeapSeconds, 760  
        AtToneMET, 760  
        AtToneSTCF, 760  
        ClockFlyState, 760  
        ClockSetState, 760  
        ClockSignal, 760  
        ClockSource, 761  
        ClockStateAPI, 761  
        ClockStateFlags, 761  
        CurrentLatch, 761  
        CurrentMET, 761  
        CurrentTAI, 761  
        CurrentUTC, 761  
        DataStoreStatus, 762  
        DelayDirection, 762  
        Forced2Fly, 762  
        LocalIntCounter, 762  
        LocalTaskCounter, 762  
        MaxElapsed, 762  
        MaxLocalClock, 762  
        MinElapsed, 763  
        OneHzAdjust, 763  
        OneHzDirection, 763  
        OneTimeAdjust, 763

OneTimeDirection, 763  
 ServerFlyState, 763  
 TimeSinceTone, 763  
 ToneDataCounter, 764  
 ToneDataLatch, 764  
 ToneIntCounter, 764  
 ToneIntErrorCounter, 764  
 ToneMatchCounter, 764  
 ToneMatchErrorCounter, 764  
 ToneOverLimit, 764  
 ToneSignalCounter, 765  
 ToneSignalLatch, 765  
 ToneTaskCounter, 765  
 ToneUnderLimit, 765  
 VersionCounter, 765  
 VirtualMET, 765  
**CFE\_TIME\_DiagnosticTim\_Payload\_t**  
 default\_cfe\_time\_msgdefs.h, 1695  
**CFE\_TIME\_DiagnosticTim\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_EQUAL**  
 cfe\_time\_api\_typedefs.h, 1399  
**cfe\_time\_eventids.h**  
 CFE\_TIME\_CC\_ERR\_EID, 1703  
 CFE\_TIME\_DELAY\_CFG\_EID, 1703  
 CFE\_TIME\_DELAY\_EID, 1704  
 CFE\_TIME\_DELAY\_ERR\_EID, 1704  
 CFE\_TIME\_DELTA\_CFG\_EID, 1704  
 CFE\_TIME\_DELTA\_EID, 1704  
 CFE\_TIME\_DELTA\_ERR\_EID, 1705  
 CFE\_TIME\_DIAG\_EID, 1705  
 CFE\_TIME\_FLY\_OFF\_EID, 1705  
 CFE\_TIME\_FLY\_ON\_EID, 1705  
 CFE\_TIME\_ID\_ERR\_EID, 1706  
 CFE\_TIME\_INIT\_EID, 1706  
 CFE\_TIME\_LEAPS\_CFG\_EID, 1706  
 CFE\_TIME\_LEAPS\_EID, 1706  
 CFE\_TIME\_LEN\_ERR\_EID, 1707  
 CFE\_TIME\_MET\_CFG\_EID, 1707  
 CFE\_TIME\_MET\_EID, 1707  
 CFE\_TIME\_MET\_ERR\_EID, 1707  
 CFE\_TIME\_NOOP\_EID, 1708  
 CFE\_TIME\_ONEHZ\_CFG\_EID, 1708  
 CFE\_TIME\_ONEHZ\_EID, 1708  
 CFE\_TIME\_RESET\_EID, 1708  
 CFE\_TIME\_SIGNAL\_CFG\_EID, 1709  
 CFE\_TIME\_SIGNAL\_EID, 1709  
 CFE\_TIME\_SIGNAL\_ERR\_EID, 1709  
 CFE\_TIME\_SOURCE\_CFG\_EID, 1709  
 CFE\_TIME\_SOURCE\_EID, 1710  
 CFE\_TIME\_SOURCE\_ERR\_EID, 1710  
 CFE\_TIME\_STATE\_EID, 1710  
 CFE\_TIME\_STATE\_ERR\_EID, 1710  
 CFE\_TIME\_STCF\_CFG\_EID, 1711  
 CFE\_TIME\_STCF\_EID, 1711  
 CFE\_TIME\_STCF\_ERR\_EID, 1711  
 CFE\_TIME\_TIME\_CFG\_EID, 1711  
 CFE\_TIME\_TIME\_EID, 1712  
 CFE\_TIME\_TIME\_ERR\_EID, 1712  
**CFE\_TIME\_ExternalGPS**  
 cFE External Time Source APIs, 386  
**CFE\_TIME\_ExternalMET**  
 cFE External Time Source APIs, 387  
**CFE\_TIME\_ExternalTime**  
 cFE External Time Source APIs, 388  
**CFE\_TIME\_ExternalTone**  
 cFE External Time Source APIs, 388  
**CFE\_TIME\_FakeToneCmd**, 766  
 CommandHeader, 766  
**CFE\_TIME\_FakeToneCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**cfe\_time\_fnccodes.h**  
 CFE\_TIME\_ADD\_ADJUST\_CC, 1713  
 CFE\_TIME\_ADD\_DELAY\_CC, 1714  
 CFE\_TIME\_ADD\_ONE\_HZ\_ADJUSTMENT\_CC,  
 1714  
 CFE\_TIME\_NOOP\_CC, 1715  
 CFE\_TIME\_RESET\_COUNTERS\_CC, 1716  
 CFE\_TIME\_SEND\_DIAGNOSTIC\_CC, 1717  
 CFE\_TIME\_SET\_LEAP\_SECONDS\_CC, 1718  
 CFE\_TIME\_SET\_MET\_CC, 1719  
 CFE\_TIME\_SET\_SIGNAL\_CC, 1720  
 CFE\_TIME\_SET\_SOURCE\_CC, 1720  
 CFE\_TIME\_SET\_STATE\_CC, 1721  
 CFE\_TIME\_SET\_STCF\_CC, 1723  
 CFE\_TIME\_SET\_TIME\_CC, 1723  
 CFE\_TIME\_SUB\_ADJUST\_CC, 1724  
 CFE\_TIME\_SUB\_DELAY\_CC, 1725  
 CFE\_TIME\_SUB\_ONE\_HZ\_ADJUSTMENT\_CC,  
 1726  
**CFE\_TIME\_FLAG\_ADD1HZ**  
 cFE Clock State Flag Defines, 394  
**CFE\_TIME\_FLAG\_ADDADJ**  
 cFE Clock State Flag Defines, 394  
**CFE\_TIME\_FLAG\_ADDTCL**  
 cFE Clock State Flag Defines, 394  
**CFE\_TIME\_FLAG\_CLKSET**  
 cFE Clock State Flag Defines, 394  
**CFE\_TIME\_FLAG\_CMDFLY**  
 cFE Clock State Flag Defines, 394  
**CFE\_TIME\_FLAG\_FLYING**  
 cFE Clock State Flag Defines, 394  
**CFE\_TIME\_FLAG\_GDTONE**  
 cFE Clock State Flag Defines, 395  
**CFE\_TIME\_FLAG\_REFERR**  
 cFE Clock State Flag Defines, 395  
**CFE\_TIME\_FLAG\_SERVER**  
 cFE Clock State Flag Defines, 395

CFE\_TIME\_FLAG\_SIGPRI  
    cFE Clock State Flag Defines, 395

CFE\_TIME\_FLAG\_SRCINT  
    cFE Clock State Flag Defines, 395

CFE\_TIME\_FLAG\_SRVFLY  
    cFE Clock State Flag Defines, 395

CFE\_TIME\_FLAG\_UNUSED  
    cFE Clock State Flag Defines, 395

CFE\_TIME\_FlagBit  
    default\_cfe\_time\_extern\_typedefs.h, 1689

CFE\_TIME\_FlagBit\_ADD1HZ  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_ADDADJ  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_ADDTCL  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_CLKSET  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_CMDFLY  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_Enum\_t  
    default\_cfe\_time\_extern\_typedefs.h, 1688

CFE\_TIME\_FlagBit\_FLYING  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_GDTONE  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_SERVER  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_SIGPRI  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_SRCINT  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlagBit\_SRVFLY  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FLY\_OFF\_EID  
    cfe\_time\_eventids.h, 1705

CFE\_TIME\_FLY\_ON\_EID  
    cfe\_time\_eventids.h, 1705

CFE\_TIME\_FlywheelState  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlywheelState\_Enum\_t  
    default\_cfe\_time\_extern\_typedefs.h, 1688

CFE\_TIME\_FlywheelState\_IS\_FLY  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FlywheelState\_NO\_FLY  
    default\_cfe\_time\_extern\_typedefs.h, 1690

CFE\_TIME\_FunctionCode\_  
    default\_cfe\_time\_fncode\_values.h, 1691

CFE\_TIME\_FunctionCode\_ADD\_ADJUST  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_ADD\_DELAY  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_ADD\_ONE\_HZ\_ADJUSTMENT  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_NOOP  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_RESET\_COUNTERS  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SEND\_DIAGNOSTIC  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SET\_LEAP\_SECONDS  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SET\_MET  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SET\_SIGNAL  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SET\_SOURCE  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SET\_STATE  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SET\_STCF  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SET\_TIME  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SUB\_ADJUST  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SUB\_DELAY  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_FunctionCode\_SUB\_ONE\_HZ\_ADJUSTMENT  
    default\_cfe\_time\_fncode\_values.h, 1692

CFE\_TIME\_GetClockInfo  
    cFE Get Time Information APIs, 380

CFE\_TIME\_GetClockState  
    cFE Get Time Information APIs, 380

CFE\_TIME\_GetLeapSeconds  
    cFE Get Time Information APIs, 381

CFE\_TIME\_GetMET  
    cFE Get Current Time APIs, 377

CFE\_TIME\_GetMETseconds  
    cFE Get Current Time APIs, 377

CFE\_TIME\_GetMETsubsecs  
    cFE Get Current Time APIs, 378

CFE\_TIME\_GetSTCF  
    cFE Get Time Information APIs, 381

CFE\_TIME\_GetTAI  
    cFE Get Current Time APIs, 378

CFE\_TIME\_GetTime  
    cFE Get Current Time APIs, 379

CFE\_TIME\_GetUTC  
    cFE Get Current Time APIs, 379

CFE\_TIME\_HK\_TLM\_MID  
    default\_cfe\_time\_msgids.h, 1697

CFE\_TIME\_HousekeepingTlm, 766  
    Payload, 766  
    TelemetryHeader, 766

CFE\_TIME\_HousekeepingTlm\_Payload, 767  
    AdjustmentFactor, 767  
    ClockStateAPI, 767

ClockStateFlags, [767](#)  
 CommandCounter, [768](#)  
 CommandErrorCounter, [768](#)  
 LeapSeconds, [768](#)  
 MET, [768](#)  
 STCF, [768](#)

CFE\_TIME\_HousekeepingTlm\_Payload\_t  
 default\_cfe\_time\_msgdefs.h, [1695](#)

CFE\_TIME\_HousekeepingTlm\_t  
 default\_cfe\_time\_msgstruct.h, [1700](#)

CFE\_TIME\_ID\_ERR\_EID  
 cfe\_time\_eventids.h, [1706](#)

CFE\_TIME\_INIT\_EID  
 cfe\_time\_eventids.h, [1706](#)

cfe\_time\_interface\_cfg.h

- CFE\_MISSION\_TIME\_AT\_TONE\_WAS, [1729](#)
- CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE, [1729](#)
- CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI, [1729](#)
- CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC, [1730](#)
- CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE, [1730](#)
- CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS, [1730](#)
- CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS, [1730](#)
- CFE\_MISSION\_TIME\_DEF\_LEAPS, [1730](#)
- CFE\_MISSION\_TIME\_DEF\_MET\_SECS, [1730](#)
- CFE\_MISSION\_TIME\_DEF\_MET\_SUBS, [1731](#)
- CFE\_MISSION\_TIME\_DEF\_STCF\_SECS, [1731](#)
- CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS, [1731](#)
- CFE\_MISSION\_TIME\_EPOCH\_DAY, [1731](#)
- CFE\_MISSION\_TIME\_EPOCH\_HOUR, [1731](#)
- CFE\_MISSION\_TIME\_EPOCH\_MICROS, [1731](#)
- CFE\_MISSION\_TIME\_EPOCH\_MINUTE, [1731](#)
- CFE\_MISSION\_TIME\_EPOCH\_SECOND, [1731](#)
- CFE\_MISSION\_TIME\_EPOCH\_SECONDS, [1732](#)
- CFE\_MISSION\_TIME\_EPOCH\_YEAR, [1732](#)
- CFE\_MISSION\_TIME\_FS\_FACTOR, [1732](#)
- CFE\_MISSION\_TIME\_MAX\_ELAPSED, [1732](#)
- CFE\_MISSION\_TIME\_MIN\_ELAPSED, [1732](#)
- DEFAULT\_CFE\_MISSION\_TIME\_AT\_TONE\_WAS, [1733](#)
- DEFAULT\_CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE, [1733](#)
- DEFAULT\_CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI, [1733](#)
- DEFAULT\_CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC, [1733](#)
- DEFAULT\_CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE, [1733](#)
- DEFAULT\_CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS, [1733](#)
- DEFAULT\_CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS, [1733](#)
- DEFAULT\_CFE\_MISSION\_TIME\_DEF\_LEAPS, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_DEF\_MET\_SECS, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_DEF\_MET\_SUBS, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_DEF\_STCF\_SECS, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_DAY, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_HOUR, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_MICROS, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_MINUTE, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_SECOND, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_SECONDS, [1734](#)
- DEFAULT\_CFE\_MISSION\_TIME\_EPOCH\_YEAR, [1735](#)
- DEFAULT\_CFE\_MISSION\_TIME\_FS\_FACTOR, [1735](#)
- DEFAULT\_CFE\_MISSION\_TIME\_MAX\_ELAPSED, [1735](#)
- DEFAULT\_CFE\_MISSION\_TIME\_MIN\_ELAPSED, [1735](#)

cfe\_time\_internal\_cfg.h

  - CFE\_PLATFORM\_TIME\_CFG\_CLIENT, [1736](#)
  - CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY, [1736](#)
  - CFE\_PLATFORM\_TIME\_CFG\_SERVER, [1737](#)
  - CFE\_PLATFORM\_TIME\_CFG\_SIGNAL, [1737](#)
  - CFE\_PLATFORM\_TIME\_CFG\_SOURCE, [1737](#)
  - CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS, [1738](#)
  - CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET, [1738](#)
  - CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME, [1738](#)
  - CFE\_PLATFORM\_TIME\_CFG\_START\_FLY, [1738](#)
  - CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT, [1739](#)
  - CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL, [1739](#)
  - CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS, [1739](#)
  - CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS, [1740](#)
  - CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS, [1740](#)
  - CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS, [1740](#)
  - CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY, [1740](#)
  - CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE, [1740](#)
  - CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY, [1740](#)
  - CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE, [1741](#)
  - CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY, [1741](#)

CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE, CFE\_TIME\_LEN\_ERR\_EID  
1741 cfe\_time\_eventids.h, 1707

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_CLIENT, CFE\_TIME\_Local1HzISR  
1741 cFE Miscellaneous Time APIs, 390

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY, CFE\_TIME\_MET2SCTime  
1741 cFE Time Conversion APIs, 384

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SERVER, CFE\_TIME\_MET\_CFG\_EID  
1742 cfe\_time\_eventids.h, 1707

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SIGNAL, CFE\_TIME\_MET\_EID  
1742 cfe\_time\_eventids.h, 1707

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SOURCE, CFE\_TIME\_MET\_ERR\_EID  
1742 cfe\_time\_eventids.h, 1707

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS, CFE\_TIME\_Micro2SubSecs  
1742 cFE Time Conversion APIs, 385

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET, CFE\_TIME\_NOOP\_CC  
1742 cfe\_time\_fncodes.h, 1715

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME, CFE\_TIME\_NOOP\_EID  
1742 cfe\_time\_eventids.h, 1708

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_START\_FLY, CFE\_TIME\_NoopCmd, 769  
1742 CommandHeader, 769

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT, CFE\_TIME\_NoopCmd\_t  
1742 default\_cfe\_time\_msgstruct.h, 1700

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL, CFE\_TIME\_NOT\_IMPLEMENTED  
1742 cFE Return Code Defines, 251

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS, CFE\_TIME\_ONEHZ\_CFG\_EID  
1742 cfe\_time\_eventids.h, 1708

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS, CFE\_TIME\_ONEHZ\_CMD\_MID  
1743 default\_cfe\_time\_msgids.h, 1698

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS, CFE\_TIME\_ONEHZ\_EID  
1743 cfe\_time\_eventids.h, 1708

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS, CFE\_TIME\_OneHzAdjustmentCmd\_Payload, 769  
1743 Seconds, 770

DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY, CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t  
1743 CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t

DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE, CFE\_TIME\_MsgDef.h, 1695  
1743 CFE\_TIME\_OneHzCmd, 770

DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY, CFE\_TIME\_CommandHeader, 770  
1743 CFE\_TIME\_OneHzCmd\_t

DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE, CFE\_TIME\_MsgDef.h, 1700  
1743 CFE\_TIME\_OUT\_OF\_RANGE

DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY, CFE Return Code Defines, 251  
1743 CFE\_TIME\_Print

DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE, CFE Miscellaneous Time APIs, 391  
1743 CFE\_TIME\_PRINTED\_STRING\_SIZE

CFE\_TIME\_INTERNAL\_ONLY  
cFE Return Code Defines, 251

CFE\_TIME\_LEAPS\_CFG\_EID  
cfe\_time\_eventids.h, 1706

CFE\_TIME\_LEAPS\_EID  
cfe\_time\_eventids.h, 1706

CFE\_TIME\_LeapsCmd\_Payload, 768  
LeapSeconds, 769

CFE\_TIME\_LeapsCmd\_Payload\_t  
default\_cfe\_time\_msgdefs.h, 1695

CFE\_TIME\_RegisterSyncCallback  
cFE External Time Source APIs, 389

CFE\_TIME\_RESET\_COUNTERS\_CC  
cfe\_time\_fncodes.h, 1716

CFE\_TIME\_RESET\_EID  
cfe\_time\_eventids.h, 1708

CFE\_TIME\_ResetCountersCmd, 770  
CommandHeader, 771

CFE\_TIME\_ResetCountersCmd\_t

default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SEND\_CMD\_MID**  
 default\_cfe\_time\_msgids.h, 1698  
**CFE\_TIME\_SEND\_DIAGNOSTIC\_CC**  
 cfe\_time\_fcncodes.h, 1717  
**CFE\_TIME\_SEND\_HK\_MID**  
 default\_cfe\_time\_msgids.h, 1698  
**CFE\_TIME\_SendDiagnosticCmd**, 771  
 CommandHeader, 771  
**CFE\_TIME\_SendDiagnosticCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SendHkCmd**, 771  
 CommandHeader, 771  
**CFE\_TIME\_SendHkCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SERVICE**  
 cfe\_error.h, 1353  
**CFE\_TIME\_SET\_LEAP\_SECONDS\_CC**  
 cfe\_time\_fcncodes.h, 1718  
**CFE\_TIME\_SET\_MET\_CC**  
 cfe\_time\_fcncodes.h, 1719  
**CFE\_TIME\_SET\_SIGNAL\_CC**  
 cfe\_time\_fcncodes.h, 1720  
**CFE\_TIME\_SET\_SOURCE\_CC**  
 cfe\_time\_fcncodes.h, 1720  
**CFE\_TIME\_SET\_STATE\_CC**  
 cfe\_time\_fcncodes.h, 1721  
**CFE\_TIME\_SET\_STCF\_CC**  
 cfe\_time\_fcncodes.h, 1723  
**CFE\_TIME\_SET\_TIME\_CC**  
 cfe\_time\_fcncodes.h, 1723  
**CFE\_TIME\_SetLeapSecondsCmd**, 772  
 CommandHeader, 772  
 Payload, 772  
**CFE\_TIME\_SetLeapSecondsCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SetMETCmd**, 772  
 CommandHeader, 773  
 Payload, 773  
**CFE\_TIME\_SetMETCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SetSignalCmd**, 773  
 CommandHeader, 773  
 Payload, 773  
**CFE\_TIME\_SetSignalCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SetSourceCmd**, 773  
 CommandHeader, 774  
 Payload, 774  
**CFE\_TIME\_SetSourceCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SetState**  
 default\_cfe\_time\_extern\_typedefs.h, 1690  
**CFE\_TIME\_SetState\_Enum\_t**  
 default\_cfe\_time\_extern\_typedefs.h, 1688  
**CFE\_TIME\_SetState\_NOT\_SET**  
 default\_cfe\_time\_extern\_typedefs.h, 1690  
**CFE\_TIME\_SetState\_WAS\_SET**  
 default\_cfe\_time\_extern\_typedefs.h, 1690  
**CFE\_TIME\_SetStateCmd**, 774  
 CommandHeader, 774  
 Payload, 774  
**CFE\_TIME\_SetStateCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SetSTCFCmd**, 775  
 CommandHeader, 775  
 Payload, 775  
**CFE\_TIME\_SetSTCFCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1700  
**CFE\_TIME\_SetTimeCmd**, 775  
 CommandHeader, 776  
 Payload, 776  
**CFE\_TIME\_SetTimeCmd\_t**  
 default\_cfe\_time\_msgstruct.h, 1701  
**CFE\_TIME\_SIGNAL\_CFG\_EID**  
 cfe\_time\_eventids.h, 1709  
**CFE\_TIME\_SIGNAL\_EID**  
 cfe\_time\_eventids.h, 1709  
**CFE\_TIME\_SIGNAL\_ERR\_EID**  
 cfe\_time\_eventids.h, 1709  
**CFE\_TIME\_SignalCmd\_Payload**, 776  
 ToneSource, 776  
**CFE\_TIME\_SignalCmd\_Payload\_t**  
 default\_cfe\_time\_msgdefs.h, 1696  
**CFE\_TIME\_SOURCE\_CFG\_EID**  
 cfe\_time\_eventids.h, 1709  
**CFE\_TIME\_SOURCE\_EID**  
 cfe\_time\_eventids.h, 1710  
**CFE\_TIME\_SOURCE\_ERR\_EID**  
 cfe\_time\_eventids.h, 1710  
**CFE\_TIME\_SourceCmd\_Payload**, 776  
 TimeSource, 777  
**CFE\_TIME\_SourceCmd\_Payload\_t**  
 default\_cfe\_time\_msgdefs.h, 1696  
**CFE\_TIME\_SourceSelect**  
 default\_cfe\_time\_extern\_typedefs.h, 1690  
**CFE\_TIME\_SourceSelect\_Enum\_t**  
 default\_cfe\_time\_extern\_typedefs.h, 1688  
**CFE\_TIME\_SourceSelect\_EXTERNAL**  
 default\_cfe\_time\_extern\_typedefs.h, 1690  
**CFE\_TIME\_SourceSelect\_INTERNAL**  
 default\_cfe\_time\_extern\_typedefs.h, 1690  
**CFE\_TIME\_STATE\_EID**  
 cfe\_time\_eventids.h, 1710  
**CFE\_TIME\_STATE\_ERR\_EID**  
 cfe\_time\_eventids.h, 1710  
**CFE\_TIME\_StateCmd\_Payload**, 777  
 ClockState, 777

CFE\_TIME\_StateCmd\_Payload\_t  
  default\_cfe\_time\_msgdefs.h, 1696

CFE\_TIME\_STCF\_CFG\_EID  
  cfe\_time\_eventids.h, 1711

CFE\_TIME\_STCF\_EID  
  cfe\_time\_eventids.h, 1711

CFE\_TIME\_STCF\_ERR\_EID  
  cfe\_time\_eventids.h, 1711

CFE\_TIME\_Sub2MicroSecs  
  cFE Time Conversion APIs, 385

CFE\_TIME\_SUB\_ADJUST\_CC  
  cfe\_time\_fcncodes.h, 1724

CFE\_TIME\_SUB\_DELAY\_CC  
  cfe\_time\_fcncodes.h, 1725

CFE\_TIME\_SUB\_ONE\_HZ\_ADJUSTMENT\_CC  
  cfe\_time\_fcncodes.h, 1726

CFE\_TIME\_SubAdjustCmd, 778  
  CommandHeader, 778  
  Payload, 778

CFE\_TIME\_SubAdjustCmd\_t  
  default\_cfe\_time\_msgstruct.h, 1701

CFE\_TIME\_SubDelayCmd, 778  
  CommandHeader, 778  
  Payload, 778

CFE\_TIME\_SubDelayCmd\_t  
  default\_cfe\_time\_msgstruct.h, 1701

CFE\_TIME\_SubOneHzAdjustmentCmd, 779  
  CommandHeader, 779  
  Payload, 779

CFE\_TIME\_SubOneHzAdjustmentCmd\_t  
  default\_cfe\_time\_msgstruct.h, 1701

CFE\_TIME\_Subtract  
  cFE Time Arithmetic APIs, 383

CFE\_TIME\_SynchCallbackPtr\_t  
  cfe\_time\_api\_typedefs.h, 1398

CFE\_TIME\_SysTime, 779  
  Seconds, 780  
  Subseconds, 780

CFE\_TIME\_SysTime\_t  
  default\_cfe\_time\_extern\_typedefs.h, 1688

CFE\_TIME\_TIME\_CFG\_EID  
  cfe\_time\_eventids.h, 1711

CFE\_TIME\_TIME\_EID  
  cfe\_time\_eventids.h, 1712

CFE\_TIME\_TIME\_ERR\_EID  
  cfe\_time\_eventids.h, 1712

CFE\_TIME\_TimeCmd\_Payload, 780  
  MicroSeconds, 780  
  Seconds, 780

CFE\_TIME\_TimeCmd\_Payload\_t  
  default\_cfe\_time\_msgdefs.h, 1696

CFE\_TIME\_TONE\_CMD\_MID  
  default\_cfe\_time\_msgids.h, 1698

CFE\_TIME\_ToneDataCmd, 781  
  CommandHeader, 781  
  Payload, 781

CFE\_TIME\_ToneDataCmd\_Payload, 781  
  AtToneLeapSeconds, 782  
  AtToneMET, 782  
  AtToneState, 782  
  AtToneSTCF, 782

CFE\_TIME\_ToneDataCmd\_Payload\_t  
  default\_cfe\_time\_msgdefs.h, 1696

CFE\_TIME\_ToneDataCmd\_t  
  default\_cfe\_time\_msgstruct.h, 1701

CFE\_TIME\_ToneSignalCmd, 782  
  CommandHeader, 782

CFE\_TIME\_ToneSignalCmd\_t  
  default\_cfe\_time\_msgstruct.h, 1701

CFE\_TIME\_ToneSignalSelect  
  default\_cfe\_time\_extern\_typedefs.h, 1691

CFE\_TIME\_ToneSignalSelect\_Enum\_t  
  default\_cfe\_time\_extern\_typedefs.h, 1689

CFE\_TIME\_ToneSignalSelect\_PRIMARY  
  default\_cfe\_time\_extern\_typedefs.h, 1691

CFE\_TIME\_ToneSignalSelect\_REDUNDANT  
  default\_cfe\_time\_extern\_typedefs.h, 1691

CFE\_TIME\_TOO\_MANY\_SYNCH\_CALLBACKS  
  cFE Return Code Defines, 251

cfe\_time\_topicids.h  
  CFE\_MISSION\_TIME\_CMD\_TOPICID, 1744  
  CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID, 1744  
  CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID, 1745  
  CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID, 1745  
  CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID,  
    1745  
  CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID, 1745  
  CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID, 1745  
  CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID, 1745  
  DEFAULT\_CFE\_MISSION\_TIME\_CMD\_TOPICID,  
    1745  
  DEFAULT\_CFE\_MISSION\_TIME\_DATA\_CMD\_TOPICID,  
    1745  
  DEFAULT\_CFE\_MISSION\_TIME\_DIAG\_TLM\_TOPICID,  
    1746  
  DEFAULT\_CFE\_MISSION\_TIME\_HK\_TLM\_TOPICID,  
    1746  
  DEFAULT\_CFE\_MISSION\_TIME\_ONEHZ\_CMD\_TOPICID,  
    1746  
  DEFAULT\_CFE\_MISSION\_TIME\_SEND\_CMD\_TOPICID,  
    1746  
  DEFAULT\_CFE\_MISSION\_TIME\_SEND\_HK\_TOPICID,  
    1746  
  DEFAULT\_CFE\_MISSION\_TIME\_TONE\_CMD\_TOPICID,  
    1746  
CFE\_TIME\_UnregisterSynchCallback  
  cFE External Time Source APIs, 390

CFE\_TIME\_ZERO\_VALUE

cfe\_time\_api\_typedefs.h, 1398  
**CFE\_TST**  
 cfe\_sb.h, 1385  
**cfe\_version.h**  
 CFE\_BUILD\_BASELINE, 1400  
 CFE\_BUILD\_CODENAME, 1400  
 CFE\_BUILD\_DEV\_CYCLE, 1400  
 CFE\_BUILD\_NUMBER, 1400  
 CFE\_CFG\_MAX\_VERSION\_STR\_LEN, 1400  
 CFE\_LAST\_OFFICIAL, 1400  
 CFE\_MAJOR\_VERSION, 1400  
 CFE\_MINOR\_VERSION, 1400  
 CFE\_MISSION\_REV, 1400  
 CFE\_REVISION, 1400  
 CFE\_SRC\_VERSION, 1401  
 CFE\_STR, 1401  
 CFE\_STR\_HELPER, 1401  
**CFECoreChecksum**  
 CFE\_ES\_HousekeepingTlm\_Payload, 644  
**CFEMajorVersion**  
 CFE\_ES\_HousekeepingTlm\_Payload, 644  
**CFEMinorVersion**  
 CFE\_ES\_HousekeepingTlm\_Payload, 644  
**CFEMissionRevision**  
 CFE\_ES\_HousekeepingTlm\_Payload, 644  
**CFERevision**  
 CFE\_ES\_HousekeepingTlm\_Payload, 644  
**CFS CFDP Command Codes**, 193  
 CF\_ABANDON\_CC, 194  
 CF\_CANCEL\_CC, 194  
 CF\_DISABLE\_DEQUEUE\_CC, 195  
 CF\_DISABLE\_DIR\_POLLING\_CC, 196  
 CF\_DISABLE\_ENGINE\_CC, 197  
 CF\_ENABLE\_DEQUEUE\_CC, 197  
 CF\_ENABLE\_DIR\_POLLING\_CC, 198  
 CF\_ENABLE\_ENGINE\_CC, 199  
 CF\_FREEZE\_CC, 200  
 CF\_GET\_PARAM\_CC, 201  
 CF\_NOOP\_CC, 201  
 CF\_NUM\_COMMANDS, 202  
 CF\_PLAYBACK\_DIR\_CC, 202  
 CF\_PURGE\_QUEUE\_CC, 203  
 CF\_RESET\_CC, 204  
 CF\_RESUME\_CC, 205  
 CF\_SET\_PARAM\_CC, 205  
 CF\_SUSPEND\_CC, 206  
 CF\_THAW\_CC, 207  
 CF\_TX\_FILE\_CC, 208  
 CF\_WRITE\_QUEUE\_CC, 208  
**CFS CFDP Command Message IDs**, 228  
 CF\_CMD\_MID, 228  
 CF\_SEND\_HK\_MID, 228  
 CF\_WAKE\_UP\_MID, 228  
**CFS CFDP Command Structures**, 221

CF\_AbandonCmd\_t, 224  
 CF\_CancelCmd\_t, 224  
 CF\_DisableDequeueCmd\_t, 224  
 CF\_DisableDirPollingCmd\_t, 225  
 CF\_DisableEngineCmd\_t, 225  
 CF\_EnableDequeueCmd\_t, 225  
 CF\_EnableDirPollingCmd\_t, 225  
 CF\_EnableEngineCmd\_t, 225  
 CF\_FreezeCmd\_t, 225  
 CF\_GetParam\_Payload\_t, 225  
 CF\_GetParamCmd\_t, 225  
 CF\_Set\_ValueID\_ack\_limit, 227  
 CF\_Set\_ValueID\_ack\_timer\_s, 227  
 CF\_Set\_ValueID\_chan\_max\_outgoing\_messages\_per\_wakeup, 227  
 CF\_Set\_ValueID\_inactivity\_timer\_s, 227  
 CF\_Set\_ValueID\_local\_eid, 227  
 CF\_Set\_ValueID\_MAX, 227  
 CF\_Set\_ValueID\_nak\_limit, 227  
 CF\_Set\_ValueID\_nak\_timer\_s, 227  
 CF\_Set\_ValueID\_outgoing\_file\_chunk\_size, 227  
 CF\_Set\_ValueID\_rx\_crc\_calc\_bytes\_per\_wakeup, 227  
 CF\_Set\_ValueID\_t, 227  
 CF\_Set\_ValueID\_ticks\_per\_second, 227  
 CF\_NoopCmd\_t, 225  
 CF\_PlaybackDirCmd\_t, 225  
 CF\_PurgeQueueCmd\_t, 225  
 CF\_Queue\_active, 227  
 CF\_Queue\_all, 227  
 CF\_Queue\_history, 227  
 CF\_Queue\_pend, 227  
 CF\_Queue\_t, 227  
 CF\_Reset\_all, 227  
 CF\_Reset\_command, 227  
 CF\_Reset\_down, 227  
 CF\_Reset\_fault, 227  
 CF\_Reset\_t, 227  
 CF\_Reset\_up, 227  
 CF\_ResetCountersCmd\_t, 225  
 CF\_ResumeCmd\_t, 225  
 CF\_SendHkCmd\_t, 226  
 CF\_SetParam\_Payload\_t, 226  
 CF\_SetParamCmd\_t, 226  
 CF\_SuspendCmd\_t, 226  
 CF\_ThawCmd\_t, 226  
 CF\_Transaction\_Payload\_t, 226  
 CF\_TxFile\_Payload\_t, 226  
 CF\_TxFileCmd\_t, 226  
 CF\_Type\_all, 228  
 CF\_Type\_down, 228  
 CF\_Type\_t, 228  
 CF\_Type\_up, 228  
 CF\_UnionArgs\_Payload\_t, 226

CF\_WakeupCmd\_t, 226  
CF\_WriteQueue\_Payload\_t, 226  
CF\_WriteQueueCmd\_t, 226  
CFS CFDP Data Interface Message IDs, 229  
CF\_CH0\_RX\_MID, 229  
CF\_CH0\_TX\_MID, 229  
CF\_CH1\_RX\_MID, 229  
CF\_CH1\_TX\_MID, 229  
CFS CFDP Event IDs, 152  
CF\_CC\_ERR\_EID, 158  
CF\_CFDP\_CLOSE\_ERR\_EID, 159  
CF\_CFDP\_DIR\_SLOT\_ERR\_EID, 159  
CF\_CFDP\_FD\_UNHANDLED\_ERR\_EID, 159  
CF\_CFDP\_IDLE\_MD\_ERR\_EID, 159  
CF\_CFDP\_INVALID\_DST\_ERR\_EID, 160  
CF\_CFDP\_MAX\_CMD\_TX\_ERR\_EID, 160  
CF\_CFDP\_NO\_CHUNKLIST\_AVAIL\_EID, 160  
CF\_CFDP\_NO\_MSG\_ERR\_EID, 160  
CF\_CFDP\_OPENDIR\_ERR\_EID, 161  
CF\_CFDP\_R\_ACK\_LIMIT\_ERR\_EID, 161  
CF\_CFDP\_R\_CRC\_ERR\_EID, 161  
CF\_CFDP\_R\_CREAT\_ERR\_EID, 161  
CF\_CFDP\_R\_DC\_INV\_ERR\_EID, 162  
CF\_CFDP\_R\_EOF\_MD\_SIZE\_ERR\_EID, 162  
CF\_CFDP\_R\_FILE\_RETAINED\_EID, 162  
CF\_CFDP\_R\_INACT\_TIMER\_ERR\_EID, 162  
CF\_CFDP\_R\_NAK\_LIMIT\_ERR\_EID, 163  
CF\_CFDP\_R\_NOT\_RETAINED\_EID, 163  
CF\_CFDP\_R\_PDU\_EOF\_ERR\_EID, 163  
CF\_CFDP\_R\_PDU\_FINACK\_ERR\_EID, 163  
CF\_CFDP\_R\_READ\_ERR\_EID, 164  
CF\_CFDP\_R\_RENAME\_ERR\_EID, 164  
CF\_CFDP\_R\_REQUEST\_MD\_INF\_EID, 164  
CF\_CFDP\_R\_SEEK\_CRC\_ERR\_EID, 164  
CF\_CFDP\_R\_SEEK\_FD\_ERR\_EID, 165  
CF\_CFDP\_R\_SIZE\_MISMATCH\_ERR\_EID, 165  
CF\_CFDP\_R\_TEMP\_FILE\_INF\_EID, 165  
CF\_CFDP\_R\_WRITE\_ERR\_EID, 165  
CF\_CFDP\_RX\_DROPPED\_ERR\_EID, 166  
CF\_CFDP\_S\_ACK\_LIMIT\_ERR\_EID, 166  
CF\_CFDP\_S\_ALREADY\_OPEN\_ERR\_EID, 166  
CF\_CFDP\_S\_DC\_INV\_ERR\_EID, 166  
CF\_CFDP\_S\_EARLY\_FIN\_ERR\_EID, 167  
CF\_CFDP\_S\_FILE\_MOVED\_EID, 167  
CF\_CFDP\_S\_FILE\_REMOVED\_EID, 167  
CF\_CFDP\_S\_INACT\_TIMER\_ERR\_EID, 167  
CF\_CFDP\_S\_INVALID\_SR\_ERR\_EID, 168  
CF\_CFDP\_S\_NON\_FD\_PDU\_ERR\_EID, 168  
CF\_CFDP\_S\_OPEN\_ERR\_EID, 168  
CF\_CFDP\_S\_PDU\_EOF\_ERR\_EID, 168  
CF\_CFDP\_S\_PDU\_NAK\_ERR\_EID, 169  
CF\_CFDP\_S\_READ\_ERR\_EID, 169  
CF\_CFDP\_S\_SEEK\_BEG\_ERR\_EID, 169  
CF\_CFDP\_S\_SEEK\_END\_ERR\_EID, 169  
CF\_CFDP\_S\_SEEK\_FD\_ERR\_EID, 170  
CF\_CFDP\_S\_SEND\_FD\_ERR\_EID, 170  
CF\_CFDP\_S\_SEND\_MD\_ERR\_EID, 170  
CF\_CFDP\_S\_START\_SEND\_INF\_EID, 170  
CF\_CMD\_ABANDON\_CHAN\_ERR\_EID, 171  
CF\_CMD\_ABANDON\_INF\_EID, 171  
CF\_CMD\_BAD\_PARAM\_ERR\_EID, 171  
CF\_CMD\_CANCEL\_CHAN\_ERR\_EID, 171  
CF\_CMD\_CANCEL\_INF\_EID, 172  
CF\_CMD\_CHAN\_PARAM\_ERR\_EID, 172  
CF\_CMD\_DISABLE\_DEQUEUE\_ERR\_EID, 172  
CF\_CMD\_DISABLE\_DEQUEUE\_INF\_EID, 172  
CF\_CMD\_DISABLE\_ENGINE\_INF\_EID, 173  
CF\_CMD\_DISABLE\_POLLDIR\_ERR\_EID, 173  
CF\_CMD\_DISABLE\_POLLDIR\_INF\_EID, 173  
CF\_CMD\_ENABLE\_DEQUEUE\_ERR\_EID, 173  
CF\_CMD\_ENABLE\_DEQUEUE\_INF\_EID, 174  
CF\_CMD\_ENABLE\_ENGINE\_ERR\_EID, 174  
CF\_CMD\_ENABLE\_ENGINE\_INF\_EID, 174  
CF\_CMD\_ENABLE\_POLLDIR\_ERR\_EID, 174  
CF\_CMD\_ENABLE\_POLLDIR\_INF\_EID, 175  
CF\_CMD\_ENG\_ALREADY\_DIS\_INF\_EID, 175  
CF\_CMD\_ENG\_ALREADY\_ENA\_INF\_EID, 175  
CF\_CMD\_FREEZE\_ERR\_EID, 175  
CF\_CMD\_FREEZE\_INF\_EID, 176  
CF\_CMD\_GETSET1\_INF\_EID, 176  
CF\_CMD\_GETSET2\_INF\_EID, 176  
CF\_CMD\_GETSET\_CHAN\_ERR\_EID, 176  
CF\_CMD\_GETSET\_PARAM\_ERR\_EID, 177  
CF\_CMD\_GETSET\_VALIDATE\_ERR\_EID, 177  
CF\_CMD\_LEN\_ERR\_EID, 177  
CF\_CMD\_PLAYBACK\_DIR\_ERR\_EID, 177  
CF\_CMD\_PLAYBACK\_DIR\_INF\_EID, 178  
CF\_CMD\_POLLDIR\_INVALID\_ERR\_EID, 178  
CF\_CMD\_PURGE\_ARG\_ERR\_EID, 178  
CF\_CMD\_PURGE\_QUEUE\_ERR\_EID, 178  
CF\_CMD\_PURGE\_QUEUE\_INF\_EID, 179  
CF\_CMD\_RESET\_INVALID\_ERR\_EID, 179  
CF\_CMD\_SUSPRES\_CHAN\_ERR\_EID, 179  
CF\_CMD\_SUSPRES\_INF\_EID, 179  
CF\_CMD\_SUSPRES\_SAME\_INF\_EID, 180  
CF\_CMD\_THAW\_ERR\_EID, 180  
CF\_CMD\_THAW\_INF\_EID, 180  
CF\_CMD\_TRANS\_NOT\_FOUND\_ERR\_EID, 180  
CF\_CMD\_TSN\_CHAN\_INVALID\_ERR\_EID, 181  
CF\_CMD\_TX\_FILE\_ERR\_EID, 181  
CF\_CMD\_TX\_FILE\_INF\_EID, 181  
CF\_CMD\_WHIST\_WRITE\_ERR\_EID, 181  
CF\_CMD\_WQ\_ARGS\_ERR\_EID, 182  
CF\_CMD\_WQ\_CHAN\_ERR\_EID, 182  
CF\_CMD\_WQ\_INF\_EID, 182  
CF\_CMD\_WQ\_OPEN\_ERR\_EID, 182  
CF\_CMD\_WQ\_WRITEHIST\_RX\_ERR\_EID, 183  
CF\_CMD\_WQ\_WRITEHIST\_TX\_ERR\_EID, 183

CF\_CMD\_WQ\_WRITEQ\_PEND\_ERR\_EID, 183  
 CF\_CMD\_WQ\_WRITEQ\_RX\_ERR\_EID, 183  
 CF\_CMD\_WQ\_WRITEQ\_TX\_ERR\_EID, 184  
 CF\_CR\_CHANNEL\_PIPE\_ERR\_EID, 184  
 CF\_CR\_PIPE\_ERR\_EID, 184  
 CF\_EID\_INF\_CFDP\_BUF\_EXCEED, 184  
 CF\_INIT\_CRC\_ALIGN\_ERR\_EID, 185  
 CF\_INIT\_INF\_EID, 185  
 CF\_INIT\_MSG\_RECV\_ERR\_EID, 185  
 CF\_INIT\_OUTGOING\_SIZE\_ERR\_EID, 185  
 CF\_INIT\_SEM\_ERR\_EID, 186  
 CF\_INIT\_SUB\_ERR\_EID, 186  
 CF\_INIT\_TBL\_CHECK\_GA\_ERR\_EID, 186  
 CF\_INIT\_TBL\_CHECK\_MAN\_ERR\_EID, 186  
 CF\_INIT\_TBL\_CHECK\_REL\_ERR\_EID, 187  
 CF\_INIT\_TBL\_GETADDR\_ERR\_EID, 187  
 CF\_INIT\_TBL\_LOAD\_ERR\_EID, 187  
 CF\_INIT\_TBL\_MANAGE\_ERR\_EID, 187  
 CF\_INIT\_TBL\_REG\_ERR\_EID, 188  
 CF\_INIT\_TPS\_ERR\_EID, 188  
 CF\_MID\_ERR\_EID, 188  
 CF\_NOOP\_INF\_EID, 188  
 CF\_PDU\_ACK\_SHORT\_ERR\_EID, 189  
 CF\_PDU\_EOF\_SHORT\_ERR\_EID, 189  
 CF\_PDU\_FD\_SHORT\_ERR\_EID, 189  
 CF\_PDU\_FD\_UNSUPPORTED\_ERR\_EID, 189  
 CF\_PDU\_FIN\_SHORT\_ERR\_EID, 190  
 CF\_PDU\_INVALID\_DST\_LEN\_ERR\_EID, 190  
 CF\_PDU\_INVALID\_SRC\_LEN\_ERR\_EID, 190  
 CF\_PDU\_LARGE\_FILE\_ERR\_EID, 190  
 CF\_PDU\_MD\_RECVD\_INF\_EID, 191  
 CF\_PDU\_MD\_SHORT\_ERR\_EID, 191  
 CF\_PDU\_NAK\_SHORT\_ERR\_EID, 191  
 CF\_PDU\_SHORT\_HEADER\_ERR\_EID, 191  
 CF\_PDU\_TRUNCATION\_ERR\_EID, 192  
 CF\_RESET\_FREED\_XACT\_DBG\_EID, 192  
 CF\_RESET\_INF\_EID, 192  
**CFS** CFDP Mission Configuration, 217  
 CF\_PERF\_ID\_APPMAIN, 218  
 CF\_PERF\_ID\_CREAT, 218  
 CF\_PERF\_ID\_CYCLE\_ENG, 218  
 CF\_PERF\_ID\_DIRREAD, 218  
 CF\_PERF\_ID\_FCLOSE, 218  
 CF\_PERF\_ID\_FOPEN, 218  
 CF\_PERF\_ID\_FREAD, 218  
 CF\_PERF\_ID\_FSEEK, 218  
 CF\_PERF\_ID\_FWRITE, 218  
 CF\_PERF\_ID\_PDURCVD, 219  
 CF\_PERF\_ID\_PDUSENT, 219  
 CF\_PERF\_ID\_RENAME, 219  
**CFS** CFDP Platform Configuration, 209  
 CF\_CONFIG\_TABLE\_FILENAME, 211  
 CF\_CONFIG\_TABLE\_NAME, 211  
 CF\_FILENAME\_MAX\_LEN, 211

CF\_FILENAME\_MAX\_NAME, 211  
 CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN, 212  
 CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN, 212  
 CF\_MAX\_PDU\_SIZE, 212  
 CF\_MAX\_POLLING\_DIR\_PER\_CHAN, 212  
 CF\_MAX\_SIMULTANEOUS\_RX, 213  
 CF\_NAK\_MAX\_SEGMENTS, 213  
 CF\_NUM\_CHANNELS, 213  
 CF\_NUM\_HISTORIES\_PER\_CHANNEL, 213  
 CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK, 214  
 CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES, 214  
 CF\_PIPE\_DEPTH, 214  
 CF\_R2\_CRC\_CHUNK\_SIZE, 215  
 CF\_RCVMMSG\_TIMEOUT, 215  
 CF\_STARTUP\_SEM\_MAX\_RETRIES, 215  
 CF\_STARTUP\_SEM\_TASK\_DELAY, 215  
 DEFAULT\_CF\_CONFIG\_TABLE\_FILENAME, 215  
 DEFAULT\_CF\_CONFIG\_TABLE\_NAME, 216  
 DEFAULT\_CF\_FILENAME\_MAX\_LEN, 216  
 DEFAULT\_CF\_FILENAME\_MAX\_NAME, 216  
 DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN, 216  
 DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN, 216  
 DEFAULT\_CF\_MAX\_PDU\_SIZE, 216  
 DEFAULT\_CF\_MAX\_POLLING\_DIR\_PER\_CHAN, 216  
 DEFAULT\_CF\_MAX\_SIMULTANEOUS\_RX, 216  
 DEFAULT\_CF\_NAK\_MAX\_SEGMENTS, 216  
 DEFAULT\_CF\_NUM\_CHANNELS, 216  
 DEFAULT\_CF\_NUM\_HISTORIES\_PER\_CHANNEL, 216  
 DEFAULT\_CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK, 216  
 DEFAULT\_CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES, 217  
 DEFAULT\_CF\_PIPE\_DEPTH, 217  
 DEFAULT\_CF\_R2\_CRC\_CHUNK\_SIZE, 217  
 DEFAULT\_CF\_RCVMMSG\_TIMEOUT, 217  
 DEFAULT\_CF\_STARTUP\_SEM\_MAX\_RETRIES, 217  
 DEFAULT\_CF\_STARTUP\_SEM\_TASK\_DELAY, 217  
**CFS** CFDP Telemetry, 220  
 CF\_EotPacket\_Payload\_t, 221  
 CF\_EotPacket\_t, 221  
 CF\_HkChannel\_Data\_t, 221  
 CF\_HkCmdCounters\_t, 221  
 CF\_HkCounters\_t, 221  
 CF\_HkFault\_t, 221  
 CF\_HkPacket\_Payload\_t, 221  
 CF\_HkPacket\_t, 221

CF\_HkRecv\_t, 221  
CF\_HkSent\_t, 221  
CFS CFDP Telemetry Message IDs, 228  
CF\_EOT\_TLM\_MID, 229  
CF\_HK\_TLM\_MID, 229  
CFS CFDP Version, 219  
CF\_MAJOR\_VERSION, 219  
CF\_MINOR\_VERSION, 219  
CF\_REVISION, 219  
chan  
CF\_CFDP\_Tick\_args, 536  
CF\_ConfigTable, 551  
CF\_Transaction\_Payload, 617  
CF\_WriteQueue\_Payload, 626  
chan\_num  
CF\_GetParam\_Payload, 567  
CF\_SetParam\_Payload, 607  
CF\_Transaction, 614  
CF\_TxFile\_Payload, 623  
channel  
CF\_EotPacket\_Payload, 560  
channel\_hk  
CF\_HkPacket\_Payload, 577  
channels  
CF\_Engine, 558  
CheckErrCtr  
CFE\_ES\_MemPoolStats, 652  
checksum\_type  
CF\_Logical\_PduMd, 592  
chunk\_mem  
CF\_Engine, 558  
chunks  
CF\_ChunkList, 547  
CF\_ChunkWrapper, 548  
CF\_Engine, 558  
CF\_Transaction, 614  
cl\_node  
CF\_ChunkWrapper, 548  
CF\_History, 569  
CF\_Transaction, 614  
ClockFlyState  
CFE\_TIME\_DiagnosticTlm\_Payload, 760  
ClockSetState  
CFE\_TIME\_DiagnosticTlm\_Payload, 760  
ClockSignal  
CFE\_TIME\_DiagnosticTlm\_Payload, 760  
ClockSource  
CFE\_TIME\_DiagnosticTlm\_Payload, 761  
ClockState  
CFE\_TIME\_StateCmd\_Payload, 777  
ClockStateAPI  
CFE\_TIME\_DiagnosticTlm\_Payload, 761  
CFE\_TIME\_HousekeepingTlm\_Payload, 767  
ClockStateFlags  
CFE\_TIME\_DiagnosticTlm\_Payload, 761  
CFE\_TIME\_HousekeepingTlm\_Payload, 767  
close\_req  
CF\_Flags\_Common, 562  
CF\_Logical\_PduMd, 592  
cmd  
CF\_HkCmdCounters, 572  
cmd\_tx  
CF\_Flags\_Tx, 565  
CmdPipe  
CF\_AppData\_t, 525  
code\_address  
OS\_module\_address\_t, 789  
code\_size  
OS\_module\_address\_t, 789  
CodeAddress  
CFE\_ES\_AppInfo, 631  
codec\_state  
CF\_DecoderState, 553  
CF\_EncoderState, 557  
CodeSize  
CFE\_ES\_AppInfo, 631  
com  
CF\_Flags\_Rx, 563  
CF\_Flags\_Tx, 565  
CF\_StateFlags, 610  
CommandCounter  
CFE\_ES\_HousekeepingTlm\_Payload, 644  
CFE\_EVS\_HousekeepingTlm\_Payload, 688  
CFE\_SB\_HousekeepingTlm\_Payload, 709  
CFE\_TBL\_HousekeepingTlm\_Payload, 741  
CFE\_TIME\_HousekeepingTlm\_Payload, 768  
CommandErrorCounter  
CFE\_ES\_HousekeepingTlm\_Payload, 645  
CFE\_EVS\_HousekeepingTlm\_Payload, 688  
CFE\_SB\_HousekeepingTlm\_Payload, 709  
CFE\_TBL\_HousekeepingTlm\_Payload, 741  
CFE\_TIME\_HousekeepingTlm\_Payload, 768  
CommandHeader  
CF\_AbandonCmd, 524  
CF\_CancelCmd, 526  
CF\_DisableDequeueCmd, 553  
CF\_DisableDirPollingCmd, 554  
CF\_DisableEngineCmd, 554  
CF\_EnableDequeueCmd, 555  
CF\_EnableDirPollingCmd, 556  
CF\_EnableEngineCmd, 556  
CF\_FreezeCmd, 566  
CF\_GetParamCmd, 568  
CF\_NoopCmd, 597  
CF\_PlaybackDirCmd, 602  
CF\_PurgeQueueCmd, 605  
CF\_ResetCountersCmd, 605  
CF\_ResumeCmd, 606

CFE\_SendHkCmd, 607  
 CFE\_SetParamCmd, 608  
 CFE\_SuspendCmd, 611  
 CFE\_ThawCmd, 612  
 CFE\_TxFileCmd, 624  
 CFE\_WakeupCmd, 626  
 CFE\_WriteQueueCmd, 627  
 CFE\_ES\_ClearERLogCmd, 637  
 CFE\_ES\_ClearSysLogCmd, 638  
 CFE\_ES\_DeleteCDSCmd, 638  
 CFE\_ES\_DumpCDSRegistryCmd, 639  
 CFE\_ES\_FileNameCmd, 640  
 CFE\_ES\_NoopCmd, 653  
 CFE\_ES\_OverWriteSysLogCmd, 655  
 CFE\_ES\_QueryAllCmd, 657  
 CFE\_ES\_QueryAllTasksCmd, 658  
 CFE\_ES\_QueryOneCmd, 658  
 CFE\_ES\_ReloadAppCmd, 659  
 CFE\_ES\_ResetCountersCmd, 660  
 CFE\_ES\_ResetPRCountCmd, 660  
 CFE\_ES\_RestartAppCmd, 660  
 CFE\_ES\_RestartCmd, 661  
 CFE\_ES\_SendHkCmd, 662  
 CFE\_ES\_SendMemPoolStatsCmd, 663  
 CFE\_ES\_SetMaxPRCountCmd, 664  
 CFE\_ES\_SetPerfFilterMaskCmd, 665  
 CFE\_ES\_SetPerfTriggerMaskCmd, 666  
 CFE\_ES\_StartApp, 667  
 CFE\_ES\_StartPerfDataCmd, 670  
 CFE\_ES\_StopAppCmd, 670  
 CFE\_ES\_StopPerfDataCmd, 671  
 CFE\_ES\_WriteERLogCmd, 674  
 CFE\_ES\_WriteSysLogCmd, 674  
 CFE\_EVS\_AddEventFilterCmd, 675  
 CFE\_EVS\_ClearLogCmd, 681  
 CFE\_EVS\_DeleteEventFilterCmd, 681  
 CFE\_EVS\_DisableAppEventsCmd, 682  
 CFE\_EVS\_DisableAppEventTypeCmd, 682  
 CFE\_EVS\_DisableEventTypeCmd, 683  
 CFE\_EVS\_DisablePortsCmd, 684  
 CFE\_EVS\_EnableAppEventsCmd, 684  
 CFE\_EVS\_EnableAppEventTypeCmd, 685  
 CFE\_EVS\_EnableEventTypeCmd, 685  
 CFE\_EVS\_EnablePortsCmd, 686  
 CFE\_EVS\_NoopCmd, 692  
 CFE\_EVS\_ResetAllFiltersCmd, 694  
 CFE\_EVS\_ResetAppCounterCmd, 695  
 CFE\_EVS\_ResetCountersCmd, 695  
 CFE\_EVS\_ResetFilterCmd, 696  
 CFE\_EVS\_SendHkCmd, 696  
 CFE\_EVS\_SetEventFormatModeCmd, 697  
 CFE\_EVS\_SetFilterCmd, 698  
 CFE\_EVS\_SetLogModeCmd, 699  
 CFE\_EVS\_WriteAppDataFileCmd, 701  
 CFE\_EVS\_WriteLogFileCmd, 701  
 CFE\_SB\_DisableRouteCmd, 706  
 CFE\_SB\_DisableSubReportingCmd, 707  
 CFE\_SB\_EnableRouteCmd, 707  
 CFE\_SB\_EnableSubReportingCmd, 708  
 CFE\_SB\_NoopCmd, 714  
 CFE\_SB\_ResetCountersCmd, 718  
 CFE\_SB\_SendHkCmd, 720  
 CFE\_SB\_SendPrevSubsCmd, 721  
 CFE\_SB\_SendSbStatsCmd, 721  
 CFE\_SB\_WriteMapInfoCmd, 728  
 CFE\_SB\_WritePipeInfoCmd, 729  
 CFE\_SB\_WriteRoutingInfoCmd, 730  
 CFE\_TBL\_AbortLoadCmd, 730  
 CFE\_TBL\_ActivateCmd, 731  
 CFE\_TBL\_DeleteCDSCmd, 734  
 CFE\_TBL\_DumpCmd, 734  
 CFE\_TBL\_DumpRegistryCmd, 736  
 CFE\_TBL\_LoadCmd, 746  
 CFE\_TBL\_NoopCmd, 747  
 CFE\_TBL\_NotifyCmd, 747  
 CFE\_TBL\_ResetCountersCmd, 748  
 CFE\_TBL\_SendHkCmd, 749  
 CFE\_TBL\_SendRegistryCmd, 749  
 CFE\_TBL\_ValidateCmd, 754  
 CFE\_TIME\_AddAdjustCmd, 755  
 CFE\_TIME\_AddDelayCmd, 756  
 CFE\_TIME\_AddOneHzAdjustmentCmd, 757  
 CFE\_TIME\_FakeToneCmd, 766  
 CFE\_TIME\_NoopCmd, 769  
 CFE\_TIME\_OneHzCmd, 770  
 CFE\_TIME\_ResetCountersCmd, 771  
 CFE\_TIME\_SendDiagnosticCmd, 771  
 CFE\_TIME\_SendHkCmd, 771  
 CFE\_TIME\_SetLeapSecondsCmd, 772  
 CFE\_TIME\_SetMETCmd, 773  
 CFE\_TIME\_SetSignalCmd, 773  
 CFE\_TIME\_SetSourceCmd, 774  
 CFE\_TIME\_SetStateCmd, 774  
 CFE\_TIME\_SetSTCFCmd, 775  
 CFE\_TIME\_SetTimeCmd, 776  
 CFE\_TIME\_SubAdjustCmd, 778  
 CFE\_TIME\_SubDelayCmd, 778  
 CFE\_TIME\_SubOneHzAdjustmentCmd, 779  
 CFE\_TIME\_ToneDataCmd, 781  
 CFE\_TIME\_ToneSignalCmd, 782  
 common\_types.h  
     \_EXTENSION\_, 1747  
     CompileTimeAssert, 1747, 1750, 1751  
     cpuaddr, 1748  
     cpudiff, 1748  
     cpusize, 1748  
     int16, 1749  
     int32, 1749

int64, 1749  
int8, 1749  
intptr, 1749  
OS\_ArgCallback\_t, 1749  
OS\_PRINTF, 1748  
OS\_USED, 1748  
OSAL\_BLOCKCOUNT\_C, 1748  
osal\_blockcount\_t, 1749  
osal\_id\_t, 1749  
OSAL\_INDEX\_C, 1748  
osal\_index\_t, 1749  
OSAL\_OBJTYPE\_C, 1748  
osal\_objtype\_t, 1749  
osal\_offset\_t, 1750  
OSAL\_SIZE\_C, 1748  
OSAL\_STATUS\_C, 1748  
osal\_status\_t, 1750  
uint16, 1750  
uint32, 1750  
uint64, 1750  
uint8, 1750  
CompileTimeAssert  
    common\_types.h, 1747, 1750, 1751  
config\_handle  
    CF\_AppData\_t, 525  
config\_table  
    CF\_AppData\_t, 525  
container\_of  
    cf\_clist.h, 1069  
content\_crc  
    CF\_Logical\_PduBuffer, 584  
ContentType  
    CFE\_FS\_Header, 703  
context  
    CF\_TraverseAll\_Arg, 621  
continuation\_state  
    CF\_Logical\_PduFileDataHeader, 587  
count  
    CF\_ChunkList, 547  
counted  
    CF\_Playback, 601  
counter  
    CF\_Traverse\_WriteHistoryFileArg, 620  
    CF\_Traverse\_WriteTxnFileArg, 621  
    CF\_TraverseAll\_Arg, 622  
counters  
    CF\_HkChannel\_Data, 571  
    CF\_HkPacket\_Payload, 577  
cpuaddr  
    common\_types.h, 1748  
cpudiff  
    common\_types.h, 1748  
cpusize  
    common\_types.h, 1748  
Crc  
    CFE\_TBL\_Info, 744  
    CFE\_TBL\_TblRegPacket\_Payload, 752  
crc  
    CF\_CFDP\_PduEof, 529  
    CF\_Logical\_PduEof, 586  
    CF\_Transaction, 614  
crc\_complete  
    CF\_Flags\_Common, 562  
crc\_flag  
    CF\_Logical\_PduHeader, 590  
crc\_mismatch  
    CF\_HkFault, 574  
crc\_result  
    CF\_EotPacket\_Payload, 560  
CreatePipeErrorCounter  
    CFE\_SB\_HousekeepingTlm\_Payload, 710  
creator  
    OS\_bin\_sem\_prop\_t, 783  
    OS\_condvar\_prop\_t, 783  
    OS\_count\_sem\_prop\_t, 784  
    OS\_mut\_sem\_prop\_t, 790  
    OS\_queue\_prop\_t, 791  
    OS\_rwlock\_prop\_t, 792  
    OS\_socket\_prop\_t, 794  
    OS\_task\_prop\_t, 796  
    OS\_timebase\_prop\_t, 797  
    OS\_timer\_prop\_t, 798  
Critical  
    CFE\_TBL\_Info, 744  
    CFE\_TBL\_TblRegPacket\_Payload, 752  
cs  
    CF\_Channel, 542  
CurrentLatch  
    CFE\_TIME\_DiagnosticTlm\_Payload, 761  
CurrentMET  
    CFE\_TIME\_DiagnosticTlm\_Payload, 761  
CurrentQueueDepth  
    CFE\_SB\_PipeDepthStats, 715  
    CFE\_SB\_PipeInfoEntry, 716  
CurrentTAI  
    CFE\_TIME\_DiagnosticTlm\_Payload, 761  
CurrentUTC  
    CFE\_TIME\_DiagnosticTlm\_Payload, 761  
data  
    CF\_CFDP\_PduFileDataContent, 529  
    CF\_ChAction\_BoolMsgArg, 540  
    CF\_ChAction\_MsgArg, 541  
    CF\_Logical\_Tlv, 595  
data\_address  
    OS\_module\_address\_t, 789  
data\_encoded\_length  
    CF\_Logical\_PduHeader, 590

data\_len  
   CF\_Logical\_PduFileDataHeader, 587  
 data\_ptr  
   CF\_Logical\_Lv, 582  
   CF\_Logical\_PduFileDataHeader, 587  
   CF\_Logical\_TlvData, 596  
 data\_size  
   OS\_module\_address\_t, 789  
 DataAddress  
   CFE\_ES\_AppInfo, 631  
 DataFileName  
   CFE\_ES\_StopPerfCmd\_Payload, 671  
 DataSize  
   CFE\_ES\_AppInfo, 632  
 DataStoreStatus  
   CFE\_TIME\_DiagnosticTlm\_Payload, 762  
 Datum  
   CFE\_Config\_ValueEntry, 629  
 decode  
   CF\_Input, 580  
 DEFAULT\_CF\_CONFIG\_TABLE\_FILENAME  
   CFS CFDP Platform Configuration, 215  
 DEFAULT\_CF\_CONFIG\_TABLE\_NAME  
   CFS CFDP Platform Configuration, 216  
 default\_cf\_extern\_typedefs.h  
   CF\_CFDP\_CLASS\_1, 800  
   CF\_CFDP\_CLASS\_2, 800  
   CF\_CFDP\_Class\_t, 800  
   CF\_EntityId\_t, 799  
   CF\_QueueIdx\_FREE, 801  
   CF\_QueueIdx\_HIST, 801  
   CF\_QueueIdx\_HIST\_FREE, 801  
   CF\_QueueIdx\_NUM, 801  
   CF\_QueueIdx\_PEND, 801  
   CF\_QueueIdx\_RX, 801  
   CF\_QueueIdx\_t, 801  
   CF\_QueueIdx\_TX, 801  
   CF\_TransactionSeq\_t, 800  
   CF\_TxnFilenames\_t, 800  
 default\_cf\_fcncode\_values.h  
   CF\_CCVAL, 801  
   CF\_FunctionCode\_, 802  
   CF\_FunctionCode\_ABANDON, 802  
   CF\_FunctionCode\_CANCEL, 802  
   CF\_FunctionCode\_DISABLE\_DEQUEUE, 802  
   CF\_FunctionCode\_DISABLE\_DIR\_POLLING, 802  
   CF\_FunctionCode\_DISABLE\_ENGINE, 802  
   CF\_FunctionCode\_ENABLE\_DEQUEUE, 802  
   CF\_FunctionCode\_ENABLE\_DIR\_POLLING, 802  
   CF\_FunctionCode\_ENABLE\_ENGINE, 802  
   CF\_FunctionCode\_FREEZE, 802  
   CF\_FunctionCode\_GET\_PARAM, 802  
   CF\_FunctionCode\_NOOP, 802  
   CF\_FunctionCode\_PLAYBACK\_DIR, 802  
 CF\_FunctionCode\_PURGE\_QUEUE, 802  
 CF\_FunctionCode\_RESET\_COUNTERS, 802  
 CF\_FunctionCode\_RESUME, 802  
 CF\_FunctionCode\_SET\_PARAM, 802  
 CF\_FunctionCode\_SUSPEND, 802  
 CF\_FunctionCode\_THAW, 802  
 CF\_FunctionCode\_TX\_FILE, 802  
 CF\_FunctionCode\_WRITE\_QUEUE, 802  
 DEFAULT\_CF\_FILENAME\_MAX\_LEN  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_FILENAME\_MAX\_NAME  
   CFS CFDP Platform Configuration, 216  
 default\_cf\_interface\_cfg\_values.h  
   CF\_INTERFACE\_CFGVAL, 802  
 default\_cf\_internal\_cfg\_values.h  
   CF\_INTERNAL\_CFGVAL, 803  
 DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_DIRECTORIES\_PER\_CHAN  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_MAX\_COMMANDED\_PLAYBACK\_FILES\_PER\_CHAN  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_MAX\_PDU\_SIZE  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_MAX\_POLLING\_DIR\_PER\_CHAN  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_MAX\_SIMULTANEOUS\_RX  
   CFS CFDP Platform Configuration, 216  
 default\_cf\_mission\_cfg.h  
   CF\_FILENAME\_MAX\_PATH, 803  
 default\_cf\_msg.h  
   CF\_ALL\_CHANNELS, 804  
   CF\_ALL\_POLLDIRS, 804  
   CF\_COMPOUND\_KEY, 804  
 default\_cf\_msgid\_values.h  
   CFE\_PLATFORM\_CF\_CMD\_MIDVAL, 807  
   CFE\_PLATFORM\_CF\_TLM\_MIDVAL, 807  
 DEFAULT\_CF\_NAK\_MAX\_SEGMENTS  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_NUM\_CHANNELS  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_NUM\_HISTORIES\_PER\_CHANNEL  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_NUM\_TRANSACTIONS\_PER\_PLAYBACK  
   CFS CFDP Platform Configuration, 216  
 DEFAULT\_CF\_PDU\_ENCAPSULATION\_EXTRA\_TRAILING\_BYTES  
   CFS CFDP Platform Configuration, 217  
 DEFAULT\_CF\_PIPE\_DEPTH  
   CFS CFDP Platform Configuration, 217  
 default\_cf\_platform\_cfg.h  
   CF\_CHANNEL\_NUM\_RX\_CHUNKS\_PER\_TRANSACTION,  
     810  
   CF\_CHANNEL\_NUM\_TX\_CHUNKS\_PER\_TRANSACTION,  
     811  
   CF\_MISSION\_REV, 811  
   CF\_TOTAL\_CHUNKS, 811

DEFAULT\_CF\_R2\_CRC\_CHUNK\_SIZE  
    CFS CFDP Platform Configuration, 217

DEFAULT\_CF\_RCVMSG\_TIMEOUT  
    CFS CFDP Platform Configuration, 217

DEFAULT\_CF\_STARTUP\_SEM\_MAX\_RETRIES  
    CFS CFDP Platform Configuration, 217

DEFAULT\_CF\_STARTUP\_SEM\_TASK\_DELAY  
    CFS CFDP Platform Configuration, 217

default\_cf\_tbldefs.h  
    CF\_ChannelConfig\_t, 812  
    CF\_PollDir\_t, 812

default\_cf\_tblstruct.h  
    CF\_ConfigTable\_t, 813

default\_cf\_topicid\_values.h  
    CFE\_MISSION\_CF\_TIDVAL, 813

default\_cfe\_core\_api\_base\_msgid\_values.h  
    CFE\_GLOBAL\_BASE\_MIDVAL, 1333  
    CFE\_PLATFORM\_BASE\_MIDVAL, 1333

default\_cfe\_core\_api\_interface\_cfg\_values.h  
    CFE\_MISSION\_CORE\_API\_CFGVAL, 1333

default\_cfe\_core\_api\_msgid\_mapping.h  
    CFE\_GLOBAL\_CMD\_TOPICID\_TO\_MIDV, 1334  
    CFE\_GLOBAL\_TLM\_TOPICID\_TO\_MIDV, 1334  
    CFE\_PLATFORM\_CMD\_TOPICID\_TO\_MIDV, 1334  
    CFE\_PLATFORM\_TLM\_TOPICID\_TO\_MIDV, 1335

DEFAULT\_CFE\_PLATFORM\_CMD\_MID\_BASE,  
    1335

DEFAULT\_CFE\_PLATFORM\_TLM\_MID\_BASE,  
    1335

DEFAULT\_GLOBAL\_CMD\_MID\_BASE, 1335

DEFAULT\_GLOBAL\_TLM\_MID\_BASE, 1335

default\_cfe\_es\_extern\_typedefs.h  
    CFE\_ES\_AppId\_t, 1403  
    CFE\_ES\_AppInfo\_t, 1403  
    CFE\_ES\_AppState, 1406  
    CFE\_ES\_AppState\_EARLY\_INIT, 1406  
    CFE\_ES\_AppState\_Enum\_t, 1403  
    CFE\_ES\_AppState\_LATE\_INIT, 1406  
    CFE\_ES\_AppState\_MAX, 1406  
    CFE\_ES\_AppState\_RUNNING, 1406  
    CFE\_ES\_AppState\_STOPPED, 1406  
    CFE\_ES\_AppState\_UNDEFINED, 1406  
    CFE\_ES\_AppState\_WAITING, 1406  
    CFE\_ES\_AppType, 1406  
    CFE\_ES\_AppType\_CORE, 1407  
    CFE\_ES\_AppType\_Enum\_t, 1403  
    CFE\_ES\_AppType\_EXTERNAL, 1407  
    CFE\_ES\_AppType\_LIBRARY, 1407  
    CFE\_ES\_BlockStats\_t, 1404  
    CFE\_ES\_CDSHandle\_t, 1404  
    CFE\_ES\_CDSRegDumpRec\_t, 1404  
    CFE\_ES\_CounterId\_t, 1404  
    CFE\_ES\_ExceptionAction, 1407  
    CFE\_ES\_ExceptionAction\_Enum\_t, 1404

CFE\_ES\_ExceptionAction\_PROC\_RESTART, 1407  
CFE\_ES\_ExceptionAction\_RESTART\_APP, 1407

CFE\_ES\_LibId\_t, 1404

CFE\_ES\_LogEntryType, 1407

CFE\_ES\_LogEntryType\_APPLICATION, 1407

CFE\_ES\_LogEntryType\_CORE, 1407

CFE\_ES\_LogEntryType\_Enum\_t, 1404

CFE\_ES\_LogMode, 1407

CFE\_ES\_LogMode\_DISCARD, 1407

CFE\_ES\_LogMode\_Enum\_t, 1405

CFE\_ES\_LogMode\_OVERWRITE, 1407

CFE\_ES\_MemHandle\_t, 1405

CFE\_ES\_MemPoolStats\_t, 1405

CFE\_ES\_RunStatus, 1407

CFE\_ES\_RunStatus\_APP\_ERROR, 1408

CFE\_ES\_RunStatus\_APP\_EXIT, 1407

CFE\_ES\_RunStatus\_APP\_RUN, 1407

CFE\_ES\_RunStatus\_CORE\_APP\_INIT\_ERROR,  
    1408

CFE\_ES\_RunStatus\_CORE\_APP\_RUNTIME\_ERROR,  
    1408

CFE\_ES\_RunStatus\_Enum\_t, 1405

CFE\_ES\_RunStatus\_MAX, 1408

CFE\_ES\_RunStatus\_SYS\_DELETE, 1408

CFE\_ES\_RunStatus\_SYS\_EXCEPTION, 1408

CFE\_ES\_RunStatus\_SYS\_RELOAD, 1408

CFE\_ES\_RunStatus\_SYS\_RESTART, 1408

CFE\_ES\_RunStatus\_UNDEFINED, 1407

CFE\_ES\_SystemState, 1408

CFE\_ES\_SystemState\_APPS\_INIT, 1408

CFE\_ES\_SystemState\_CORE\_READY, 1408

CFE\_ES\_SystemState\_CORE\_STARTUP, 1408

CFE\_ES\_SystemState\_EARLY\_INIT, 1408

CFE\_ES\_SystemState\_Enum\_t, 1405

CFE\_ES\_SystemState\_MAX, 1408

CFE\_ES\_SystemState\_OPERATIONAL, 1408

CFE\_ES\_SystemState\_SHUTDOWN, 1408

CFE\_ES\_SystemState\_UNDEFINED, 1408

CFE\_ES\_TaskId\_t, 1405

CFE\_ES\_TaskInfo\_t, 1406

CFE\_ES\_TaskPriority\_Atom\_t, 1406

default\_cfe\_es\_fcncode\_values.h  
    CFE\_ES\_CCVAL, 1409  
    CFE\_ES\_FunctionCode\_, 1409  
    CFE\_ES\_FunctionCode\_CLEAR\_ER\_LOG, 1409  
    CFE\_ES\_FunctionCode\_CLEAR\_SYS\_LOG, 1409  
    CFE\_ES\_FunctionCode\_DELETE\_CDS, 1409  
    CFE\_ES\_FunctionCode\_DUMP\_CDS\_REGISTRY,  
        1409  
    CFE\_ES\_FunctionCode\_NOOP, 1409  
    CFE\_ES\_FunctionCode\_OVER\_WRITE\_SYS\_LOG,  
        1409  
    CFE\_ES\_FunctionCode\_QUERY\_ALL, 1409

CFE\_ES\_FunctionCode\_QUERY\_ALL\_TASKS,  
1409  
 CFE\_ES\_FunctionCode\_QUERY\_ONE, 1409  
 CFE\_ES\_FunctionCode\_RELOAD\_APP, 1409  
 CFE\_ES\_FunctionCode\_RESET\_COUNTERS, 1409  
 CFE\_ES\_FunctionCode\_RESET\_PR\_COUNT, 1409  
 CFE\_ES\_FunctionCode\_RESTART, 1409  
 CFE\_ES\_FunctionCode\_RESTART\_APP, 1409  
 CFE\_ES\_FunctionCode\_SEND\_MEM\_POOL\_STATS,  
1409  
 CFE\_ES\_FunctionCode\_SET\_MAX\_PR\_COUNT,  
1409  
 CFE\_ES\_FunctionCode\_SET\_PERF\_FILTER\_MASK,  
1409  
 CFE\_ES\_FunctionCode\_SET\_PERF\_TRIGGER\_MASK,  
1409  
 CFE\_ES\_FunctionCode\_START\_APP, 1409  
 CFE\_ES\_FunctionCode\_START\_PERF\_DATA, 1409  
 CFE\_ES\_FunctionCode\_STOP\_APP, 1409  
 CFE\_ES\_FunctionCode\_STOP\_PERF\_DATA, 1409  
 CFE\_ES\_FunctionCode\_WRITE\_ER\_LOG, 1409  
 CFE\_ES\_FunctionCode\_WRITE\_SYS\_LOG, 1409  
 default\_cfe\_es\_interface\_cfg\_values.h  
 CFE\_MISSION\_ES\_CFGVAL, 1410  
 default\_cfe\_es\_internal\_cfg\_values.h  
 CFE\_PLATFORM\_ES\_CFGVAL, 1410  
 default\_cfe\_es\_memaddress.h  
 CFE\_ES\_MEMADDRESS\_C, 1411  
 CFE\_ES\_MemAddress\_t, 1412  
 CFE\_ES\_MEMADDRESS\_TO\_PTR, 1411  
 CFE\_ES\_MEMOFFSET\_C, 1411  
 CFE\_ES\_MemOffset\_t, 1412  
 CFE\_ES\_MEMOFFSET\_TO\_SIZE, 1411  
 default\_cfe\_es\_msgdefs.h  
 CFE\_ES\_AppNameCmd\_Payload\_t, 1414  
 CFE\_ES\_AppReloadCmd\_Payload\_t, 1414  
 CFE\_ES\_DeleteCDSCmd\_Payload\_t, 1414  
 CFE\_ES\_DumpCDSRegistryCmd\_Payload\_t, 1415  
 CFE\_ES\_FileNameCmd\_Payload\_t, 1415  
 CFE\_ES\_HousekeepingTlm\_Payload\_t, 1415  
 CFE\_ES\_OneAppTlm\_Payload\_t, 1415  
 CFE\_ES\_OverWriteSysLogCmd\_Payload\_t, 1415  
 CFE\_ES\_PerfMode, 1416  
 CFE\_ES\_PerfMode\_Enum\_t, 1415  
 CFE\_ES\_PerfTrigger\_CENTER, 1416  
 CFE\_ES\_PerfTrigger\_END, 1416  
 CFE\_ES\_PerfTrigger\_START, 1416  
 CFE\_ES\_PoolStatsTlm\_Payload\_t, 1415  
 CFE\_ES\_RestartCmd\_Payload\_t, 1415  
 CFE\_ES\_SendMemPoolStatsCmd\_Payload\_t, 1415  
 CFE\_ES\_SetMaxPRCountCmd\_Payload\_t, 1415  
 CFE\_ES\_SetPerfFilterMaskCmd\_Payload\_t, 1415  
 CFE\_ES\_SetPerfTrigMaskCmd\_Payload\_t, 1416  
 CFE\_ES\_StartAppCmd\_Payload\_t, 1416  
 CFE\_ES\_StartPerfCmd\_Payload\_t, 1416  
 CFE\_ES\_StopPerfCmd\_Payload\_t, 1416  
 default\_cfe\_es\_msgid\_values.h  
 CFE\_PLATFORM\_ES\_CMD\_MIDVAL, 1417  
 CFE\_PLATFORM\_ES\_TLM\_MIDVAL, 1417  
 default\_cfe\_es\_msgids.h  
 CFE\_ES\_APP\_TLM\_MID, 1417  
 CFE\_ES\_CMD\_MID, 1417  
 CFE\_ES\_HK\_TLM\_MID, 1417  
 CFE\_ES\_MEMSTATS\_TLM\_MID, 1417  
 CFE\_ES\_SEND\_HK\_MID, 1417  
 default\_cfe\_es\_msgstruct.h  
 CFE\_ES\_ClearERLogCmd\_t, 1420  
 CFE\_ES\_ClearSysLogCmd\_t, 1420  
 CFE\_ES\_DeleteCDSCmd\_t, 1420  
 CFE\_ES\_DumpCDSRegistryCmd\_t, 1420  
 CFE\_ES\_FileNameCmd\_t, 1420  
 CFE\_ES\_HousekeepingTlm\_t, 1420  
 CFE\_ES\_MemStatsTlm\_t, 1420  
 CFE\_ES\_NoopCmd\_t, 1420  
 CFE\_ES\_OneAppTlm\_t, 1420  
 CFE\_ES\_OverWriteSysLogCmd\_t, 1420  
 CFE\_ES\_QueryAllCmd\_t, 1420  
 CFE\_ES\_QueryAllTasksCmd\_t, 1420  
 CFE\_ES\_QueryOneCmd\_t, 1421  
 CFE\_ES\_ReloadAppCmd\_t, 1421  
 CFE\_ES\_ResetCountersCmd\_t, 1421  
 CFE\_ES\_ResetPRCountCmd\_t, 1421  
 CFE\_ES\_RestartAppCmd\_t, 1421  
 CFE\_ES\_RestartCmd\_t, 1421  
 CFE\_ES\_SendHkCmd\_t, 1421  
 CFE\_ES\_SendMemPoolStatsCmd\_t, 1421  
 CFE\_ES\_SetMaxPRCountCmd\_t, 1421  
 CFE\_ES\_SetPerfFilterMaskCmd\_t, 1421  
 CFE\_ES\_SetPerfTriggerMaskCmd\_t, 1421  
 CFE\_ES\_StartAppCmd\_t, 1421  
 CFE\_ES\_StartPerfDataCmd\_t, 1421  
 CFE\_ES\_StopAppCmd\_t, 1421  
 CFE\_ES\_StopPerfDataCmd\_t, 1422  
 CFE\_ES\_WriteERLogCmd\_t, 1422  
 CFE\_ES\_WriteSysLogCmd\_t, 1422  
 default\_cfe\_es\_topicid\_values.h  
 CFE\_MISSION\_ES\_TIDVAL, 1422  
 default\_cfe\_evs\_extern\_typedefs.h  
 CFE\_EVS\_EventFilter, 1508  
 CFE\_EVS\_EventFilter\_BINARY, 1508  
 CFE\_EVS\_EventFilter\_Enum\_t, 1507  
 CFE\_EVS\_EventOutput, 1508  
 CFE\_EVS\_EventOutput\_Enum\_t, 1507  
 CFE\_EVS\_EventOutput\_PORT1, 1508  
 CFE\_EVS\_EventOutput\_PORT2, 1508  
 CFE\_EVS\_EventOutput\_PORT3, 1508  
 CFE\_EVS\_EventOutput\_PORT4, 1508  
 CFE\_EVS\_EventType, 1508

CFE\_EVS\_EventType\_CRITICAL, 1509  
CFE\_EVS\_EventType\_DEBUG, 1509  
CFE\_EVS\_EventType\_Enum\_t, 1507  
CFE\_EVS\_EventType\_ERROR, 1509  
CFE\_EVS\_EventType\_INFORMATION, 1509  
CFE\_EVS\_LogMode, 1509  
CFE\_EVS\_LogMode\_DISCARD, 1509  
CFE\_EVS\_LogMode\_Enum\_t, 1508  
CFE\_EVS\_LogMode\_OVERWRITE, 1509  
CFE\_EVS\_MsgFormat, 1509  
CFE\_EVS\_MsgFormat\_Enum\_t, 1508  
CFE\_EVS\_MsgFormat\_LONG, 1509  
CFE\_EVS\_MsgFormat\_SHORT, 1509  
default\_cfe\_evs\_fnccode\_values.h  
CFE\_EVS\_CCVAL, 1510  
CFE\_EVS\_FunctionCode\_, 1510  
CFE\_EVS\_FunctionCode\_ADD\_EVENT\_FILTER, 1510  
CFE\_EVS\_FunctionCode\_CLEAR\_LOG, 1510  
CFE\_EVS\_FunctionCode\_DELETE\_EVENT\_FILTER, 1510  
CFE\_EVS\_FunctionCode\_DISABLE\_APP\_EVENT\_TYPE, 1510  
CFE\_EVS\_FunctionCode\_DISABLE\_APP\_EVENTS, 1510  
CFE\_EVS\_FunctionCode\_DISABLE\_EVENT\_TYPE, 1510  
CFE\_EVS\_FunctionCode\_DISABLE\_PORTS, 1510  
CFE\_EVS\_FunctionCode\_ENABLE\_APP\_EVENT\_TYPE, 1510  
CFE\_EVS\_FunctionCode\_ENABLE\_APP\_EVENTS, 1510  
CFE\_EVS\_FunctionCode\_ENABLE\_EVENT\_TYPE, 1510  
CFE\_EVS\_FunctionCode\_ENABLE\_PORTS, 1510  
CFE\_EVS\_FunctionCode\_NOOP, 1510  
CFE\_EVS\_FunctionCode\_RESET\_ALL\_FILTERS, 1510  
CFE\_EVS\_FunctionCode\_RESET\_APP\_COUNTER, 1510  
CFE\_EVS\_FunctionCode\_RESET\_COUNTERS, 1510  
CFE\_EVS\_FunctionCode\_RESET\_FILTER, 1510  
CFE\_EVS\_FunctionCode\_SET\_EVENT\_FORMAT\_MODE, 1510  
CFE\_EVS\_FunctionCode\_SET\_FILTER, 1510  
CFE\_EVS\_FunctionCode\_SET\_LOG\_MODE, 1510  
CFE\_EVS\_FunctionCode\_WRITE\_APP\_DATA\_FILE, 1510  
CFE\_EVS\_FunctionCode\_WRITE\_LOG\_DATA\_FILE, 1510  
default\_cfe\_evs\_interface\_cfg\_values.h  
CFE\_MISSION\_EVS\_CFGVAL, 1511  
default\_cfe\_evs\_internal\_cfg\_values.h  
CFE\_PLATFORM\_EVS\_CFGVAL, 1511  
default\_cfe\_evs\_msgdefs.h  
CFE\_EVS\_AppDataCmd\_Payload\_t, 1514  
CFE\_EVS\_AppNameBitMaskCmd\_Payload\_t, 1514  
CFE\_EVS\_AppNameCmd\_Payload\_t, 1514  
CFE\_EVS\_AppNameEventIDCmd\_Payload\_t, 1515  
CFE\_EVS\_AppNameEventIDMaskCmd\_Payload\_t, 1515  
CFE\_EVS\_AppTlmData\_t, 1515  
CFE\_EVS\_BitMaskCmd\_Payload\_t, 1515  
CFE\_EVS\_CRITICAL\_BIT, 1514  
CFE\_EVS\_DEBUG\_BIT, 1514  
CFE\_EVS\_ERROR\_BIT, 1514  
CFE\_EVS\_HousekeepingTlm\_Payload\_t, 1515  
CFE\_EVS\_INFORMATION\_BIT, 1514  
CFE\_EVS\_LogFileCmd\_Payload\_t, 1515  
CFE\_EVS\_LongEventTlm\_Payload\_t, 1515  
CFE\_EVS\_PacketID\_t, 1515  
CFE\_EVS\_PORT1\_BIT, 1514  
CFE\_EVS\_PORT2\_BIT, 1514  
CFE\_EVS\_PORT3\_BIT, 1514  
CFE\_EVS\_PORT4\_BIT, 1514  
CFE\_EVS\_SetEventFormatMode\_Payload\_t, 1515  
CFE\_EVS\_SetLogMode\_Payload\_t, 1515  
CFE\_EVS\_ShortEventTlm\_Payload\_t, 1515  
default\_cfe\_evs\_mgid\_values.h  
CFE\_PLATFORM\_EVS\_CMD\_MIDVAL, 1516  
CFE\_PLATFORM\_EVS\_TLM\_MIDVAL, 1516  
default\_cfe\_evs\_msgids.h  
CFE\_EVS\_CMD\_MID, 1516  
CFE\_EVS\_HK\_TLM\_MID, 1517  
CFE\_EVS\_LONG\_EVENT\_MSG\_MID, 1517  
CFE\_EVS\_SEND\_HK\_MID, 1517  
CFE\_EVS\_SHORT\_EVENT\_MSG\_MID, 1517  
default\_cfe\_evs\_msgstruct.h  
CFE\_EVS\_AddEventFilterCmd\_t, 1518  
CFE\_EVS\_ClearLogCmd\_t, 1519  
CFE\_EVS\_DeleteEventFilterCmd\_t, 1519  
CFE\_EVS\_DisableAppEventsCmd\_t, 1519  
CFE\_EVS\_DisableAppEventTypeCmd\_t, 1519  
CFE\_EVS\_DisableEventCmd\_t, 1519  
CFE\_EVS\_DisablePortsCmd\_t, 1519  
CFE\_EVS\_EnableAppEventsCmd\_t, 1519  
CFE\_EVS\_EnableAppEventTypeCmd\_t, 1519  
CFE\_EVS\_EnableEventCmd\_t, 1519  
CFE\_EVS\_EnablePortsCmd\_t, 1519  
CFE\_EVS\_HousekeepingTlm\_t, 1519  
CFE\_EVS\_LongEventTlm\_t, 1519  
CFE\_EVS\_NoopCmd\_t, 1519  
CFE\_EVS\_ResetAllFiltersCmd\_t, 1519  
CFE\_EVS\_ResetAppCounterCmd\_t, 1519  
CFE\_EVS\_ResetCountersCmd\_t, 1519  
CFE\_EVS\_ResetFilterCmd\_t, 1520  
CFE\_EVS\_SendHkCmd\_t, 1520

CFE\_EVS\_SetEventFormatModeCmd\_t, 1520  
 CFE\_EVS\_SetFilterCmd\_t, 1520  
 CFE\_EVS\_SetLogModeCmd\_t, 1520  
 CFE\_EVS\_ShortEventTlm\_t, 1520  
 CFE\_EVS\_WriteAppDataFileCmd\_t, 1520  
 CFE\_EVS\_WriteLogDataFileCmd\_t, 1520  
 default\_cfe\_evs\_topicid\_values.h  
 CFE\_MISSION\_EVS\_TIDVAL, 1521  
 DEFAULT\_CFE\_FS\_FILE\_CONTENT\_ID  
 cfe\_fs\_interface\_cfg.h, 1563  
 default\_cfe\_fs\_filedef.h  
 CFE\_FS\_Header\_t, 1561  
 CFE\_FS\_SubType, 1561  
 CFE\_FS\_SubType\_Enum\_t, 1561  
 CFE\_FS\_SubType\_ES\_CDS\_REG, 1561  
 CFE\_FS\_SubType\_ES\_ERLOG, 1561  
 CFE\_FS\_SubType\_ES\_PERFDATA, 1561  
 CFE\_FS\_SubType\_ES\_QUERYALL, 1561  
 CFE\_FS\_SubType\_ES\_QUERYALLTASKS, 1562  
 CFE\_FS\_SubType\_ES\_SYSLOG, 1561  
 CFE\_FS\_SubType\_EVS\_APPDATA, 1562  
 CFE\_FS\_SubType\_EVS\_EVENTLOG, 1562  
 CFE\_FS\_SubType\_SB\_MAPDATA, 1562  
 CFE\_FS\_SubType\_SB\_PIPEDATA, 1562  
 CFE\_FS\_SubType\_SB\_ROUTEDATA, 1562  
 CFE\_FS\_SubType\_TBL\_IMG, 1561  
 CFE\_FS\_SubType\_TBL\_REG, 1561  
 DEFAULT\_CFE\_FS\_HDR\_DESC\_MAX\_LEN  
 cfe\_fs\_interface\_cfg.h, 1564  
 default\_cfe\_fs\_interface\_cfg\_values.h  
 CFE\_MISSION\_FS\_CFGVAL, 1562  
 DEFAULT\_CFE\_MISSION\_CF\_CH0\_RX\_TOPICID  
 cf\_topicids.h, 828  
 DEFAULT\_CFE\_MISSION\_CF\_CH0\_TX\_TOPICID  
 cf\_topicids.h, 828  
 DEFAULT\_CFE\_MISSION\_CF\_CH1\_RX\_TOPICID  
 cf\_topicids.h, 828  
 DEFAULT\_CFE\_MISSION\_CF\_CH1\_TX\_TOPICID  
 cf\_topicids.h, 828  
 DEFAULT\_CFE\_MISSION\_CF\_CMD\_TOPICID  
 cf\_topicids.h, 828  
 DEFAULT\_CFE\_MISSION\_CF\_EOT\_TLM\_TOPICID  
 cf\_topicids.h, 829  
 DEFAULT\_CFE\_MISSION\_CF\_HK\_TLM\_TOPICID  
 cf\_topicids.h, 829  
 DEFAULT\_CFE\_MISSION\_CF\_SEND\_HK\_TOPICID  
 cf\_topicids.h, 829  
 DEFAULT\_CFE\_MISSION\_CF\_WAKE\_UP\_TOPICID  
 cf\_topicids.h, 829  
 DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_API\_LEN  
 cfe\_core\_api\_interface\_cfg.h, 1344  
 DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_FILE\_LEN  
 cfe\_core\_api\_interface\_cfg.h, 1344  
 DEFAULT\_CFE\_MISSION\_CORE\_API\_MAX\_NUM\_FILES  
 cfe\_core\_api\_interface\_cfg.h, 1344  
 DEFAULT\_CFE\_MISSION\_ES\_APP\_TLM\_TOPICID  
 cfe\_es\_topicids.h, 1506  
 DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN  
 cfe\_es\_interface\_cfg.h, 1472  
 DEFAULT\_CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH  
 cfe\_es\_interface\_cfg.h, 1472  
 DEFAULT\_CFE\_MISSION\_ES\_CMD\_TOPICID  
 cfe\_es\_topicids.h, 1506  
 DEFAULT\_CFE\_MISSION\_ES\_CRC\_16  
 cfe\_es\_interface\_cfg.h, 1472  
 DEFAULT\_CFE\_MISSION\_ES\_CRC\_32  
 cfe\_es\_interface\_cfg.h, 1472  
 DEFAULT\_CFE\_MISSION\_ES\_CRC\_8  
 cfe\_es\_interface\_cfg.h, 1472  
 DEFAULT\_CFE\_MISSION\_ES\_DEFAULT\_CRC  
 cfe\_es\_interface\_cfg.h, 1472  
 DEFAULT\_CFE\_MISSION\_ES\_HK\_TLM\_TOPICID  
 cfe\_es\_topicids.h, 1506  
 DEFAULT\_CFE\_MISSION\_ES\_MAX\_APPLICATIONS  
 cfe\_es\_interface\_cfg.h, 1472  
 DEFAULT\_CFE\_MISSION\_ES\_MEMSTATS\_TLM\_TOPICID  
 cfe\_es\_topicids.h, 1506  
 DEFAULT\_CFE\_MISSION\_ES\_PERF\_MAX\_IDS  
 cfe\_es\_interface\_cfg.h, 1473  
 DEFAULT\_CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS  
 cfe\_es\_interface\_cfg.h, 1473  
 DEFAULT\_CFE\_MISSION\_ES\_SEND\_HK\_TOPICID  
 cfe\_es\_topicids.h, 1506  
 DEFAULT\_CFE\_MISSION\_EVS\_CMD\_TOPICID  
 cfe\_evs\_topicids.h, 1559  
 DEFAULT\_CFE\_MISSION\_EVS\_HK\_TLM\_TOPICID  
 cfe\_evs\_topicids.h, 1559  
 DEFAULT\_CFE\_MISSION\_EVS\_LONG\_EVENT\_MSG\_TOPICID  
 cfe\_evs\_topicids.h, 1559  
 DEFAULT\_CFE\_MISSION\_EVS\_MAX\_MESSAGE\_LENGTH  
 cfe\_evs\_interface\_cfg.h, 1552  
 DEFAULT\_CFE\_MISSION\_EVS\_SEND\_HK\_TOPICID  
 cfe\_evs\_topicids.h, 1560  
 DEFAULT\_CFE\_MISSION\_EVS\_SHORT\_EVENT\_MSG\_TOPICID  
 cfe\_evs\_topicids.h, 1560  
 DEFAULT\_CFE\_MISSION\_SB\_ALLSUBS\_TLM\_TOPICID  
 cfe\_sb\_topicids.h, 1624  
 DEFAULT\_CFE\_MISSION\_SB\_CMD\_TOPICID  
 cfe\_sb\_topicids.h, 1624  
 DEFAULT\_CFE\_MISSION\_SB\_HK\_TLM\_TOPICID  
 cfe\_sb\_topicids.h, 1625  
 DEFAULT\_CFE\_MISSION\_SB\_MAX\_PIPES  
 cfe\_sb\_interface\_cfg.h, 1607  
 DEFAULT\_CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE  
 cfe\_sb\_interface\_cfg.h, 1608  
 DEFAULT\_CFE\_MISSION\_SB\_ONESUB\_TLM\_TOPICID

cfe\_sb\_topicids.h, 1625  
DEFAULT\_CFE\_MISSION\_SB\_SEND\_HK\_TOPICID  
    cfe\_sb\_topicids.h, 1625  
DEFAULT\_CFE\_MISSION\_SB\_STATS\_TLM\_TOPICID  
    cfe\_sb\_topicids.h, 1625  
DEFAULT\_CFE\_MISSION\_SB\_SUB\_ENTRIES\_PER\_PKT  
    cfe\_sb\_interface\_cfg.h, 1608



cfe\_es\_internal\_cfg.h, 1503  
DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL DEFAULT\_CFE\_PLATFORM\_SB\_BUFS\_MEMORY\_BYT  
cfe\_es\_internal\_cfg.h, 1503 cfe\_sb\_internal\_cfg.h, 1618  
DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FIL  
cfe\_es\_internal\_cfg.h, 1503 cfe\_sb\_internal\_cfg.h, 1618  
DEFAULT\_CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONDEF DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIM  
cfe\_es\_internal\_cfg.h, 1503 cfe\_sb\_internal\_cfg.h, 1618  
DEFAULT\_CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FIL  
cfe\_es\_internal\_cfg.h, 1503 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STDF DEFAULT\_CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FIL  
cfe\_es\_internal\_cfg.h, 1503 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK1  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK2  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK3  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK4  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK5  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMedefault CFE\_PLATFORM\_SB\_FILTER\_MASK6  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_DSEFAULT CFE\_PLATFORM\_SB\_FILTER\_MASK7  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE DEFAULT\_CFE\_PLATFORM\_SB\_FILTER\_MASK8  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1619  
DEFAULT\_CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT1  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT2  
cfe\_es\_internal\_cfg.h, 1504 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT3  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FIL DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT4  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT5  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT6  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT MODE DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT7  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG DEFAULT\_CFE\_PLATFORM\_SB\_FILTERED\_EVENT8  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_LOG\_MAX DEFAULT\_CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE  
cfe\_evs\_internal\_cfg.h, 1557 cfe\_sb\_internal\_cfg.h, 1620  
DEFAULT\_CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT  
cfe\_evs\_internal\_cfg.h, 1558 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_EVS\_PORT\_DEFAULT DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS  
cfe\_evs\_internal\_cfg.h, 1558 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY DEFAULT\_CFE\_PLATFORM\_SB\_MAX\_PIPES  
cfe\_evs\_internal\_cfg.h, 1558 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01

cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07 cfe\_sb\_internal\_cfg.h, 1621  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16 cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_POOL\_MAX\_BUCKETS cfe\_sb\_internal\_cfg.h, 1622  
DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY cfe\_sb\_internal\_cfg.h, 1623  
DEFAULT\_CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE cfe\_sb\_internal\_cfg.h, 1623  
DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1643  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE cfe\_tbl\_internal\_cfg.h, 1683  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1 cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2 cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3 cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4 cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1 cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_2 cfe\_tbl\_internal\_cfg.h, 1684  
default\_cfe\_tbl\_platform\_cfg.h, 1644  
DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_CLIENT cfe\_time\_internal\_cfg.h, 1741  
DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY cfe\_time\_internal\_cfg.h, 1741  
DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SERVER cfe\_time\_internal\_cfg.h, 1742  
DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SIGNAL cfe\_time\_internal\_cfg.h, 1742  
DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SOURCE cfe\_time\_internal\_cfg.h, 1742  
DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS cfe\_time\_internal\_cfg.h, 1742  
DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET cfe\_time\_internal\_cfg.h, 1742

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME  
  `cfe_time_internal_cfg.h`, [1742](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_START\_FLY  
  `cfe_time_internal_cfg.h`, [1742](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT  
  `cfe_time_internal_cfg.h`, [1742](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL  
  `cfe_time_internal_cfg.h`, [1742](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS  
  `cfe_time_internal_cfg.h`, [1742](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY  
  `cfe_time_internal_cfg.h`, [1743](#)

DEFAULT\_CFE\_PLATFORM\_TLM\_MID\_BASE  
  `default_cfe_core_api_msgid_mapping.h`, [1335](#)

default\_cfe\_sb\_extern\_typedefs.h  
  `CFE_SB_MsgId_Atom_t`, [1567](#)  
  `CFE_SB_Pipeld_t`, [1567](#)  
  `CFE_SB_QosPriority`, [1568](#)  
  `CFE_SB_QosPriority_Enum_t`, [1567](#)  
  `CFE_SB_QosPriority_HIGH`, [1568](#)  
  `CFE_SB_QosPriority_LOW`, [1568](#)  
  `CFE_SB_QosReliability`, [1568](#)  
  `CFE_SB_QosReliability_Enum_t`, [1567](#)  
  `CFE_SB_QosReliability_HIGH`, [1568](#)  
  `CFE_SB_QosReliability_LOW`, [1568](#)  
  `CFE_SB_Routeld_Atom_t`, [1567](#)

default\_cfe\_sb\_fcncode\_values.h  
  `CFE_SB_CCVAL`, [1568](#)  
  `CFE_SB_FunctionCode_`, [1569](#)  
  `CFE_SB_FunctionCode_DISABLE_ROUTE`, [1569](#)  
  `CFE_SB_FunctionCode_DISABLE_SUB_REPORTING`,  
    [1569](#)  
  `CFE_SB_FunctionCode_ENABLE_ROUTE`, [1569](#)  
  `CFE_SB_FunctionCode_ENABLE_SUB_REPORTING`,  
    [1569](#)  
  `CFE_SB_FunctionCode_NOOP`, [1569](#)  
  `CFE_SB_FunctionCode_RESET_COUNTERS`, [1569](#)  
  `CFE_SB_FunctionCode_SEND_PREV_SUBS`, [1569](#)

CFE\_SB\_FunctionCode\_SEND\_SB\_STATS, [1569](#)  
CFE\_SB\_FunctionCode\_WRITE\_MAP\_INFO, [1569](#)  
CFE\_SB\_FunctionCode\_WRITE\_PIPE\_INFO, [1569](#)  
CFE\_SB\_FunctionCode\_WRITE\_ROUTING\_INFO,  
  [1569](#)

default\_cfe\_sb\_interface\_cfg\_values.h  
  `CFE_MISSION_SB_CFGVAL`, [1569](#)

default\_cfe\_sb\_internal\_cfg\_values.h  
  `CFE_PLATFORM_SB_CFGVAL`, [1570](#)

default\_cfe\_sb\_msgdefs.h  
  `CFE_SB_AllSubscriptionsTlm_Payload_t`, [1572](#)  
  `CFE_SB_HousekeepingTlm_Payload_t`, [1572](#)  
  `CFE_SB_MsgMapFileEntry_t`, [1572](#)  
  `CFE_SB_PipeDepthStats_t`, [1572](#)  
  `CFE_SB_PipeInfoEntry_t`, [1572](#)  
  `CFE_SB_RouteCmd_Payload_t`, [1572](#)  
  `CFE_SB_RoutingFileEntry_t`, [1572](#)  
  `CFE_SB_SingleSubscriptionTlm_Payload_t`, [1573](#)  
  `CFE_SB_StatsTlm_Payload_t`, [1573](#)  
  `CFE_SB_SubEntries_t`, [1573](#)  
  `CFE_SB_WriteFileInfoCmd_Payload_t`, [1573](#)  
  `default_cfe_sb_msgid_values.h`

CFE\_PLATFORM\_SB\_CMD\_MIDVAL, [1573](#)  
CFE\_PLATFORM\_SB\_TLM\_MIDVAL, [1574](#)

default\_cfe\_sb\_msgids.h  
  `CFE_SB_ALLSUBS_TLM_MID`, [1574](#)  
  `CFE_SB_CMD_MID`, [1574](#)  
  `CFE_SB_HK_TLM_MID`, [1574](#)  
  `CFE_SB_ONESUB_TLM_MID`, [1574](#)  
  `CFE_SB_SEND_HK_MID`, [1574](#)  
  `CFE_SB_STATS_TLM_MID`, [1575](#)  
  `CFE_SB_SUB_RPT_CTRL_MID`, [1575](#)

default\_cfe\_sb\_msgstruct.h  
  `CFE_SB_AllSubscriptionsTlm_t`, [1576](#)  
  `CFE_SB_DisableRouteCmd_t`, [1576](#)  
  `CFE_SB_DisableSubReportingCmd_t`, [1576](#)  
  `CFE_SB_EnableRouteCmd_t`, [1576](#)  
  `CFE_SB_EnableSubReportingCmd_t`, [1576](#)  
  `CFE_SB_HousekeepingTlm_t`, [1576](#)  
  `CFE_SB_NoopCmd_t`, [1576](#)  
  `CFE_SB_ResetCountersCmd_t`, [1576](#)  
  `CFE_SB_SendHkCmd_t`, [1576](#)  
  `CFE_SB_SendPrevSubsCmd_t`, [1576](#)  
  `CFE_SB_SendSbStatsCmd_t`, [1576](#)  
  `CFE_SB_SingleSubscriptionTlm_t`, [1576](#)  
  `CFE_SB_StatsTlm_t`, [1576](#)  
  `CFE_SB_WriteMapInfoCmd_t`, [1577](#)  
  `CFE_SB_WritePipeInfoCmd_t`, [1577](#)  
  `CFE_SB_WriteRoutingInfoCmd_t`, [1577](#)  
  `default_cfe_sb_topcid_values.h`  
    `CFE_MISSION_SB_TIDVAL`, [1577](#)

default\_cfe\_tbl\_extern\_typedefs.h  
  `CFE_TBL_BufferSelect`, [1627](#)  
  `CFE_TBL_BufferSelect_ACTIVE`, [1627](#)

CFE\_TBL\_BufferSelect\_Enum\_t, 1626  
 CFE\_TBL\_BufferSelect\_INACTIVE, 1627  
 CFE\_TBL\_CombinedFileHdr\_t, 1626  
 CFE\_TBL\_File\_Hdr\_t, 1626  
 CFE\_TBL\_HandleId\_t, 1626  
 CFE\_TBL\_RegId\_t, 1626  
 default\_cfe\_tbl\_fcncode\_values.h  
   CFE\_TBL\_CCVAL, 1627  
   CFE\_TBL\_FunctionCode\_, 1627  
   CFE\_TBL\_FunctionCode\_ABORT\_LOAD, 1628  
   CFE\_TBL\_FunctionCode\_ACTIVATE, 1628  
   CFE\_TBL\_FunctionCode\_DELETE\_CDS, 1628  
   CFE\_TBL\_FunctionCode\_DUMP, 1628  
   CFE\_TBL\_FunctionCode\_DUMP\_REGISTRY, 1628  
   CFE\_TBL\_FunctionCode\_LOAD, 1628  
   CFE\_TBL\_FunctionCode\_NOOP, 1627  
   CFE\_TBL\_FunctionCode\_RESET\_COUNTERS, 1628  
   CFE\_TBL\_FunctionCode\_SEND\_REGISTRY, 1628  
   CFE\_TBL\_FunctionCode\_VALIDATE, 1628  
 default\_cfe\_tbl\_interface\_cfg\_values.h  
   CFE\_MISSION\_TBL\_CFGVAL, 1628  
 default\_cfe\_tbl\_internal\_cfg\_values.h  
   CFE\_PLATFORM\_TBL\_CFGVAL, 1629  
 default\_cfe\_tbl\_mission\_cfg.h  
   CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN, 1629  
   CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, 1629  
   DEFAULT\_CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN, 1630  
   DEFAULT\_CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, 1630  
 default\_cfe\_tbl\_msgdefs.h  
   CFE\_TBL\_AbortLoadCmd\_Payload\_t, 1632  
   CFE\_TBL\_ActivateCmd\_Payload\_t, 1632  
   CFE\_TBL\_DelCDSCmd\_Payload\_t, 1632  
   CFE\_TBL\_DumpCmd\_Payload\_t, 1632  
   CFE\_TBL\_DumpRegistryCmd\_Payload\_t, 1632  
   CFE\_TBL\_HousekeepingTlm\_Payload\_t, 1632  
   CFE\_TBL\_LoadCmd\_Payload\_t, 1632  
   CFE\_TBL\_NotifyCmd\_Payload\_t, 1632  
   CFE\_TBL\_SendRegistryCmd\_Payload\_t, 1632  
   CFE\_TBL\_TblRegPacket\_Payload\_t, 1632  
   CFE\_TBL\_ValidateCmd\_Payload\_t, 1633  
 default\_cfe\_tblmsgid\_values.h  
   CFE\_PLATFORM\_TBL\_CMD\_MIDVAL, 1633  
   CFE\_PLATFORM\_TBL\_TLM\_MIDVAL, 1633  
 default\_cfe\_tbl\_msgids.h  
   CFE\_TBL\_CMD\_MID, 1634  
   CFE\_TBL\_HK\_TLM\_MID, 1634  
   CFE\_TBL\_REG\_TLM\_MID, 1634  
   CFE\_TBL\_SEND\_HK\_MID, 1634  
 default\_cfe\_tbl\_msgstruct.h  
   CFE\_TBL\_AbortLoadCmd\_t, 1635  
   CFE\_TBL\_ActivateCmd\_t, 1635  
   CFE\_TBL\_DeleteCDSCmd\_t, 1635  
   CFE\_TBL\_DumpCmd\_t, 1635  
   CFE\_TBL\_DumpRegistryCmd\_t, 1635  
   CFE\_TBL\_HousekeepingTlm\_t, 1635  
   CFE\_TBL\_LoadCmd\_t, 1636  
   CFE\_TBL\_NoopCmd\_t, 1636  
   CFE\_TBL\_NotifyCmd\_t, 1636  
   CFE\_TBL\_ResetCountersCmd\_t, 1636  
   CFE\_TBL\_SendHkCmd\_t, 1636  
   CFE\_TBL\_SendRegistryCmd\_t, 1636  
   CFE\_TBL\_TableRegistryTlm\_t, 1636  
   CFE\_TBL\_ValidateCmd\_t, 1636  
 default\_cfe\_tbl\_platform\_cfg.h  
   CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES, 1638  
   CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE, 1638  
   CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES, 1638  
   CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE, 1638  
   CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES, 1639  
   CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES, 1639  
   CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS, 1639  
   CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS LOADS, 1640  
   CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE, 1640  
   CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY, 1640  
   CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE, 1641  
   CFE\_PLATFORM\_TBL\_U32FROM4CHARS, 1641  
   CFE\_PLATFORM\_TBL\_VALID\_PRID\_1, 1641  
   CFE\_PLATFORM\_TBL\_VALID\_PRID\_2, 1641  
   CFE\_PLATFORM\_TBL\_VALID\_PRID\_3, 1642  
   CFE\_PLATFORM\_TBL\_VALID\_PRID\_4, 1642  
   CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT, 1642  
   CFE\_PLATFORM\_TBL\_VALID\_SCID\_1, 1642  
   CFE\_PLATFORM\_TBL\_VALID\_SCID\_2, 1642  
   CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT, 1642  
   DEFAULT\_CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES, 1643  
   DEFAULT\_CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE, 1643  
   DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES, 1643  
   DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE, 1643  
   DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES, 1643

DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES, CFE\_TIME\_FlywheelState\_Enum\_t, 1688  
1643 CFE\_TIME\_FlywheelState\_IS\_FLY, 1690  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIDATIONS, CFE\_TIME\_FlywheelState\_NO\_FLY, 1690  
1643 CFE\_TIME\_SetState, 1690  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS, CFE\_TIME\_SetState\_Enum\_t, 1688  
1643 CFE\_TIME\_SetState\_NOT\_SET, 1690  
DEFAULT\_CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE, CFE\_TIME\_SetState\_WAS\_SET, 1690  
1644 CFE\_TIME\_SourceSelect, 1690  
DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY, CFE\_TIME\_SourceSelect\_Enum\_t, 1688  
1644 CFE\_TIME\_SourceSelect\_EXTERNAL, 1690  
DEFAULT\_CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE, CFE\_TIME\_SourceSelect\_INTERNAL, 1690  
1644 CFE\_TIME\_SysTime\_t, 1688  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_1, CFE\_TIME\_ToneSignalSelect, 1691  
1644 CFE\_TIME\_ToneSignalSelect\_Enum\_t, 1689  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_2, CFE\_TIME\_ToneSignalSelect\_PRIMARY, 1691  
1644 CFE\_TIME\_ToneSignalSelect\_REDUNDANT, 1691  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_3, default\_cfe\_time\_fcncode\_values.h  
1644 CFE\_TIME\_CCVAL, 1691  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_4, CFE\_TIME\_FunctionCode\_, 1691  
1644 CFE\_TIME\_FunctionCode\_ADD\_ADJUST, 1692  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT, CFE\_TIME\_FunctionCode\_ADD\_DELAY, 1692  
1644 CFE\_TIME\_FunctionCode\_ADD\_ONE\_HZ\_ADJUSTMENT,  
1644 1692  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_1, CFE\_TIME\_FunctionCode\_NOOP, 1692  
1644 CFE\_TIME\_FunctionCode\_RESET\_COUNTERS,  
1644 1692  
DEFAULT\_CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT, CFE\_TIME\_FunctionCode\_SEND\_DIAGNOSTIC,  
1645 1692  
default\_cfe\_tbl\_topicid\_values.h CFE\_TIME\_FunctionCode\_SET\_LEAP\_SECONDS,  
CFE\_MISSION\_TBL\_TIDVAL, 1645 1692  
default\_cfe\_time\_extern\_typedefs.h CFE\_TIME\_FunctionCode\_SET\_MET, 1692  
CFE\_TIME\_AdjustDirection, 1689 CFE\_TIME\_FunctionCode\_SET\_SIGNAL, 1692  
CFE\_TIME\_AdjustDirection\_ADD, 1689 CFE\_TIME\_FunctionCode\_SET\_SOURCE, 1692  
CFE\_TIME\_AdjustDirection\_Enum\_t, 1687 CFE\_TIME\_FunctionCode\_SET\_STATE, 1692  
CFE\_TIME\_AdjustDirection\_SUBTRACT, 1689 CFE\_TIME\_FunctionCode\_SET\_STCF, 1692  
CFE\_TIME\_ClockState, 1689 CFE\_TIME\_FunctionCode\_SET\_TIME, 1692  
CFE\_TIME\_ClockState\_Enum\_t, 1687 CFE\_TIME\_FunctionCode\_SUB\_ADJUST, 1692  
CFE\_TIME\_ClockState\_FLYWHEEL, 1689 CFE\_TIME\_FunctionCode\_SUB\_DELAY, 1692  
CFE\_TIME\_ClockState\_INVALID, 1689 CFE\_TIME\_FunctionCode\_SUB\_ONE\_HZ\_ADJUSTMENT,  
CFE\_TIME\_ClockState\_VALID, 1689 1692  
CFE\_TIME\_FlagBit, 1689 default\_cfe\_time\_interface\_cfg\_values.h  
CFE\_TIME\_FlagBit\_ADD1HZ, 1690 CFE\_MISSION\_TIME\_CFGVAL, 1692  
CFE\_TIME\_FlagBit\_ADDADJ, 1690 default\_cfe\_time\_internal\_cfg\_values.h  
CFE\_TIME\_FlagBit\_ADDTCL, 1690 CFE\_PLATFORM\_TIME\_CFGVAL, 1693  
CFE\_TIME\_FlagBit\_CLKSET, 1690 default\_cfe\_time\_msgdefs.h  
CFE\_TIME\_FlagBit\_CMDFLY, 1690 CFE\_TIME\_DiagnosticTlm\_Payload\_t, 1695  
CFE\_TIME\_FlagBit\_Enum\_t, 1688 CFE\_TIME\_HousekeepingTlm\_Payload\_t, 1695  
CFE\_TIME\_FlagBit\_FLYING, 1690 CFE\_TIME\_LeapsCmd\_Payload\_t, 1695  
CFE\_TIME\_FlagBit\_GDTONE, 1690 CFE\_TIME\_OneHzAdjustmentCmd\_Payload\_t, 1695  
CFE\_TIME\_FlagBit\_SERVER, 1690 CFE\_TIME\_SignalCmd\_Payload\_t, 1696  
CFE\_TIME\_FlagBit\_SIGPRI, 1690 CFE\_TIME\_SourceCmd\_Payload\_t, 1696  
CFE\_TIME\_FlagBit\_SRCINT, 1690 CFE\_TIME\_StateCmd\_Payload\_t, 1696  
CFE\_TIME\_FlagBit\_SRVFLY, 1690 CFE\_TIME\_TimeCmd\_Payload\_t, 1696  
CFE\_TIME\_FlywheelState, 1690 CFE\_TIME\_ToneDataCmd\_Payload\_t, 1696

default\_cfe\_time\_msgid\_values.h  
   CFE\_PLATFORM\_TIME\_CMD\_MIDVAL, 1696  
   CFE\_PLATFORM\_TIME\_GLBCMD\_MIDVAL, 1696  
   CFE\_PLATFORM\_TIME\_TLM\_MIDVAL, 1697  
 default\_cfe\_time\_msgids.h  
   CFE\_TIME\_1HZ\_CMD\_MID, 1697  
   CFE\_TIME\_CMD\_MID, 1697  
   CFE\_TIME\_DATA\_CMD\_MID, 1697  
   CFE\_TIME\_DIAG\_TLM\_MID, 1697  
   CFE\_TIME\_HK\_TLM\_MID, 1697  
   CFE\_TIME\_ONEHZ\_CMD\_MID, 1698  
   CFE\_TIME\_SEND\_CMD\_MID, 1698  
   CFE\_TIME\_SEND\_HK\_MID, 1698  
   CFE\_TIME\_TONE\_CMD\_MID, 1698  
 default\_cfe\_time\_msgstruct.h  
   CFE\_TIME\_AddAdjustCmd\_t, 1699  
   CFE\_TIME\_AddDelayCmd\_t, 1699  
   CFE\_TIME\_AddOneHzAdjustmentCmd\_t, 1700  
   CFE\_TIME\_DiagnosticTlm\_t, 1700  
   CFE\_TIME\_FakeToneCmd\_t, 1700  
   CFE\_TIME\_HousekeepingTlm\_t, 1700  
   CFE\_TIME\_NoopCmd\_t, 1700  
   CFE\_TIME\_OneHzCmd\_t, 1700  
   CFE\_TIME\_ResetCountersCmd\_t, 1700  
   CFE\_TIME\_SendDiagnosticCmd\_t, 1700  
   CFE\_TIME\_SendHkCmd\_t, 1700  
   CFE\_TIME\_SetLeapSecondsCmd\_t, 1700  
   CFE\_TIME\_SetMETCCmd\_t, 1700  
   CFE\_TIME\_SetSignalCmd\_t, 1700  
   CFE\_TIME\_SetSourceCmd\_t, 1700  
   CFE\_TIME\_SetStateCmd\_t, 1700  
   CFE\_TIME\_SetSTCFCCmd\_t, 1700  
   CFE\_TIME\_SetTimeCmd\_t, 1701  
   CFE\_TIME\_SubAdjustCmd\_t, 1701  
   CFE\_TIME\_SubDelayCmd\_t, 1701  
   CFE\_TIME\_SubOneHzAdjustmentCmd\_t, 1701  
   CFE\_TIME\_ToneDataCmd\_t, 1701  
   CFE\_TIME\_ToneSignalCmd\_t, 1701  
 default\_cfe\_time\_topicid\_values.h  
   CFE\_MISSION\_TIME\_TIDVAL, 1701  
 DEFAULT\_GLOBAL\_CMD\_MID\_BASE  
   default\_cfe\_core\_api\_msgid\_mapping.h, 1335  
 DEFAULT\_GLOBAL\_TLM\_MID\_BASE  
   default\_cfe\_core\_api\_msgid\_mapping.h, 1335  
 DelayDirection  
   CFE\_TIME\_DiagnosticTlm\_Payload, 762  
 delivery\_code  
   CF\_Logical\_PduFin, 588  
 dequeue\_enabled  
   CF\_ChannelConfig, 544  
 Description  
   CFE\_FS\_FileWriteMetaData, 702  
   CFE\_FS\_Header, 704  
   CFE\_TBL\_FileDef, 738  
 dest\_eid  
   CF\_PollDir, 603  
 dest\_filename  
   CF\_Logical\_PduMd, 593  
 dest\_id  
   CF\_Playback, 601  
   CF\_TxFile\_Payload, 623  
 destination\_eid  
   CF\_Logical\_PduHeader, 590  
 dir  
   CF\_History, 569  
 dir\_id  
   CF\_Playback, 601  
 direction  
   CF\_EotPacket\_Payload, 560  
   CF\_Logical\_PduHeader, 590  
 directive\_and\_subtype\_code  
   CF\_CFDP\_PduAck, 528  
 directive\_code  
   CF\_CFDP\_PduFileDirectiveHeader, 530  
   CF\_Logical\_PduFileDirectiveHeader, 588  
 directory\_read  
   CF\_HkFault, 574  
 diropen  
   CF\_Playback, 601  
 DoubleBuffered  
   CFE\_TBL\_Info, 744  
   CFE\_TBL\_TblRegPacket\_Payload, 752  
 dropped  
   CF\_HkRecv, 578  
 dst\_dir  
   CF\_PollDir, 604  
 dst\_filename  
   CF\_TxFile\_Payload, 623  
   CF\_TxnFilenames, 624  
 DumpFilename  
   CFE\_ES\_DumpCDSRegistryCmd\_Payload, 640  
   CFE\_TBL\_DumpCmd\_Payload, 735  
   CFE\_TBL\_DumpRegistryCmd\_Payload, 736  
 DumpOnly  
   CFE\_TBL\_Info, 744  
   CFE\_TBL\_TblRegPacket\_Payload, 752  
 DuplicateSubscriptionsCounter  
   CFE\_SB\_HousekeepingTlm\_Payload, 710  
 dword  
   CF\_UnionArgs\_Payload, 625  
 eds\_of\_extern\_typedefs.h  
   CF\_CFDP\_Class\_t, 814  
   CF\_EntityId\_t, 814  
   CF\_FileName\_t, 814  
   CF\_GetSet\_ValueID\_MAX, 814  
   CF\_GetSet\_ValueID\_t, 814  
   CF\_PathName\_t, 814

CF\_QueueIdx\_NUM, 814  
CF\_QueueIdx\_t, 814  
CF\_TransactionSeq\_t, 814  
eds\_cf\_fcncode\_values.h  
    CF\_CCVAL, 815  
eds\_cf\_interface\_cfg\_values.h  
    CF\_INTERFACE\_CFGVAL, 815  
eds\_cf\_topicid\_values.h  
    CFE\_MISSION\_CF\_TIDVAL, 816  
eid  
    CF\_Logical\_TlvData, 596  
    CF\_Transaction\_Payload, 617  
eid\_length  
    CF\_Logical\_PduHeader, 590  
eid\_tsn\_lengths  
    CF\_CFDP\_PduHeader, 531  
ElementPtr  
    CFE\_Config\_ArrayValue, 628  
enabled  
    CF\_Engine, 558  
    CF\_PollDir, 604  
encode  
    CF\_Output, 598  
engine  
    CF\_AppData\_t, 525  
Entries  
    CFE\_SB\_AllSubscriptionsTlm\_Payload, 705  
Entry  
    CFE\_SB\_AllSubscriptionsTlm\_Payload, 705  
entry\_point  
    OS\_module\_prop\_t, 790  
EntryPoint  
    CFE\_ES\_AppInfo, 632  
eof  
    CF\_Logical\_IntHeader, 581  
eof\_ack\_count  
    CF\_Flags\_Rx, 563  
eof\_ack\_recv  
    CF\_Flags\_Tx, 565  
eof\_count  
    CF\_Flags\_Rx, 563  
eof\_crc  
    CF\_StateData, 609  
eof\_size  
    CF\_StateData, 609  
ERLogEntries  
    CFE\_ES\_HousekeepingTlm\_Payload, 645  
ERLogIndex  
    CFE\_ES\_HousekeepingTlm\_Payload, 645  
err  
    CF\_HkCmdCounters, 572  
error  
    CF\_HkRecv, 578  
    CF\_Traverse\_WriteHistoryFileArg, 620  
    CF\_Traverse\_WriteTxnFileArg, 621  
EventID  
    CFE\_EVS\_AppNameEventIDCmd\_Payload, 677  
    CFE\_EVS\_AppNameEventIDMaskCmd\_Payload, 678  
    CFE\_EVS\_BinFilter, 680  
    CFE\_EVS\_PacketID, 693  
EventType  
    CFE\_EVS\_PacketID, 693  
example\_2x32bit\_cfe\_es\_memaddress.h  
    CFE\_ES\_MEMADDRESS\_C, 1270  
    CFE\_ES\_MemAddress\_FromNative, 1271  
    CFE\_ES\_MEMADDRESS\_TO\_PTR, 1270  
    CFE\_ES\_MemAddress\_ToNative, 1271  
    CFE\_ES\_MEMOFFSET\_C, 1270  
    CFE\_ES\_MemOffset\_FromNative, 1271  
    CFE\_ES\_MEMOFFSET\_TO\_SIZE, 1270  
    CFE\_ES\_MemOffset\_ToNative, 1271  
example\_32bit\_cfe\_es\_memaddress.h  
    CFE\_ES\_MEMADDRESS\_C, 1272  
    CFE\_ES\_MemAddress\_t, 1272  
    CFE\_ES\_MEMADDRESS\_TO\_PTR, 1272  
    CFE\_ES\_MEMOFFSET\_C, 1272  
    CFE\_ES\_MemOffset\_t, 1273  
    CFE\_ES\_MEMOFFSET\_TO\_SIZE, 1272  
example\_64bit\_cfe\_es\_memaddress.h  
    CFE\_ES\_MEMADDRESS\_C, 1274  
    CFE\_ES\_MemAddress\_t, 1274  
    CFE\_ES\_MEMADDRESS\_TO\_PTR, 1274  
    CFE\_ES\_MEMOFFSET\_C, 1274  
    CFE\_ES\_MemOffset\_t, 1274  
    CFE\_ES\_MEMOFFSET\_TO\_SIZE, 1274  
example\_mission\_cfg.h  
    CFE\_FS\_FILE\_CONTENT\_ID, 1276  
    CFE\_FS\_HDR\_DESC\_MAX\_LEN, 1276  
    CFE\_MISSION\_ES\_CDS\_MAX\_FULL\_NAME\_LEN, 1276  
    CFE\_MISSION\_ES\_CDS\_MAX\_NAME\_LENGTH, 1276  
    CFE\_MISSION\_ES\_CRC\_16, 1276  
    CFE\_MISSION\_ES\_CRC\_32, 1276  
    CFE\_MISSION\_ES\_CRC\_8, 1277  
    CFE\_MISSION\_ES\_DEFAULT\_CRC, 1277  
    CFE\_MISSION\_ES\_MAX\_APPLICATIONS, 1277  
    CFE\_MISSION\_ES\_PERF\_MAX\_IDS, 1277  
    CFE\_MISSION\_ES\_POOL\_MAX\_BUCKETS, 1277  
    CFE\_MISSION\_ES\_MAX\_MESSAGE\_LENGTH, 1278  
    CFE\_MISSION\_MAX\_API\_LEN, 1278  
    CFE\_MISSION\_MAX\_FILE\_LEN, 1278  
    CFE\_MISSION\_MAX\_NUM\_FILES, 1279  
    CFE\_MISSION\_MAX\_PATH\_LEN, 1279  
    CFE\_MISSION\_SB\_MAX\_PIPES, 1280  
    CFE\_MISSION\_SB\_MAX\_SB\_MSG\_SIZE, 1280

CFE\_MISSION\_TBL\_MAX\_FULL\_NAME\_LEN,  
1280  
CFE\_MISSION\_TBL\_MAX\_NAME\_LENGTH, 1281  
CFE\_MISSION\_TIME\_AT\_TONE\_WAS, 1281  
CFE\_MISSION\_TIME\_AT\_TONE\_WILL\_BE, 1282  
CFE\_MISSION\_TIME\_CFG\_DEFAULT\_TAI, 1282  
CFE\_MISSION\_TIME\_CFG\_DEFAULT\_UTC, 1282  
CFE\_MISSION\_TIME\_CFG\_FAKE\_TONE, 1282  
CFE\_MISSION\_TIME\_DEF\_DELAY\_SECS, 1282  
CFE\_MISSION\_TIME\_DEF\_DELAY\_SUBS, 1282  
CFE\_MISSION\_TIME\_DEF\_LEAPS, 1283  
CFE\_MISSION\_TIME\_DEF\_MET\_SECS, 1283  
CFE\_MISSION\_TIME\_DEF\_MET\_SUBS, 1283  
CFE\_MISSION\_TIME\_DEF\_STCF\_SECS, 1283  
CFE\_MISSION\_TIME\_DEF\_STCF\_SUBS, 1283  
CFE\_MISSION\_TIME\_EPOCH\_DAY, 1283  
CFE\_MISSION\_TIME\_EPOCH\_HOUR, 1283  
CFE\_MISSION\_TIME\_EPOCH\_MICROS, 1283  
CFE\_MISSION\_TIME\_EPOCH\_MINUTE, 1284  
CFE\_MISSION\_TIME\_EPOCH\_SECOND, 1284  
CFE\_MISSION\_TIME\_EPOCH\_YEAR, 1284  
CFE\_MISSION\_TIME\_FS\_FACTOR, 1284  
CFE\_MISSION\_TIME\_MAX\_ELAPSED, 1284  
CFE\_MISSION\_TIME\_MIN\_ELAPSED, 1284  
example\_platform\_cfg.h  
    CFE\_PLATFORM\_CORE\_MAX\_STARTUP\_MSEC,  
        1289  
    CFE\_PLATFORM\_ENDIAN, 1289  
    CFE\_PLATFORM\_ES\_APP\_KILL\_TIMEOUT, 1289  
    CFE\_PLATFORM\_ES\_APP\_SCAN\_RATE, 1290  
    CFE\_PLATFORM\_ES\_CDS\_MAX\_BLOCK\_SIZE,  
        1290  
    CFE\_PLATFORM\_ES\_CDS\_MAX\_NUM\_ENTRIES,  
        1290  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_01,  
        1291  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_02,  
        1291  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_03,  
        1291  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_04,  
        1291  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_05,  
        1291  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_06,  
        1291  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_07,  
        1292  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_08,  
        1292  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_09,  
        1292  
    CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_10,  
        1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_11,  
1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_12,  
1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_13,  
1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_14,  
1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_15,  
1292  
CFE\_PLATFORM\_ES\_CDS\_MEM\_BLOCK\_SIZE\_16,  
1292  
CFE\_PLATFORM\_ES\_CDS\_SIZE, 1293  
CFE\_PLATFORM\_ES\_DEFAULT\_APP\_LOG\_FILE,  
1293  
CFE\_PLATFORM\_ES\_DEFAULT\_CDS\_REG\_DUMP\_FILE,  
1293  
CFE\_PLATFORM\_ES\_DEFAULT\_ER\_LOG\_FILE,  
1293  
CFE\_PLATFORM\_ES\_DEFAULT\_PERF\_DUMP\_FILENAME,  
1294  
CFE\_PLATFORM\_ES\_DEFAULT\_POR\_SYSLOG\_MODE,  
1294  
CFE\_PLATFORM\_ES\_DEFAULT\_PR\_SYSLOG\_MODE,  
1294  
CFE\_PLATFORM\_ES\_DEFAULT\_STACK\_SIZE,  
1295  
CFE\_PLATFORM\_ES\_DEFAULT\_SYSLOG\_FILE,  
1295  
CFE\_PLATFORM\_ES\_DEFAULT\_TASK\_LOG\_FILE,  
1295  
CFE\_PLATFORM\_ES\_ER\_LOG\_ENTRIES, 1296  
CFE\_PLATFORM\_ES\_ER\_LOG\_MAX\_CONTEXT\_SIZE,  
1296  
CFE\_PLATFORM\_ES\_MAX\_APPLICATIONS, 1296  
CFE\_PLATFORM\_ES\_MAX\_BLOCK\_SIZE, 1297  
CFE\_PLATFORM\_ES\_MAX\_GEN\_COUNTERS,  
1297  
CFE\_PLATFORM\_ES\_MAX\_LIBRARIES, 1297  
CFE\_PLATFORM\_ES\_MAX\_MEMORY\_POOLS,  
1297  
CFE\_PLATFORM\_ES\_MAX\_PROCESSOR\_RESETS,  
1298  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_01,  
1298  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_02,  
1298  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_03,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_04,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_05,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_06,

1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_07,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_08,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_09,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_10,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_11,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_12,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_13,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_14,  
1299  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_15,  
1300  
CFE\_PLATFORM\_ES\_MEM\_BLOCK\_SIZE\_16,  
1300  
CFE\_PLATFORM\_ES\_MEMPOOL\_ALIGN\_SIZE\_MIN,  
1300  
CFE\_PLATFORM\_ES\_NONVOL\_DISK\_MOUNT\_STRING,  
1300  
CFE\_PLATFORM\_ES\_NONVOL\_STARTUP\_FILE,  
1300  
CFE\_PLATFORM\_ES\_OBJECT\_TABLE\_SIZE, 1301  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_MS\_DELAY,  
1301  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_PRIORITY,  
1301  
CFE\_PLATFORM\_ES\_PERF\_CHILD\_STACK\_SIZE,  
1301  
CFE\_PLATFORM\_ES\_PERF\_DATA\_BUFFER\_SIZE,  
1302  
CFE\_PLATFORM\_ES\_PERF\_ENTRIES\_BTWN\_DLYS,  
1302  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_ALL, 1302  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_INIT,  
1302  
CFE\_PLATFORM\_ES\_PERF\_FILTMASK\_NONE,  
1303  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_ALL,  
1303  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_INIT,  
1303  
CFE\_PLATFORM\_ES\_PERF\_TRIGMASK\_NONE,  
1303  
CFE\_PLATFORM\_ES\_POOL\_MAX\_BUCKETS,  
1303  
CFE\_PLATFORM\_ES\_RAM\_DISK\_MOUNT\_STRING,  
1304  
CFE\_PLATFORM\_ES\_RAM\_DISK\_NUM\_SECTORS,  
1304  
CFE\_PLATFORM\_ES\_RAM\_DISK\_PERCENT\_RESERVED,  
1304  
CFE\_PLATFORM\_ES\_RAM\_DISK\_SECTOR\_SIZE,  
1305  
CFE\_PLATFORM\_ES\_START\_TASK\_PRIORITY,  
1305  
CFE\_PLATFORM\_ES\_START\_TASK\_STACK\_SIZE,  
1305  
CFE\_PLATFORM\_ES\_STARTUP\_SCRIPT\_TIMEOUT\_MSEC,  
1306  
CFE\_PLATFORM\_ES\_STARTUP\_SYNC\_POLL\_MSEC,  
1306  
CFE\_PLATFORM\_ES\_SYSTEM\_LOG\_SIZE, 1306  
CFE\_PLATFORM\_ES\_USER\_RESERVED\_SIZE,  
1307  
CFE\_PLATFORM\_ES\_VOLATILE\_STARTUP\_FILE,  
1307  
CFE\_PLATFORM\_EVS\_APP\_EVENTS\_PER\_SEC,  
1307  
CFE\_PLATFORM\_EVS\_DEFAULT\_APP\_DATA\_FILE,  
1308  
CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_FILE, 1308  
CFE\_PLATFORM\_EVS\_DEFAULT\_LOG\_MODE,  
1308  
CFE\_PLATFORM\_EVS\_DEFAULT\_MSG\_FORMAT\_MODE,  
1309  
CFE\_PLATFORM\_EVS\_DEFAULT\_TYPE\_FLAG,  
1309  
CFE\_PLATFORM\_EVS\_LOG\_MAX, 1309  
CFE\_PLATFORM\_EVS\_MAX\_APP\_EVENT\_BURST,  
1310  
CFE\_PLATFORM\_EVS\_MAX\_EVENT\_FILTERS,  
1310  
CFE\_PLATFORM\_EVS\_PORT\_DEFAULT, 1310  
CFE\_PLATFORM\_EVS\_START\_TASK\_PRIORITY,  
1310  
CFE\_PLATFORM\_EVS\_START\_TASK\_STACK\_SIZE,  
1311  
CFE\_PLATFORM\_SB\_BUF\_MEMORY\_BYTES,  
1311  
CFE\_PLATFORM\_SB\_DEFAULT\_MAP\_FILENAME,  
1311  
CFE\_PLATFORM\_SB\_DEFAULT\_MSG\_LIMIT, 1312  
CFE\_PLATFORM\_SB\_DEFAULT\_PIPE\_FILENAME,  
1312  
CFE\_PLATFORM\_SB\_DEFAULT\_ROUTING\_FILENAME,  
1312  
CFE\_PLATFORM\_SB\_FILTER\_MASK1, 1313  
CFE\_PLATFORM\_SB\_FILTER\_MASK2, 1313  
CFE\_PLATFORM\_SB\_FILTER\_MASK3, 1313  
CFE\_PLATFORM\_SB\_FILTER\_MASK4, 1313  
CFE\_PLATFORM\_SB\_FILTER\_MASK5, 1313  
CFE\_PLATFORM\_SB\_FILTER\_MASK6, 1313

CFE\_PLATFORM\_SB\_FILTER\_MASK7, 1313  
CFE\_PLATFORM\_SB\_FILTER\_MASK8, 1313  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT1, 1313  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT2, 1314  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT3, 1314  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT4, 1314  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT5, 1314  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT6, 1314  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT7, 1314  
CFE\_PLATFORM\_SB\_FILTERED\_EVENT8, 1314  
CFE\_PLATFORM\_SB\_HIGHEST\_VALID\_MSGID, 1314  
CFE\_PLATFORM\_SB\_MAX\_BLOCK\_SIZE, 1315  
CFE\_PLATFORM\_SB\_MAX\_DEST\_PER\_PKT, 1315  
CFE\_PLATFORM\_SB\_MAX\_MSG\_IDS, 1315  
CFE\_PLATFORM\_SB\_MAX\_PIPES, 1315  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_01, 1316  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_02, 1316  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_03, 1316  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_04, 1316  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_05, 1316  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_06, 1316  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_07, 1316  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_08, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_09, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_10, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_11, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_12, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_13, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_14, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_15, 1317  
CFE\_PLATFORM\_SB\_MEM\_BLOCK\_SIZE\_16, 1317  
CFE\_PLATFORM\_SB\_START\_TASK\_PRIORITY, 1317  
CFE\_PLATFORM\_SB\_START\_TASK\_STACK\_SIZE, 1318  
CFE\_PLATFORM\_TBL\_BUF\_MEMORY\_BYTES, 1318  
CFE\_PLATFORM\_TBL\_DEFAULT\_REG\_DUMP\_FILE, 1318  
CFE\_PLATFORM\_TBL\_MAX\_CRITICAL\_TABLES, 1319  
CFE\_PLATFORM\_TBL\_MAX\_DBL\_TABLE\_SIZE, 1319  
CFE\_PLATFORM\_TBL\_MAX\_NUM\_HANDLES, 1319  
CFE\_PLATFORM\_TBL\_MAX\_NUM\_TABLES, 1320  
CFE\_PLATFORM\_TBL\_MAX\_NUM\_VALIFICATIONS, 1320  
CFE\_PLATFORM\_TBL\_MAX\_SIMULTANEOUS\_LOADS, 1320  
CFE\_PLATFORM\_TBL\_MAX\_SNGL\_TABLE\_SIZE, 1321  
CFE\_PLATFORM\_TBL\_START\_TASK\_PRIORITY, 1321  
CFE\_PLATFORM\_TBL\_START\_TASK\_STACK\_SIZE, 1321  
CFE\_PLATFORM\_TBL\_U32FROM4CHARS, 1322  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_1, 1322  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_2, 1322  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_3, 1322  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_4, 1322  
CFE\_PLATFORM\_TBL\_VALID\_PRID\_COUNT, 1322  
CFE\_PLATFORM\_TBL\_VALID\_SCID\_1, 1323  
CFE\_PLATFORM\_TBL\_VALID\_SCID\_2, 1323  
CFE\_PLATFORM\_TBL\_VALID\_SCID\_COUNT, 1323  
CFE\_PLATFORM\_TIME\_CFG\_CLIENT, 1323  
CFE\_PLATFORM\_TIME\_CFG\_LATCH\_FLY, 1324  
CFE\_PLATFORM\_TIME\_CFG\_SERVER, 1324  
CFE\_PLATFORM\_TIME\_CFG\_SIGNAL, 1324  
CFE\_PLATFORM\_TIME\_CFG\_SOURCE, 1324  
CFE\_PLATFORM\_TIME\_CFG\_SRC\_GPS, 1325  
CFE\_PLATFORM\_TIME\_CFG\_SRC\_MET, 1325  
CFE\_PLATFORM\_TIME\_CFG\_SRC\_TIME, 1325  
CFE\_PLATFORM\_TIME\_CFG\_START\_FLY, 1325  
CFE\_PLATFORM\_TIME\_CFG\_TONE\_LIMIT, 1326  
CFE\_PLATFORM\_TIME\_CFG\_VIRTUAL, 1326  
CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SECS, 1326  
CFE\_PLATFORM\_TIME\_MAX\_DELTA\_SUBS, 1327  
CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SECS, 1327  
CFE\_PLATFORM\_TIME\_MAX\_LOCAL\_SUBS, 1327  
CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_PRIORITY, 1327  
CFE\_PLATFORM\_TIME\_ONEHZ\_TASK\_STACK\_SIZE, 1327  
CFE\_PLATFORM\_TIME\_START\_TASK\_PRIORITY, 1327  
CFE\_PLATFORM\_TIME\_START\_TASK\_STACK\_SIZE, 1328  
CFE\_PLATFORM\_TIME\_TONE\_TASK\_PRIORITY, 1328

CFE\_PLATFORM\_TIME\_TONE\_TASK\_STACK\_SIZE,  
1328  
ExceptionAction  
  CFE\_ES\_AppInfo, 632  
  CFE\_ES\_StartAppCmd\_Payload, 669  
ExecutionCounter  
  CFE\_ES\_AppInfo, 632  
  CFE\_ES\_TaskInfo, 673  
  
fail\_dir  
  CF\_ConfigTable, 551  
FailedValCounter  
  CFE\_TBL\_HousekeepingTlm\_Payload, 741  
fault  
  CF\_HkCounters, 573  
fd  
  CF\_Logical\_IntHeader, 581  
  CF\_Transaction, 614  
  CF\_Traverse\_WriteHistoryFileArg, 620  
  CF\_Traverse\_WriteTxnFileArg, 621  
fd\_nak\_pending  
  CF\_Flags\_Tx, 565  
fdirective  
  CF\_CFDP\_FileDirectiveDispatchTable\_t, 527  
  CF\_Logical\_PduBuffer, 584  
FGV  
  cf\_codec.c, 1140  
file\_data\_bytes  
  CF\_HkRecv, 578  
  CF\_HkSent, 579  
file\_open  
  CF\_HkFault, 575  
file\_read  
  CF\_HkFault, 575  
file\_rename  
  CF\_HkFault, 575  
file\_seek  
  CF\_HkFault, 575  
file\_size\_mismatch  
  CF\_HkFault, 575  
file\_status  
  CF\_Logical\_PduFin, 589  
file\_write  
  CF\_HkFault, 575  
 FileModeBits  
  os\_fstat\_t, 787  
FileName  
  CFE\_ES\_AppInfo, 632  
  CFE\_ES\_FileNameCmd\_Payload, 641  
  CFE\_FS\_FileWriteMetaData, 702  
  os\_dirent\_t, 785  
Filename  
  CFE\_SB\_WriteFileInfoCmd\_Payload, 728  
filename  
  os\_fsinfo\_t, 786  
  CFE\_FS\_FileWriteMetaData, 626  
  OS\_module\_prop\_t, 790  
FileSize  
  os\_fstat\_t, 787  
FileSubType  
  CFE\_FS\_FileWriteMetaData, 702  
FileTime  
  CFE\_TBL\_Info, 745  
  CFE\_TBL\_TblRegPacket\_Payload, 752  
  os\_fstat\_t, 787  
filter\_dir  
  CF\_Traverse\_WriteHistoryFileArg, 620  
FilterMask  
  CFE\_ES\_SetPerfFilterMaskCmd\_Payload, 666  
FilterMaskNum  
  CFE\_ES\_SetPerfFilterMaskCmd\_Payload, 666  
fin  
  CF\_Logical\_IntHeader, 582  
fin\_ack\_count  
  CF\_Flags\_Tx, 565  
fin\_count  
  CF\_Flags\_Tx, 565  
fin\_dc  
  CF\_StateData, 609  
fin\_fs  
  CF\_StateData, 609  
finack\_recv  
  CF\_Flags\_Rx, 564  
flags  
  CF\_CFDP\_PduFin, 531  
  CF\_CFDP\_PduHeader, 532  
  CF\_Transaction, 615  
  OS\_module\_address\_t, 789  
fn  
  CF\_CFDP\_Tick\_args, 536  
  CF\_TraverseAll\_Arg, 622  
fnames  
  CF\_EotPacket\_Payload, 560  
  CF\_History, 569  
  CF\_Playback, 601  
foffs  
  CF\_Transaction, 615  
Forced2Fly  
  CFE\_TIME\_DiagnosticTlm\_Payload, 762  
free\_blocks  
  OS\_heap\_prop\_t, 788  
free\_bytes  
  OS\_heap\_prop\_t, 788  
FreeFds  
  os\_fsinfo\_t, 786  
freerun\_time  
  OS\_timebase\_prop\_t, 797  
FreeVolumes  
  os\_fsinfo\_t, 786

frozen  
     CF\_HkChannel\_Data, 571

fsize  
     CF\_EotPacket\_Payload, 560  
     CF\_Transaction, 615

FSV  
     cf\_codec.c, 1140

GetData  
     CFE\_FS\_FileWriteMetaData, 702

GetPipeIdByNameErrorCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 710

Handle  
     CFE\_ES\_CDSRegDumpRec, 636

hdr  
     CF\_PduCmdMsg, 599  
     CF\_PduTlmMsg, 600

header\_encoded\_length  
     CF\_Logical\_PduHeader, 591

HeapBlocksFree  
     CFE\_ES\_HousekeepingTlm\_Payload, 645

HeapBytesFree  
     CFE\_ES\_HousekeepingTlm\_Payload, 645

HeapMaxBlockSize  
     CFE\_ES\_HousekeepingTlm\_Payload, 645

histories  
     CF\_Engine, 558

history  
     CF\_Transaction, 615

hk  
     CF\_AppData\_t, 526

host\_module\_id  
     OS\_module\_prop\_t, 790

hword  
     CF\_UnionArgs\_Payload, 625

in  
     CF\_Engine, 558

InactiveBufferAddr  
     CFE\_TBL\_TblRegPacket\_Payload, 752

inactivity\_fired  
     CF\_Flags\_Common, 562

inactivity\_timer  
     CF\_HkFault, 575  
     CF\_Transaction, 616

inactivity\_timer\_s  
     CF\_ChannelConfig, 544

Index  
     CFE\_SB\_MsgMapFileEntry, 714

index  
     CF\_Crc, 552

int16  
     common\_types.h, 1749

int32  
     common\_types.h, 1749

int64  
     common\_types.h, 1749

int8  
     common\_types.h, 1749

int\_header  
     CF\_Logical\_PduBuffer, 585

InternalErrorCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 710

interval\_sec  
     CF\_PollDir, 604

interval\_time  
     OS\_timer\_prop\_t, 798

interval\_timer  
     CF\_Poll, 602

intptr  
     common\_types.h, 1749

IntVal  
     OS\_socket\_optval, 794

is\_complete  
     CF\_Flags\_Common, 562

is\_valid  
     CF\_CodecState, 550

IsPending  
     CFE\_FS\_FileWriteMetaData, 702

IsValid  
     OS\_file\_prop\_t, 786

keep  
     CF\_Playback, 601  
     CF\_Transaction, 616  
     CF\_TxFile\_Payload, 623

keep\_history  
     CF\_Flags\_Common, 562

key  
     CF\_GetParam\_Payload, 568  
     CF\_SetParam\_Payload, 607

large\_flag  
     CF\_Logical\_PduHeader, 591

largest\_free\_block  
     OS\_heap\_prop\_t, 788

LastFileDumped  
     CFE\_TBL\_HousekeepingTlm\_Payload, 741

LastFileLoaded  
     CFE\_TBL\_HousekeepingTlm\_Payload, 741  
     CFE\_TBL\_Info, 745  
     CFE\_TBL\_TblRegPacket\_Payload, 753

LastTableLoaded  
     CFE\_TBL\_HousekeepingTlm\_Payload, 741

LastUpdatedTable  
     CFE\_TBL\_HousekeepingTlm\_Payload, 742

LastUpdateTime  
     CFE\_TBL\_HousekeepingTlm\_Payload, 742

LastValCrc

CFE\_TBL\_HousekeepingTlm\_Payload, 742  
LastValStatus  
    CFE\_TBL\_HousekeepingTlm\_Payload, 742  
LastValTableName  
    CFE\_TBL\_HousekeepingTlm\_Payload, 742  
LeapSeconds  
    CFE\_TIME\_HousekeepingTlm\_Payload, 768  
    CFE\_TIME\_LeapsCmd\_Payload, 769  
Length  
    CCSDS\_PrimaryHeader, 524  
    CFE\_FS\_Header, 704  
length  
    CF\_CFDP\_lv, 528  
    CF\_CFDP\_PduHeader, 532  
    CF\_CFDP\_tlv, 536  
    CF\_Logical\_Lv, 583  
    CF\_Logical\_Tlv, 595  
LENGTHCHECK  
    osapi-macros.h, 1771  
LoadFilename  
    CFE\_TBL\_LoadCmd\_Payload, 746  
LoadPending  
    CFE\_TBL\_TblRegPacket\_Payload, 753  
local\_eid  
    CF\_ConfigTable, 551  
LocalIntCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 762  
LocalTaskCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 762  
LogEnabled  
    CFE\_EVS\_HousekeepingTlm\_Payload, 688  
LogFilename  
    CFE\_EVS\_LogFileCmd\_Payload, 690  
LogFullFlag  
    CFE\_EVS\_HousekeepingTlm\_Payload, 688  
LogMode  
    CFE\_EVS\_HousekeepingTlm\_Payload, 688  
    CFE\_EVS\_SetLogMode\_Payload, 698  
LogOverflowCounter  
    CFE\_EVS\_HousekeepingTlm\_Payload, 688  
LongDouble  
    CFE\_ES\_PoolAlign, 656  
    CFE\_SB\_Msg, 712  
LongInt  
    CFE\_ES\_PoolAlign, 656  
    CFE\_SB\_Msg, 712  
MainTaskId  
    CFE\_ES\_AppInfo, 632  
MainTaskName  
    CFE\_ES\_AppInfo, 632  
Mask  
    CFE\_EVS\_AppNameEventIDMaskCmd\_Payload,  
        678  
            CFE\_EVS\_BinFilter, 680  
mask  
    CF\_Codec\_BitField, 549  
max\_chunks  
    CF\_ChunkList, 547  
max\_outgoing\_messages\_per\_wakeup  
    CF\_ChannelConfig, 545  
max\_size  
    CF\_CodecState, 550  
MaxElapsed  
    CFE\_TIME\_DiagnosticTlm\_Payload, 762  
MaxFds  
    os\_finfo\_t, 787  
MaxLocalClock  
    CFE\_TIME\_DiagnosticTlm\_Payload, 762  
MaxMemAllowed  
    CFE\_SB\_StatsTlm\_Payload, 724  
MaxMsgIdsAllowed  
    CFE\_SB\_StatsTlm\_Payload, 725  
MaxPipeDepthAllowed  
    CFE\_SB\_StatsTlm\_Payload, 725  
MaxPipesAllowed  
    CFE\_SB\_StatsTlm\_Payload, 725  
MaxPRCount  
    CFE\_ES\_SetMaxPRCountCmd\_Payload, 664  
MaxProcessorResets  
    CFE\_ES\_HousekeepingTlm\_Payload, 645  
MaxQueueDepth  
    CFE\_SB\_PipeDepthStats, 715  
    CFE\_SB\_PipeInfoEntry, 716  
MaxSubscriptionsAllowed  
    CFE\_SB\_StatsTlm\_Payload, 725  
MaxVolumes  
    os\_finfo\_t, 787  
md  
    CF\_Logical\_IntHeader, 582  
md\_recv  
    CF\_Flags\_Rx, 564  
MemInUse  
    CFE\_SB\_HousekeepingTlm\_Payload, 710  
    CFE\_SB\_StatsTlm\_Payload, 725  
MemPoolHandle  
    CFE\_SB\_HousekeepingTlm\_Payload, 710  
    CFE\_TBL\_HousekeepingTlm\_Payload, 742  
Message  
    CFE\_EVS\_LongEventTlm\_Payload, 691  
MessageFormatMode  
    CFE\_EVS\_HousekeepingTlm\_Payload, 689  
MessageSendCounter  
    CFE\_EVS\_HousekeepingTlm\_Payload, 689  
MessageTruncCounter  
    CFE\_EVS\_HousekeepingTlm\_Payload, 689  
MET  
    CFE\_TIME\_HousekeepingTlm\_Payload, 768

MicroSeconds  
     CFE\_TIME\_TimeCmd\_Payload, 780

mid\_input  
     CF\_ChannelConfig, 545

mid\_output  
     CF\_ChannelConfig, 545

MinElapsed  
     CFE\_TIME\_DiagnosticTlm\_Payload, 763

Mode  
     CFE\_ES\_OverWriteSysLogCmd\_Payload, 655

Module  
     OS\_static\_symbol\_record\_t, 795

move\_dir  
     CF\_ChannelConfig, 545

Msg  
     CFE\_SB\_Msg, 712

msg  
     CF\_Input, 580  
     CF\_Output, 598

MsgCnt  
     CFE\_SB\_RoutingFileEntry, 720

MsgFormat  
     CFE\_EVS\_SetEventFormatCode\_Payload, 697

MsgId  
     CFE\_SB\_MsgMapFileEntry, 714  
     CFE\_SB\_RouteCmd\_Payload, 719  
     CFE\_SB\_RoutingFileEntry, 720  
     CFE\_SB\_SingleSubscriptionTlm\_Payload, 722  
     CFE\_SB\_SubEntries, 727

MsgIdsInUse  
     CFE\_SB\_StatsTlm\_Payload, 725

MsgLimitErrorCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 710

MsgReceiveErrorCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 711

MsgSendErrorCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 711

nak  
     CF\_GapComputeArgs\_t, 567  
     CF\_Logical\_IntHeader, 582

nak\_limit  
     CF\_ChannelConfig, 545  
     CF\_HkFault, 575

nak\_segment\_requests  
     CF\_HkRecv, 578  
     CF\_HkSent, 579

nak\_timer\_s  
     CF\_ChannelConfig, 545

Name  
     CFE\_Config\_IdNameEntry, 628  
     CFE\_ES\_AppInfo, 633  
     CFE\_ES\_CDSRegDumpRec, 636  
     CFE\_TBL\_TblRegPacket\_Payload, 753

OS\_static\_symbol\_record\_t, 795

name  
     OS\_bin\_sem\_prop\_t, 783  
     OS\_condvar\_prop\_t, 783  
     OS\_count\_sem\_prop\_t, 784  
     OS\_module\_prop\_t, 790  
     OS\_mut\_sem\_prop\_t, 791  
     OS\_queue\_prop\_t, 791  
     OS\_rwlock\_prop\_t, 792  
     OS\_socket\_prop\_t, 794  
     OS\_task\_prop\_t, 796  
     OS\_timebase\_prop\_t, 798  
     OS\_timer\_prop\_t, 798

next  
     CF\_CListNode, 548

next\_offset  
     CF\_CodecState, 550

nominal\_interval\_time  
     OS\_timebase\_prop\_t, 798

NoSubscribersCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 711

num\_cmd\_tx  
     CF\_Channel, 542

num\_segments  
     CF\_Logical\_SegmentList, 594

num\_tlv  
     CF\_Logical\_TlvList, 597

num\_ts  
     CF\_Playback, 601

NumBlocksRequested  
     CFE\_ES\_MemPoolStats, 652

NumBytes  
     CFE\_TBL\_File\_Hdr, 737

NumCreated  
     CFE\_ES\_BlockStats, 635

NumElements  
     CFE\_Config\_ArrayValue, 628

NumFree  
     CFE\_ES\_BlockStats, 635

NumFreeBytes  
     CFE\_ES\_MemPoolStats, 652

NumFreeSharedBufs  
     CFE\_TBL\_HousekeepingTlm\_Payload, 742

NumLoadPending  
     CFE\_TBL\_HousekeepingTlm\_Payload, 743

NumOfChildTasks  
     CFE\_ES\_AppInfo, 633

NumTables  
     CFE\_TBL\_HousekeepingTlm\_Payload, 743

NumUsers  
     CFE\_TBL\_Info, 745

NumValRequests  
     CFE\_TBL\_HousekeepingTlm\_Payload, 743

object\_ids  
    OS\_FdSet, 785

ObjectName  
    CFE\_TBL\_FileDef, 738

ObjectSize  
    CFE\_TBL\_FileDef, 738

octets  
    CF\_CFDP\_uint16\_t, 538  
    CF\_CFDP\_uint32\_t, 539  
    CF\_CFDP\_uint64\_t, 539  
    CF\_CFDP\_uint8\_t, 539

Offset  
    CFE\_TBL\_File\_Hdr, 737

offset  
    CF\_CFDP\_PduFileDataHeader, 530  
    CF\_Chunk, 546  
    CF\_Logical\_PduFileDataHeader, 587

offset\_end  
    CF\_CFDP\_SegmentRequest, 535  
    CF\_Logical\_SegmentRequest, 595

offset\_start  
    CF\_CFDP\_SegmentRequest, 535  
    CF\_Logical\_SegmentRequest, 595

OneHzAdjust  
    CFE\_TIME\_DiagnosticTlm\_Payload, 763

OneHzDirection  
    CFE\_TIME\_DiagnosticTlm\_Payload, 763

OneTimeAdjust  
    CFE\_TIME\_DiagnosticTlm\_Payload, 763

OneTimeDirection  
    CFE\_TIME\_DiagnosticTlm\_Payload, 763

OnEvent  
    CFE\_FS\_FileWriteMetaData, 703

Opts  
    CFE\_SB\_PipeInfoEntry, 716

OS\_ADD\_TASK\_FLAGS  
    osconfig.h, 1263

OS\_API\_Init  
    OSAL Core Operation APIs, 416

OS\_API\_Teardown  
    OSAL Core Operation APIs, 416

OS\_Application\_Run  
    OSAL Core Operation APIs, 416

OS\_Application\_Startup  
    OSAL Core Operation APIs, 416

OS\_ApplicationExit  
    OSAL Core Operation APIs, 417

OS\_ApplicationShutdown  
    OSAL Core Operation APIs, 417

OS\_ArgCallback\_t  
    common\_types.h, 1749

OS\_bin\_sem\_prop\_t, 782  
    creator, 783  
    name, 783

                value, 783

OS\_BinSemCreate  
    OSAL Binary Semaphore APIs, 396

OS\_BinSemDelete  
    OSAL Binary Semaphore APIs, 397

OS\_BinSemFlush  
    OSAL Binary Semaphore APIs, 397

OS\_BinSemGetIdByName  
    OSAL Binary Semaphore APIs, 398

OS\_BinSemGetInfo  
    OSAL Binary Semaphore APIs, 398

OS\_BinSemGive  
    OSAL Binary Semaphore APIs, 399

OS\_BinSemTake  
    OSAL Binary Semaphore APIs, 399

OS\_BinSemTimedWait  
    OSAL Binary Semaphore APIs, 400

OS\_BSP\_GetArgC  
    OSAL BSP low level access APIs, 401

OS\_BSP\_GetArgV  
    OSAL BSP low level access APIs, 401

OS\_BSP\_GetResourceTypeConfig  
    OSAL BSP low level access APIs, 401

OS\_BSP\_SetExitCode  
    OSAL BSP low level access APIs, 401

OS\_BSP\_SetResourceTypeConfig  
    OSAL BSP low level access APIs, 401

OS\_BUFFER\_MSG\_DEPTH  
    osconfig.h, 1263

OS\_BUFFER\_SIZE  
    osconfig.h, 1263

OS\_BUILD\_BASELINE  
    osapi-version.h, 1786

OS\_BUILD\_CODENAME  
    osapi-version.h, 1786

OS\_BUILD\_DEV\_CYCLE  
    osapi-version.h, 1786

OS\_BUILD\_NUMBER  
    osapi-version.h, 1786

OS\_CFG\_MAX\_VERSION\_STR\_LEN  
    osapi-version.h, 1786

OS\_CHECK  
    osapi-constants.h, 1758

OS\_CHK\_ONLY  
    osapi-fs.h, 1768

OS\_chkfs  
    OSAL File System Level APIs, 455

OS\_chmod  
    OSAL Standard File APIs, 442

OS\_close  
    OSAL Standard File APIs, 442

OS\_CloseAllFiles  
    OSAL Standard File APIs, 443

OS\_CloseFileByName

OSAL Standard File APIs, 443  
**OS\_condvar\_prop\_t**, 783  
  creator, 783  
  name, 783  
**OS\_CondVarBroadcast**  
  OSAL Condition Variable APIs, 419  
**OS\_CondVarCreate**  
  OSAL Condition Variable APIs, 419  
**OS\_CondVarDelete**  
  OSAL Condition Variable APIs, 420  
**OS\_CondVarGetIdByName**  
  OSAL Condition Variable APIs, 421  
**OS\_CondVarGetInfo**  
  OSAL Condition Variable APIs, 421  
**OS\_CondVarLock**  
  OSAL Condition Variable APIs, 422  
**OS\_CondVarSignal**  
  OSAL Condition Variable APIs, 422  
**OS\_CondVarTimedWait**  
  OSAL Condition Variable APIs, 423  
**OS\_CondVarUnlock**  
  OSAL Condition Variable APIs, 423  
**OS\_CondVarWait**  
  OSAL Condition Variable APIs, 424  
**OS\_ConvertToArrayIndex**  
  OSAL Object ID Utility APIs, 465  
**OS\_count\_sem\_prop\_t**, 784  
  creator, 784  
  name, 784  
  value, 784  
**OS\_CountSemCreate**  
  OSAL Counting Semaphore APIs, 425  
**OS\_CountSemDelete**  
  OSAL Counting Semaphore APIs, 425  
**OS\_CountSemGetIdByName**  
  OSAL Counting Semaphore APIs, 426  
**OS\_CountSemGetInfo**  
  OSAL Counting Semaphore APIs, 426  
**OS\_CountSemGive**  
  OSAL Counting Semaphore APIs, 427  
**OS\_CountSemTake**  
  OSAL Counting Semaphore APIs, 427  
**OS\_CountSemTimedWait**  
  OSAL Counting Semaphore APIs, 428  
**OS\_cp**  
  OSAL Standard File APIs, 444  
**OS\_DeleteAllObjects**  
  OSAL Core Operation APIs, 417  
**OS\_DirectoryClose**  
  OSAL Directory APIs, 429  
**OS\_DirectoryOpen**  
  OSAL Directory APIs, 429  
**OS\_DirectoryRead**  
  OSAL Directory APIs, 430  
**OS\_DirectoryRewind**  
  OSAL Directory APIs, 430  
**os\_dirent\_t**, 784  
  FileName, 785  
**OS\_DIRENTRY\_NAME**  
  osapi-dir.h, 1760  
**OS\_ERR\_BAD\_ADDRESS**  
  OSAL Return Code Defines, 434  
**OS\_ERR\_EMPTY\_SET**  
  OSAL Return Code Defines, 434  
**OS\_ERR\_FILE**  
  OSAL Return Code Defines, 434  
**OS\_ERR\_INCORRECT\_OBJ\_STATE**  
  OSAL Return Code Defines, 434  
**OS\_ERR\_INCORRECT\_OBJ\_TYPE**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_INVALID\_ARGUMENT**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_INVALID\_ID**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_INVALID\_PRIORITY**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_INVALID\_SIZE**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_NAME\_NOT\_FOUND**  
  OSAL Return Code Defines, 435  
**os\_err\_name\_t**  
  osapi-error.h, 1763  
**OS\_ERR\_NAME\_TAKEN**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_NAME\_TOO\_LONG**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_NO\_FREE\_IDS**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_NOT\_IMPLEMENTED**  
  OSAL Return Code Defines, 435  
**OS\_ERR\_OBJECT\_IN\_USE**  
  OSAL Return Code Defines, 436  
**OS\_ERR\_OPERATION\_NOT\_SUPPORTED**  
  OSAL Return Code Defines, 436  
**OS\_ERR\_OUTPUT\_TOO\_LARGE**  
  OSAL Return Code Defines, 436  
**OS\_ERR\_SEM\_NOT\_FULL**  
  OSAL Return Code Defines, 436  
**OS\_ERR\_STREAM\_DISCONNECTED**  
  OSAL Return Code Defines, 436  
**OS\_ERR\_TRY AGAIN**  
  OSAL Return Code Defines, 436  
**OS\_ERROR**  
  OSAL Return Code Defines, 436  
**OS\_ERROR\_ADDRESS\_MISALIGNED**  
  OSAL Return Code Defines, 436  
**OS\_ERROR\_NAME\_LENGTH**  
  osapi-error.h, 1763

OS\_ERROR\_TIMEOUT  
    OSAL Return Code Defines, 436

OS\_EVENT\_MAX  
    osapi-common.h, 1757

OS\_EVENT\_RESERVED  
    osapi-common.h, 1756

OS\_EVENT\_RESOURCE\_ALLOCATED  
    osapi-common.h, 1756

OS\_EVENT\_RESOURCE\_CREATED  
    osapi-common.h, 1757

OS\_EVENT\_RESOURCE\_DELETED  
    osapi-common.h, 1757

OS\_Event\_t  
    osapi-common.h, 1756

OS\_EVENT\_TASK\_STARTUP  
    osapi-common.h, 1757

OS\_EventHandler\_t  
    osapi-common.h, 1756

OS\_FDGetInfo  
    OSAL Standard File APIs, 444

OS\_FdSet, 785  
    object\_ids, 785

OS\_FILE\_FLAG\_CREATE  
    osapi-file.h, 1767

OS\_FILE\_FLAG\_NONE  
    osapi-file.h, 1767

OS\_file\_flag\_t  
    osapi-file.h, 1766

OS\_FILE\_FLAG\_TRUNCATE  
    osapi-file.h, 1767

OS\_file\_prop\_t, 785  
    IsValid, 786  
    Path, 786  
    User, 786

OS\_FileAllocate  
    OSAL Standard File APIs, 445

OS\_FileOpenCheck  
    OSAL Standard File APIs, 445

OS\_FILESTAT\_EXEC  
    osapi-file.h, 1765

OS\_FILESTAT\_ISDIR  
    osapi-file.h, 1765

OS\_FILESTAT\_MODE  
    osapi-file.h, 1765

OS\_FILESTAT\_MODE\_DIR  
    osapi-file.h, 1766

OS\_FILESTAT\_MODE\_EXEC  
    osapi-file.h, 1766

OS\_FILESTAT\_MODE\_READ  
    osapi-file.h, 1766

OS\_FILESTAT\_MODE\_WRITE  
    osapi-file.h, 1766

OS\_FILESTAT\_READ  
    osapi-file.h, 1766

OS\_FILESTAT\_SIZE  
    osapi-file.h, 1766

OS\_FILESTAT\_TIME  
    osapi-file.h, 1766

OS\_FILESTAT\_WRITE  
    osapi-file.h, 1766

OS\_FileSysAddFixedMap  
    OSAL File System Level APIs, 456

OS\_FileSysStatVolume  
    OSAL File System Level APIs, 456

OS\_FileTruncate  
    OSAL Standard File APIs, 446

OS\_ForEachObject  
    OSAL Object ID Utility APIs, 466

OS\_ForEachObjectOfType  
    OSAL Object ID Utility APIs, 466

OS\_FP\_ENABLED  
    osapi-task.h, 1782

OS\_FS\_DEV\_NAME\_LEN  
    osconfig.h, 1263

OS\_FS\_ERR\_DEVICE\_NOT\_FREE  
    OSAL Return Code Defines, 436

OS\_FS\_ERR\_DRIVE\_NOT\_CREATED  
    OSAL Return Code Defines, 437

OS\_FS\_ERR\_NAME\_TOO\_LONG  
    OSAL Return Code Defines, 437

OS\_FS\_ERR\_PATH\_INVALID  
    OSAL Return Code Defines, 437

OS\_FS\_ERR\_PATH\_TOO\_LONG  
    OSAL Return Code Defines, 437

OS\_FS\_GetPhysDriveName  
    OSAL File System Level APIs, 457

OS\_FS\_PHYS\_NAME\_LEN  
    osconfig.h, 1263

OS\_FS\_VOL\_NAME\_LEN  
    osconfig.h, 1263

os\_fsinfo\_t, 786  
    FreeFds, 786  
    FreeVolumes, 786  
    MaxFds, 787  
    MaxVolumes, 787

os\_fstat\_t, 787  
     FileModeBits, 787  
    FileSize, 787  
    FileTime, 787

OS\_GetBuildNumber  
    osapi-version.h, 1787

OS\_GetErrorName  
    OSAL Error Info APIs, 439

OS\_GetFsInfo  
    OSAL File System Level APIs, 457

OS\_GetLocalTime  
    OSAL Real Time Clock APIs, 402

OS\_GetMonotonicTime

OSAL Real Time Clock APIs, 403  
**OS\_GetResourceName**  
  OSAL Object ID Utility APIs, 467  
**OS\_GetVersionCodeName**  
  osapi-version.h, 1788  
**OS\_GetVersionNumber**  
  osapi-version.h, 1788  
**OS\_GetVersionString**  
  osapi-version.h, 1788  
**OS\_heap\_prop\_t**, 788  
  free\_blocks, 788  
  free\_bytes, 788  
  largest\_free\_block, 788  
**OS\_HeapGetInfo**  
  OSAL Heap APIs, 462  
**OS\_IdentifyObject**  
  OSAL Object ID Utility APIs, 467  
**OS\_IdleLoop**  
  OSAL Core Operation APIs, 417  
**OS\_initfs**  
  OSAL File System Level APIs, 458  
**OS\_INVALID\_INT\_NUM**  
  OSAL Return Code Defines, 437  
**OS\_INVALID\_POINTER**  
  OSAL Return Code Defines, 437  
**OS\_INVALID\_SEM\_VALUE**  
  OSAL Return Code Defines, 437  
**OS\_LAST\_OFFICIAL**  
  osapi-version.h, 1786  
**OS\_lseek**  
  OSAL Standard File APIs, 446  
**OS\_MAJOR\_VERSION**  
  osapi-version.h, 1786  
**OS\_MAX\_API\_NAME**  
  osconfig.h, 1263  
**OS\_MAX\_BIN\_SEMAPHORES**  
  osconfig.h, 1263  
**OS\_MAX\_CMD\_LEN**  
  osconfig.h, 1263  
**OS\_MAX\_CONDVARS**  
  osconfig.h, 1264  
**OS\_MAX\_CONSOLES**  
  osconfig.h, 1264  
**OS\_MAX\_COUNT\_SEMAPHORES**  
  osconfig.h, 1264  
**OS\_MAX\_FILE\_NAME**  
  osconfig.h, 1264  
**OS\_MAX\_FILE\_SYSTEMS**  
  osconfig.h, 1264  
**OS\_MAX\_LOCAL\_PATH\_LEN**  
  osapi-constants.h, 1758  
**OS\_MAX\_MODULES**  
  osconfig.h, 1264  
**OS\_MAX\_MUTEXES**  
  osconfig.h, 1264  
**OS\_MAX\_NUM\_OPEN\_DIRS**  
  osconfig.h, 1265  
**OS\_MAX\_NUM\_OPEN\_FILES**  
  osconfig.h, 1265  
**OS\_MAX\_PATH\_LEN**  
  osconfig.h, 1265  
**OS\_MAX\_QUEUES**  
  osconfig.h, 1265  
**OS\_MAX\_RWLOCKS**  
  osconfig.h, 1265  
**OS\_MAX\_SYM\_LEN**  
  osconfig.h, 1265  
**OS\_MAX\_TASK\_PRIORITY**  
  osapi-task.h, 1782  
**OS\_MAX\_TASKS**  
  osconfig.h, 1265  
**OS\_MAX\_TIMEBASES**  
  osconfig.h, 1266  
**OS\_MAX\_TIMERS**  
  osconfig.h, 1266  
**OS\_MINOR\_VERSION**  
  osapi-version.h, 1786  
**OS\_MISSION\_REV**  
  osapi-version.h, 1786  
**OS\_mkdir**  
  OSAL Directory APIs, 431  
**OS\_mkfs**  
  OSAL File System Level APIs, 459  
**OS\_module\_address\_t**, 788  
  bss\_address, 789  
  bss\_size, 789  
  code\_address, 789  
  code\_size, 789  
  data\_address, 789  
  data\_size, 789  
  flags, 789  
  valid, 789  
**OS\_MODULE\_FILE\_EXTENSION**  
  osconfig.h, 1266  
**OS\_MODULE\_FLAG\_GLOBAL\_SYMBOLS**  
  osapi-module.h, 1773  
**OS\_MODULE\_FLAG\_LOCAL\_SYMBOLS**  
  osapi-module.h, 1773  
**OS\_module\_prop\_t**, 789  
  addr, 790  
  entry\_point, 790  
  filename, 790  
  host\_module\_id, 790  
  name, 790  
**OS\_ModuleInfo**  
  OSAL Dynamic Loader and Symbol APIs, 470  
**OS\_ModuleLoad**  
  OSAL Dynamic Loader and Symbol APIs, 471

OS\_ModuleSymbolLookup  
    OSAL Dynamic Loader and Symbol APIs, 471

OS\_ModuleUnload  
    OSAL Dynamic Loader and Symbol APIs, 472

OS\_mount  
    OSAL File System Level APIs, 459

OS\_mut\_sem\_prop\_t, 790  
    creator, 790  
    name, 791

OS\_MutSemCreate  
    OSAL Mutex APIs, 474

OS\_MutSemDelete  
    OSAL Mutex APIs, 474

OS\_MutSemGetIdByName  
    OSAL Mutex APIs, 475

OS\_MutSemGetInfo  
    OSAL Mutex APIs, 475

OS\_MutSemGive  
    OSAL Mutex APIs, 476

OS\_MutSemTake  
    OSAL Mutex APIs, 476

OS\_mv  
    OSAL Standard File APIs, 447

OS\_NetworkGetHostName  
    OSAL Network ID APIs, 477

OS\_NetworkGetID  
    OSAL Network ID APIs, 478

OS\_OBJECT\_CREATOR\_ANY  
    osapi-constants.h, 1758

OS\_OBJECT\_ID\_UNDEFINED  
    osapi-constants.h, 1758

OS\_OBJECT\_INDEX\_MASK  
    osapi-idmap.h, 1770

OS\_OBJECT\_TYPE\_OS\_BINSEM  
    OSAL Object Type Defines, 463

OS\_OBJECT\_TYPE\_OS\_CONDVAR  
    OSAL Object Type Defines, 463

OS\_OBJECT\_TYPE\_OS\_CONSOLE  
    OSAL Object Type Defines, 463

OS\_OBJECT\_TYPE\_OS\_COUNTSEM  
    OSAL Object Type Defines, 463

OS\_OBJECT\_TYPE\_OS\_DIR  
    OSAL Object Type Defines, 463

OS\_OBJECT\_TYPE\_OS\_FILESYS  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_MODULE  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_MUTEX  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_QUEUE  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_RWLOCK  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_STREAM  
    OSAL Object Type Defines, 464

OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_TASK  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_TIMEBASE  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_OS\_TIMECB  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_SHIFT  
    osapi-idmap.h, 1770

OS\_OBJECT\_TYPE\_UNDEFINED  
    OSAL Object Type Defines, 464

OS\_OBJECT\_TYPE\_USER  
    OSAL Object Type Defines, 465

OS\_ObjectIdDefined  
    OSAL Object ID Utility APIs, 468

OS\_ObjectIdEqual  
    OSAL Object ID Utility APIs, 468

OS\_ObjectIdFromInteger  
    OSAL Object ID Utility APIs, 468

OS\_ObjectIdToArrayIndex  
    OSAL Object ID Utility APIs, 469

OS\_ObjectIdToInteger  
    OSAL Object ID Utility APIs, 469

OS\_OpenCreate  
    OSAL Standard File APIs, 447

OS\_PEND  
    osapi-constants.h, 1759

OS\_PRINTF  
    cfe\_es.h, 1356  
    common\_types.h, 1748

OS\_printf  
    OSAL Printf APIs, 478

OS\_PRINTF\_CONSOLE\_NAME  
    osconfig.h, 1266

OS\_printf\_disable  
    OSAL Printf APIs, 479

OS\_printf\_enable  
    OSAL Printf APIs, 479

OS\_QUEUE\_EMPTY  
    OSAL Return Code Defines, 437

OS\_QUEUE\_FULL  
    OSAL Return Code Defines, 437

OS\_QUEUE\_ID\_ERROR  
    OSAL Return Code Defines, 437

OS\_QUEUE\_INVALID\_SIZE  
    OSAL Return Code Defines, 438

OS\_QUEUE\_MAX\_DEPTH  
    osconfig.h, 1266

OS\_queue\_prop\_t, 791  
    creator, 791  
    name, 791

OS\_QUEUE\_TIMEOUT  
    OSAL Return Code Defines, 438

OS\_QueueCreate

OSAL Message Queue APIs, 479  
OS\_QueueDelete  
    OSAL Message Queue APIs, 480  
OS\_QueueGet  
    OSAL Message Queue APIs, 480  
OS\_QueueGetByName  
    OSAL Message Queue APIs, 481  
OS\_QueueGetInfo  
    OSAL Message Queue APIs, 482  
OS\_QueuePut  
    OSAL Message Queue APIs, 482  
OS\_read  
    OSAL Standard File APIs, 448  
OS\_READ\_ONLY  
    OSAL File Access Option Defines, 440  
OS\_READ\_WRITE  
    OSAL File Access Option Defines, 440  
OS\_RegisterEventHandler  
    OSAL Core Operation APIs, 417  
OS\_remove  
    OSAL Standard File APIs, 449  
OS\_rename  
    OSAL Standard File APIs, 449  
OS\_REPAIR  
    osapi-filesystem.h, 1768  
OS\_REVISION  
    osapi-version.h, 1787  
OS\_rmdir  
    OSAL Directory APIs, 431  
OS\_rmfs  
    OSAL File System Level APIs, 460  
OS\_rwlock\_prop\_t, 791  
    creator, 792  
    name, 792  
OS\_RwLockCreate  
    OSAL RwLock APIs, 483  
OS\_RwLockDelete  
    OSAL RwLock APIs, 484  
OS\_RwLockGetIdByName  
    OSAL RwLock APIs, 484  
OS\_RwLockGetInfo  
    OSAL RwLock APIs, 485  
OS\_RwLockReadGive  
    OSAL RwLock APIs, 485  
OS\_RwLockReadTake  
    OSAL RwLock APIs, 486  
OS\_RwLockWriteGive  
    OSAL RwLock APIs, 486  
OS\_RwLockWriteTake  
    OSAL RwLock APIs, 487  
OS\_SEEK\_CUR  
    OSAL Reference Point For Seek Offset Defines, 441  
OS\_SEEK\_END  
    OSAL Reference Point For Seek Offset Defines, 441  
OS\_SEEK\_SET  
    OSAL Reference Point For Seek Offset Defines, 441  
OS\_SelectFdAdd  
    OSAL Select APIs, 488  
OS\_SelectFdClear  
    OSAL Select APIs, 488  
OS\_SelectFdIsSet  
    OSAL Select APIs, 489  
OS\_SelectFdZero  
    OSAL Select APIs, 489  
OS\_SelectMultiple  
    OSAL Select APIs, 489  
OS\_SelectMultipleAbs  
    OSAL Select APIs, 490  
OS\_SelectSingle  
    OSAL Select APIs, 491  
OS\_SelectSingleAbs  
    OSAL Select APIs, 492  
OS\_SEM\_EMPTY  
    OSAL Semaphore State Defines, 395  
OS\_SEM\_FAILURE  
    OSAL Return Code Defines, 438  
OS\_SEM\_FULL  
    OSAL Semaphore State Defines, 396  
OS\_SEM\_TIMEOUT  
    OSAL Return Code Defines, 438  
OS\_SetLocalTime  
    OSAL Real Time Clock APIs, 403  
OS\_SHELL\_CMD\_INPUT\_FILE\_NAME  
    osconfig.h, 1266  
OS\_ShellOutputToFile  
    OSAL Shell APIs, 493  
OS\_SOCKET\_MAX\_LEN  
    osapi-sockets.h, 1779  
    osconfig.h, 1266  
OS\_SockAddr\_t, 792  
    ActualLength, 792  
    AddrData, 792  
OS\_SockAddrData\_t, 792  
    AlignPtr, 793  
    AlignU32, 793  
    Buffer, 793  
OS\_socket\_option  
    osapi-sockets.h, 1780  
OS\_socket\_option\_IP\_DSCP  
    osapi-sockets.h, 1780  
OS\_socket\_option\_MAX  
    osapi-sockets.h, 1780  
OS\_socket\_option\_t  
    osapi-sockets.h, 1779  
OS\_socket\_option\_UNDEFINED  
    osapi-sockets.h, 1780  
OS\_socket\_optval, 793  
    IntVal, 794

OS\_socket\_optval\_t  
    osapi-sockets.h, 1779

OS\_socket\_prop\_t, 794  
    creator, 794  
    name, 794

OS\_SocketAccept  
    OSAL Socket Management APIs, 498

OS\_SocketAcceptAbs  
    OSAL Socket Management APIs, 499

OS\_SocketAddrFromString  
    OSAL Socket Address APIs, 494

OS\_SocketAddrGetPort  
    OSAL Socket Address APIs, 495

OS\_SocketAddrInit  
    OSAL Socket Address APIs, 495

OS\_SocketAddrSetPort  
    OSAL Socket Address APIs, 496

OS\_SocketAddrToString  
    OSAL Socket Address APIs, 496

OS\_SocketBind  
    OSAL Socket Management APIs, 500

OS\_SocketBindAddress  
    OSAL Socket Management APIs, 500

OS\_SocketConnect  
    OSAL Socket Management APIs, 501

OS\_SocketConnectAbs  
    OSAL Socket Management APIs, 501

OS\_SocketDomain\_INET  
    osapi-sockets.h, 1780

OS\_SocketDomain\_INET6  
    osapi-sockets.h, 1780

OS\_SocketDomain\_INVALID  
    osapi-sockets.h, 1780

OS\_SocketDomain\_MAX  
    osapi-sockets.h, 1780

OS\_SocketDomain\_t  
    osapi-sockets.h, 1780

OS\_SocketGetIdByName  
    OSAL Socket Management APIs, 502

OS\_SocketGetInfo  
    OSAL Socket Management APIs, 503

OS\_SocketGetOption  
    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\_SocketRecvFromAbs  
    OSAL Socket Management APIs, 505

OS\_SocketSendTo  
    OSAL Socket Management APIs, 506

OS\_SocketSetOption  
    OSAL Socket Management APIs, 507

OS\_SocketShutdown  
    OSAL Socket Management APIs, 507

OS\_SocketShutdownMode\_NONE  
    osapi-sockets.h, 1780

OS\_SocketShutdownMode\_SHUT\_READ  
    osapi-sockets.h, 1780

OS\_SocketShutdownMode\_SHUT\_READWRITE  
    osapi-sockets.h, 1780

OS\_SocketShutdownMode\_SHUT\_WRITE  
    osapi-sockets.h, 1780

OS\_SocketShutdownMode\_t  
    osapi-sockets.h, 1780

OS\_SocketType\_DATAGRAM  
    osapi-sockets.h, 1781

OS\_SocketType\_INVALID  
    osapi-sockets.h, 1781

OS\_SocketType\_MAX  
    osapi-sockets.h, 1781

OS\_SocketType\_STREAM  
    osapi-sockets.h, 1781

OS\_SocketType\_t  
    osapi-sockets.h, 1780

OS\_stat  
    OSAL Standard File APIs, 450

OS\_static\_symbol\_record\_t, 794  
    Address, 795  
    Module, 795  
    Name, 795

OS\_STATUS\_STRING\_LENGTH  
    osapi-error.h, 1763

os\_status\_string\_t  
    osapi-error.h, 1763

OS\_StatusToInteger  
    OSAL Error Info APIs, 439

OS\_StatusToString  
    OSAL Error Info APIs, 440

OS\_statvfs\_t, 795  
    block\_size, 795  
    blocks\_free, 795  
    total\_blocks, 795

OS\_STR  
    osapi-version.h, 1787

OS\_STR\_HELPER  
    osapi-version.h, 1787

OS\_STREAM\_STATE\_BOUND  
    osapi-select.h, 1777

OS\_STREAM\_STATE\_CONNECTED  
    osapi-select.h, 1777

OS\_STREAM\_STATE\_LISTENING  
    osapi-select.h, 1777

OS\_STREAM\_STATE\_READABLE  
    osapi-select.h, 1777

OS\_STREAM\_STATE\_WRITABLE

osapi-select.h, 1777  
OS\_StreamState\_t  
    osapi-select.h, 1777  
OS\_strlen  
    OSAL Core Operation APIs, 418  
OS\_SUCCESS  
    OSAL Return Code Defines, 438  
OS\_SymbolLookup  
    OSAL Dynamic Loader and Symbol APIs, 472  
OS\_SymbolTableDump  
    OSAL Dynamic Loader and Symbol APIs, 473  
OS\_task\_prop\_t, 796  
    creator, 796  
    name, 796  
    priority, 796  
    stack\_size, 796  
OS\_TaskCreate  
    OSAL Task APIs, 508  
OS\_TaskDelay  
    OSAL Task APIs, 509  
OS\_TaskDelete  
    OSAL Task APIs, 510  
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, 512  
OS\_TaskInstallDeleteHandler  
    OSAL Task APIs, 512  
OS\_TaskSetPriority  
    OSAL Task APIs, 512  
OS\_TIME\_MAX  
    osapi-clock.h, 1754  
OS\_TIME\_MIN  
    osapi-clock.h, 1754  
OS\_time\_t, 796  
    ticks, 797  
OS\_TIME\_TICK\_RESOLUTION\_NS  
    osapi-clock.h, 1755  
OS\_TIME\_TICKS\_PER\_MSEC  
    osapi-clock.h, 1755  
OS\_TIME\_TICKS\_PER\_SECOND  
    osapi-clock.h, 1755  
OS\_TIME\_TICKS\_PER\_USEC  
    osapi-clock.h, 1755  
OS\_TIME\_ZERO  
    osapi-clock.h, 1754  
OS\_TimeAdd  
    OSAL Real Time Clock APIs, 404  
OS\_TimeAssembleFromMicroseconds  
    OSAL Real Time Clock APIs, 404  
OS\_TimeAssembleFromMilliseconds  
    OSAL Real Time Clock APIs, 405  
OS\_TimeAssembleFromNanoseconds  
    OSAL Real Time Clock APIs, 405  
OS\_TimeAssembleFromSubseconds  
    OSAL Real Time Clock APIs, 406  
OS\_timebase\_prop\_t, 797  
    accuracy, 797  
    creator, 797  
    freerun\_time, 797  
    name, 798  
    nominal\_interval\_time, 798  
OS\_TimeBaseCreate  
    OSAL Time Base APIs, 513  
OS\_TimeBaseDelete  
    OSAL Time Base APIs, 514  
OS\_TimeBaseGetFreeRun  
    OSAL Time Base APIs, 515  
OS\_TimeBaseGetIdByName  
    OSAL Time Base APIs, 515  
OS\_TimeBaseGetInfo  
    OSAL Time Base APIs, 516  
OS\_TimeBaseSet  
    OSAL Time Base APIs, 517  
OS\_TimeCompare  
    OSAL Real Time Clock APIs, 406  
OS\_TimedRead  
    OSAL Standard File APIs, 451  
OS\_TimedReadAbs  
    OSAL Standard File APIs, 451  
OS\_TimedWrite  
    OSAL Standard File APIs, 452  
OS\_TimedWriteAbs  
    OSAL Standard File APIs, 453  
OS\_TimeEqual  
    OSAL Real Time Clock APIs, 407  
OS\_TimeFromRelativeMilliseconds  
    OSAL Real Time Clock APIs, 407  
OS\_TimeFromTotalMicroseconds  
    OSAL Real Time Clock APIs, 408  
OS\_TimeFromTotalMilliseconds  
    OSAL Real Time Clock APIs, 408  
OS\_TimeFromTotalNanoseconds  
    OSAL Real Time Clock APIs, 408  
OS\_TimeFromTotalSeconds  
    OSAL Real Time Clock APIs, 409  
OS\_TimeGetFractionalPart  
    OSAL Real Time Clock APIs, 409  
OS\_TimeGetMicrosecondsPart  
    OSAL Real Time Clock APIs, 410  
OS\_TimeGetMillisecondsPart  
    OSAL Real Time Clock APIs, 410

OS\_TimeGetNanosecondsPart  
    OSAL Real Time Clock APIs, 411

OS\_TimeGetSign  
    OSAL Real Time Clock APIs, 411

OS\_TimeGetSubsecondsPart  
    OSAL Real Time Clock APIs, 412

OS\_TimeGetTotalMicroseconds  
    OSAL Real Time Clock APIs, 412

OS\_TimeGetTotalMilliseconds  
    OSAL Real Time Clock APIs, 413

OS\_TimeGetTotalNanoseconds  
    OSAL Real Time Clock APIs, 413

OS\_TimeGetTotalSeconds  
    OSAL Real Time Clock APIs, 414

OS\_TIMER\_ERR\_INTERNAL  
    OSAL Return Code Defines, 438

OS\_TIMER\_ERR\_INVALID\_ARGS  
    OSAL Return Code Defines, 438

OS\_TIMER\_ERR\_TIMER\_ID  
    OSAL Return Code Defines, 438

OS\_TIMER\_ERR\_UNAVAILABLE  
    OSAL Return Code Defines, 438

OS\_timer\_prop\_t, 798

- accuracy, 798
- creator, 798
- interval\_time, 798
- name, 798
- start\_time, 798

OS\_TimerAdd  
    OSAL Timer APIs, 518

OS\_TimerCallback\_t  
    osapi-timer.h, 1785

OS\_TimerCreate  
    OSAL Timer APIs, 519

OS\_TimerDelete  
    OSAL Timer APIs, 520

OS\_TimerGetIdByName  
    OSAL Timer APIs, 520

OS\_TimerGetInfo  
    OSAL Timer APIs, 521

OS\_TimerSet  
    OSAL Timer APIs, 522

OS\_TimerSync\_t  
    osapi-timebase.h, 1784

OS\_TimeSubtract  
    OSAL Real Time Clock APIs, 414

OS\_TimeToRelativeMilliseconds  
    OSAL Real Time Clock APIs, 414

OS\_TranslatePath  
    OSAL File System Level APIs, 460

OS\_unmount  
    OSAL File System Level APIs, 461

OS\_USED  
    common\_types.h, 1748

OS\_UTILITYTASK\_PRIORITY  
    osconfig.h, 1266

OS\_UTILITYTASK\_STACK\_SIZE  
    osconfig.h, 1267

OS\_VERSION  
    osapi-version.h, 1787

OS\_write  
    OSAL Standard File APIs, 454

OS\_WRITE\_ONLY  
    OSAL File Access Option Defines, 440

OSAL Binary Semaphore APIs, 396

- OS\_BinSemCreate, 396
- OS\_BinSemDelete, 397
- OS\_BinSemFlush, 397
- OS\_BinSemGetIdByName, 398
- OS\_BinSemGetInfo, 398
- OS\_BinSemGive, 399
- OS\_BinSemTake, 399
- OS\_BinSemTimedWait, 400

OSAL BSP low level access APIs, 400

- OS\_BSP\_GetArgC, 401
- OS\_BSP\_GetArgV, 401
- OS\_BSP\_GetResourceTypeConfig, 401
- OS\_BSP\_SetExitCode, 401
- OS\_BSP\_SetResourceTypeConfig, 401

OSAL Condition Variable APIs, 418

- OS\_CondVarBroadcast, 419
- OS\_CondVarCreate, 419
- OS\_CondVarDelete, 420
- OS\_CondVarGetIdByName, 421
- OS\_CondVarGetInfo, 421
- OS\_CondVarLock, 422
- OS\_CondVarSignal, 422
- OS\_CondVarTimedWait, 423
- OS\_CondVarUnlock, 423
- OS\_CondVarWait, 424

OSAL Core Operation APIs, 415

- OS\_API\_Init, 416
- OS\_API\_Teardown, 416
- OS\_Application\_Run, 416
- OS\_Application\_Startup, 416
- OS\_ApplicationExit, 417
- OS\_ApplicationShutdown, 417
- OS\_DeleteAllObjects, 417
- OS\_IdleLoop, 417
- OS\_RegisterEventHandler, 417
- OS\_strnlen, 418

OSAL Counting Semaphore APIs, 424

- OS\_CountSemCreate, 425
- OS\_CountSemDelete, 425
- OS\_CountSemGetIdByName, 426
- OS\_CountSemGetInfo, 426
- OS\_CountSemGive, 427
- OS\_CountSemTake, 427

OS\_CountSemTimedWait, 428  
 OSAL Directory APIs, 428  
   OS\_DirectoryClose, 429  
   OS\_DirectoryOpen, 429  
   OS\_DirectoryRead, 430  
   OS\_DirectoryRewind, 430  
   OS\_mkdir, 431  
   OS\_rmdir, 431  
 OSAL Dynamic Loader and Symbol APIs, 470  
   OS\_ModuleInfo, 470  
   OS\_ModuleLoad, 471  
   OS\_ModuleSymbolLookup, 471  
   OS\_ModuleUnload, 472  
   OS\_SymbolLookup, 472  
   OS\_SymbolTableDump, 473  
 OSAL Error Info APIs, 439  
   OS\_GetErrorName, 439  
   OS\_StatusToInteger, 439  
   OS\_StatusToString, 440  
 OSAL File Access Option Defines, 440  
   OS\_READ\_ONLY, 440  
   OS\_READ\_WRITE, 440  
   OS\_WRITE\_ONLY, 440  
 OSAL File System Level APIs, 454  
   OS\_chkfs, 455  
   OS\_FileSysAddFixedMap, 456  
   OS\_FileSysStatVolume, 456  
   OS\_FS\_GetPhysDriveName, 457  
   OS\_GetFsInfo, 457  
   OS\_initfs, 458  
   OS\_mkfs, 459  
   OS\_mount, 459  
   OS\_rmfs, 460  
   OS\_TranslatePath, 460  
   OS\_unmount, 461  
 OSAL Heap APIs, 462  
   OS\_HeapGetInfo, 462  
 OSAL Message Queue APIs, 479  
   OS\_QueueCreate, 479  
   OS\_QueueDelete, 480  
   OS\_QueueGet, 480  
   OS\_QueueGetIdByName, 481  
   OS\_QueueGetInfo, 482  
   OS\_QueuePut, 482  
 OSAL Mutex APIs, 474  
   OS\_MutSemCreate, 474  
   OS\_MutSemDelete, 474  
   OS\_MutSemGetIdByName, 475  
   OS\_MutSemGetInfo, 475  
   OS\_MutSemGive, 476  
   OS\_MutSemTake, 476  
 OSAL Network ID APIs, 477  
   OS\_NetworkGetHostName, 477  
   OS\_NetworkGetID, 478  
 OSAL Object ID Utility APIs, 465  
   OS\_ConvertToArrayIndex, 465  
   OS\_ForEachObject, 466  
   OS\_ForEachObjectOfType, 466  
   OS\_GetResourceName, 467  
   OS\_IdentifyObject, 467  
   OS\_ObjectIdDefined, 468  
   OS\_ObjectIdEqual, 468  
   OS\_ObjectIdFromInteger, 468  
   OS\_ObjectIdToArrayIndex, 469  
   OS\_ObjectIdToInteger, 469  
 OSAL Object Type Defines, 462  
   OS\_OBJECT\_TYPE\_OS\_BINSEM, 463  
   OS\_OBJECT\_TYPE\_OS\_CONDVAR, 463  
   OS\_OBJECT\_TYPE\_OS\_CONSOLE, 463  
   OS\_OBJECT\_TYPE\_OS\_COUNTSEM, 463  
   OS\_OBJECT\_TYPE\_OS\_DIR, 463  
   OS\_OBJECT\_TYPE\_OS\_FILESYS, 464  
   OS\_OBJECT\_TYPE\_OS\_MODULE, 464  
   OS\_OBJECT\_TYPE\_OS\_MUTEX, 464  
   OS\_OBJECT\_TYPE\_OS\_QUEUE, 464  
   OS\_OBJECT\_TYPE\_OS\_RWLOCK, 464  
   OS\_OBJECT\_TYPE\_OS\_STREAM, 464  
   OS\_OBJECT\_TYPE\_OS\_TASK, 464  
   OS\_OBJECT\_TYPE\_OS\_TIMEBASE, 464  
   OS\_OBJECT\_TYPE\_OS\_TIMECB, 464  
   OS\_OBJECT\_TYPE\_UNDEFINED, 464  
   OS\_OBJECT\_TYPE\_USER, 465  
 OSAL Printf APIs, 478  
   OS\_printf, 478  
   OS\_printf\_disable, 479  
   OS\_printf\_enable, 479  
 OSAL Real Time Clock APIs, 401  
   OS\_GetLocalTime, 402  
   OS\_GetMonotonicTime, 403  
   OS\_SetLocalTime, 403  
   OS\_TimeAdd, 404  
   OS\_TimeAssembleFromMicroseconds, 404  
   OS\_TimeAssembleFromMilliseconds, 405  
   OS\_TimeAssembleFromNanoseconds, 405  
   OS\_TimeAssembleFromSubseconds, 406  
   OS\_TimeCompare, 406  
   OS\_TimeEqual, 407  
   OS\_TimeFromRelativeMilliseconds, 407  
   OS\_TimeFromTotalMicroseconds, 408  
   OS\_TimeFromTotalMilliseconds, 408  
   OS\_TimeFromTotalNanoseconds, 408  
   OS\_TimeFromTotalSeconds, 409  
   OS\_TimeGetFractionalPart, 409  
   OS\_TimeGetMicrosecondsPart, 410  
   OS\_TimeGetMillisecondsPart, 410  
   OS\_TimeGetNanosecondsPart, 411  
   OS\_TimeGetSign, 411  
   OS\_TimeGetSubsecondsPart, 412

OS\_TimeGetTotalMicroseconds, 412  
OS\_TimeGetTotalMilliseconds, 413  
OS\_TimeGetTotalNanoseconds, 413  
OS\_TimeGetTotalSeconds, 414  
OS\_TimeSubtract, 414  
OS\_TimeToRelativeMilliseconds, 414  
OSAL Reference Point For Seek Offset Defines, 440  
    OS\_SEEK\_CUR, 441  
    OS\_SEEK\_END, 441  
    OS\_SEEK\_SET, 441  
OSAL Return Code Defines, 432  
    OS\_ERR\_BAD\_ADDRESS, 434  
    OS\_ERR\_EMPTY\_SET, 434  
    OS\_ERR\_FILE, 434  
    OS\_ERR\_INCORRECT\_OBJ\_STATE, 434  
    OS\_ERR\_INCORRECT\_OBJ\_TYPE, 435  
    OS\_ERR\_INVALID\_ARGUMENT, 435  
    OS\_ERR\_INVALID\_ID, 435  
    OS\_ERR\_INVALID\_PRIORITY, 435  
    OS\_ERR\_INVALID\_SIZE, 435  
    OS\_ERR\_NAME\_NOT\_FOUND, 435  
    OS\_ERR\_NAME\_TAKEN, 435  
    OS\_ERR\_NAME\_TOO\_LONG, 435  
    OS\_ERR\_NO\_FREE\_IDS, 435  
    OS\_ERR\_NOT\_IMPLEMENTED, 435  
    OS\_ERR\_OBJECT\_IN\_USE, 436  
    OS\_ERR\_OPERATION\_NOT\_SUPPORTED, 436  
    OS\_ERR\_OUTPUT\_TOO\_LARGE, 436  
    OS\_ERR\_SEM\_NOT\_FULL, 436  
    OS\_ERR\_STREAM\_DISCONNECTED, 436  
    OS\_ERR\_TRY AGAIN, 436  
    OS\_ERROR, 436  
    OS\_ERROR\_ADDRESS\_MISALIGNED, 436  
    OS\_ERROR\_TIMEOUT, 436  
    OS\_FS\_ERR\_DEVICE\_NOT\_FREE, 436  
    OS\_FS\_ERR\_DRIVE\_NOT\_CREATED, 437  
    OS\_FS\_ERR\_NAME\_TOO\_LONG, 437  
    OS\_FS\_ERR\_PATH\_INVALID, 437  
    OS\_FS\_ERR\_PATH\_TOO\_LONG, 437  
    OS\_INVALID\_INT\_NUM, 437  
    OS\_INVALID\_POINTER, 437  
    OS\_INVALID\_SEM\_VALUE, 437  
    OS\_QUEUE\_EMPTY, 437  
    OS\_QUEUE\_FULL, 437  
    OS\_QUEUE\_ID\_ERROR, 437  
    OS\_QUEUE\_INVALID\_SIZE, 438  
    OS\_QUEUE\_TIMEOUT, 438  
    OS\_SEM\_FAILURE, 438  
    OS\_SEM\_TIMEOUT, 438  
    OS\_SUCCESS, 438  
    OS\_TIMER\_ERR\_INTERNAL, 438  
    OS\_TIMER\_ERR\_INVALID\_ARGS, 438  
    OS\_TIMER\_ERR\_TIMER\_ID, 438  
    OS\_TIMER\_ERR\_UNAVAILABLE, 438  
OSAL RwLock APIs, 483  
    OS\_RwLockCreate, 483  
    OS\_RwLockDelete, 484  
    OS\_RwLockGetIdByName, 484  
    OS\_RwLockGetInfo, 485  
    OS\_RwLockReadGive, 485  
    OS\_RwLockReadTake, 486  
    OS\_RwLockWriteGive, 486  
    OS\_RwLockWriteTake, 487  
OSAL Select APIs, 487  
    OS\_SelectFdAdd, 488  
    OS\_SelectFdClear, 488  
    OS\_SelectFdIsSet, 489  
    OS\_SelectFdZero, 489  
    OS\_SelectMultiple, 489  
    OS\_SelectMultipleAbs, 490  
    OS\_SelectSingle, 491  
    OS\_SelectSingleAbs, 492  
OSAL Semaphore State Defines, 395  
    OS\_SEM\_EMPTY, 395  
    OS\_SEM\_FULL, 396  
OSAL Shell APIs, 493  
    OS\_ShellOutputToFile, 493  
OSAL Socket Address APIs, 494  
    OS\_SocketAddrFromString, 494  
    OS\_SocketAddrGetPort, 495  
    OS\_SocketAddrInit, 495  
    OS\_SocketAddrSetPort, 496  
    OS\_SocketAddrToString, 496  
OSAL Socket Management APIs, 497  
    OS\_SocketAccept, 498  
    OS\_SocketAcceptAbs, 499  
    OS\_SocketBind, 500  
    OS\_SocketBindAddress, 500  
    OS\_SocketConnect, 501  
    OS\_SocketConnectAbs, 501  
    OS\_SocketGetIdByName, 502  
    OS\_SocketGetInfo, 503  
    OS\_SocketGetOption, 503  
    OS\_SocketListen, 504  
    OS\_SocketOpen, 504  
    OS\_SocketRecvFrom, 505  
    OS\_SocketRecvFromAbs, 505  
    OS\_SocketSendTo, 506  
    OS\_SocketSetOption, 507  
    OS\_SocketShutdown, 507  
OSAL Standard File APIs, 441  
    OS\_chmod, 442  
    OS\_close, 442  
    OS\_CloseAllFiles, 443  
    OS\_CloseFileByName, 443  
    OS\_cp, 444  
    OS\_FDGetInfo, 444  
    OS\_FileAllocate, 445

OS\_FileOpenCheck, 445  
OS\_FileTruncate, 446  
OS\_Iseek, 446  
OS\_mv, 447  
OS\_OpenCreate, 447  
OS\_read, 448  
OS\_remove, 449  
OS\_rename, 449  
OS\_stat, 450  
OS\_TimedRead, 451  
OS\_TimedReadAbs, 451  
OS\_TimedWrite, 452  
OS\_TimedWriteAbs, 453  
OS\_write, 454  
OSAL Task APIs, 508  
  OS\_TaskCreate, 508  
  OS\_TaskDelay, 509  
  OS\_TaskDelete, 510  
  OS\_TaskExit, 510  
  OS\_TaskFindIdBySystemData, 510  
  OS\_TaskGetId, 511  
  OS\_TaskGetIdByName, 511  
  OS\_TaskGetInfo, 512  
  OS\_TaskInstallDeleteHandler, 512  
  OS\_TaskSetPriority, 512  
OSAL Time Base APIs, 513  
  OS\_TimeBaseCreate, 513  
  OS\_TimeBaseDelete, 514  
  OS\_TimeBaseGetFreeRun, 515  
  OS\_TimeBaseGetIdByName, 515  
  OS\_TimeBaseGetInfo, 516  
  OS\_TimeBaseSet, 517  
OSAL Timer APIs, 517  
  OS\_TimerAdd, 518  
  OS\_TimerCreate, 519  
  OS\_TimerDelete, 520  
  OS\_TimerGetIdByName, 520  
  OS\_TimerGetInfo, 521  
  OS\_TimerSet, 522  
osal/docs/src/osal\_frontpage.dox, 1746  
osal/docs/src/osal\_fs.dox, 1746  
osal/docs/src/osal\_timer.dox, 1746  
osal/src/os/inc/common\_types.h, 1746  
osal/src/os/inc/osapi-binsem.h, 1751  
osal/src/os/inc/osapi-bsp.h, 1752  
osal/src/os/inc/osapi-clock.h, 1752  
osal/src/os/inc/osapi-common.h, 1755  
osal/src/os/inc/osapi-condvar.h, 1757  
osal/src/os/inc/osapi-constants.h, 1758  
osal/src/os/inc/osapi-countsem.h, 1759  
osal/src/os/inc/osapi-dir.h, 1759  
osal/src/os/inc/osapi-error.h, 1760  
osal/src/os/inc/osapi-file.h, 1763  
osal/src/os/inc/osapi-filesystems.h, 1767  
osal/src/os/inc/osapi-heap.h, 1768  
osal/src/os/inc/osapi-idmap.h, 1768  
osal/src/os/inc/osapi-macros.h, 1770  
osal/src/os/inc/osapi-module.h, 1772  
osal/src/os/inc/osapi-mutex.h, 1773  
osal/src/os/inc/osapi-network.h, 1774  
osal/src/os/inc/osapi-printf.h, 1774  
osal/src/os/inc/osapi-queue.h, 1775  
osal/src/os/inc/osapi-rwlock.h, 1775  
osal/src/os/inc/osapi-select.h, 1776  
osal/src/os/inc/osapi-shell.h, 1777  
osal/src/os/inc/osapi-sockets.h, 1777  
osal/src/os/inc/osapi-task.h, 1781  
osal/src/os/inc/osapi-timebase.h, 1783  
osal/src/os/inc/osapi-timer.h, 1784  
osal/src/os/inc/osapi-version.h, 1785  
osal/src/os/inc/osapi.h, 1788  
OSAL\_API\_VERSION  
  osapi-version.h, 1787  
OSAL\_BLOCKCOUNT\_C  
  common\_types.h, 1748  
osal\_blockcount\_t  
  common\_types.h, 1749  
OSAL\_CONFIG\_CONSOLE\_ASYNC  
  osconfig.h, 1267  
OSAL\_CONFIG\_INCLUDE\_DYNAMIC\_LOADER  
  osconfig.h, 1267  
OSAL\_CONFIG\_INCLUDE\_NETWORK  
  osconfig.h, 1267  
OSAL\_CONFIG\_INCLUDE\_STATIC\_LOADER  
  osconfig.h, 1267  
osal\_id\_t  
  common\_types.h, 1749  
OSAL\_INDEX\_C  
  common\_types.h, 1748  
osal\_index\_t  
  common\_types.h, 1749  
OSAL\_OBJTYPE\_C  
  common\_types.h, 1748  
osal\_objtype\_t  
  common\_types.h, 1749  
osal\_offset\_t  
  common\_types.h, 1750  
OSAL\_PRIORITY\_C  
  osapi-task.h, 1782  
osal\_priority\_t  
  osapi-task.h, 1782  
OSAL\_SIZE\_C  
  common\_types.h, 1748  
OSAL\_STACKPTR\_C  
  osapi-task.h, 1782  
osal\_stackptr\_t  
  osapi-task.h, 1783  
OSAL\_STATUS\_C

common\_types.h, 1748  
osal\_status\_t  
    common\_types.h, 1750  
osal\_task  
    osapi-task.h, 1783  
OSAL\_TASK\_STACK\_ALLOCATE  
    osapi-task.h, 1782  
OSALMajorVersion  
    CFE\_ES\_HousekeepingTlm\_Payload, 646  
OSALMinorVersion  
    CFE\_ES\_HousekeepingTlm\_Payload, 646  
OSALMissionRevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 646  
OSALRevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 646  
osapi-clock.h  
    OS\_TIME\_MAX, 1754  
    OS\_TIME\_MIN, 1754  
    OS\_TIME\_TICK\_RESOLUTION\_NS, 1755  
    OS\_TIME\_TICKS\_PER\_MSEC, 1755  
    OS\_TIME\_TICKS\_PER\_SECOND, 1755  
    OS\_TIME\_TICKS\_PER\_USEC, 1755  
    OS\_TIME\_ZERO, 1754  
osapi-common.h  
    OS\_EVENT\_MAX, 1757  
    OS\_EVENT\_RESERVED, 1756  
    OS\_EVENT\_RESOURCE\_ALLOCATED, 1756  
    OS\_EVENT\_RESOURCE\_CREATED, 1757  
    OS\_EVENT\_RESOURCE\_DELETED, 1757  
    OS\_Event\_t, 1756  
    OS\_EVENT\_TASK\_STARTUP, 1757  
    OS\_EventHandler\_t, 1756  
osapi-constants.h  
    OS\_CHECK, 1758  
    OS\_MAX\_LOCAL\_PATH\_LEN, 1758  
    OS\_OBJECT\_CREATOR\_ANY, 1758  
    OS\_OBJECT\_ID\_UNDEFINED, 1758  
    OS\_PEND, 1759  
osapi-dir.h  
    OS\_DIRENTRY\_NAME, 1760  
osapi-error.h  
    os\_err\_name\_t, 1763  
    OS\_ERROR\_NAME\_LENGTH, 1763  
    OS\_STATUS\_STRING\_LENGTH, 1763  
    os\_status\_string\_t, 1763  
osapi-file.h  
    OS\_FILE\_FLAG\_CREATE, 1767  
    OS\_FILE\_FLAG\_NONE, 1767  
    OS\_file\_flag\_t, 1766  
    OS\_FILE\_FLAG\_TRUNCATE, 1767  
    OS\_FILESTAT\_EXEC, 1765  
    OS\_FILESTAT\_ISDIR, 1765  
    OS\_FILESTAT\_MODE, 1765  
    OS\_FILESTAT\_MODE\_DIR, 1766  
    OS\_FILESTAT\_MODE\_EXEC, 1766  
    OS\_FILESTAT\_MODE\_READ, 1766  
    OS\_FILESTAT\_MODE\_WRITE, 1766  
    OS\_FILESTAT\_READ, 1766  
    OS\_FILESTAT\_SIZE, 1766  
    OS\_FILESTAT\_TIME, 1766  
    OS\_FILESTAT\_WRITE, 1766  
osapi-filesys.h  
    OS\_CHK\_ONLY, 1768  
    OS\_REPAIR, 1768  
osapi-idmap.h  
    OS\_OBJECT\_INDEX\_MASK, 1770  
    OS\_OBJECT\_TYPE\_SHIFT, 1770  
osapi-macros.h  
    ARGCHECK, 1770  
    BUGCHECK, 1771  
    BUGCHECK\_VOID, 1771  
    BUGREPORT, 1771  
    LENGTHCHECK, 1771  
osapi-module.h  
    OS\_MODULE\_FLAG\_GLOBAL\_SYMBOLS, 1773  
    OS\_MODULE\_FLAG\_LOCAL\_SYMBOLS, 1773  
osapi-select.h  
    OS\_STREAM\_STATE\_BOUND, 1777  
    OS\_STREAM\_STATE\_CONNECTED, 1777  
    OS\_STREAM\_STATE\_LISTENING, 1777  
    OS\_STREAM\_STATE\_READABLE, 1777  
    OS\_STREAM\_STATE\_WRITABLE, 1777  
    OS\_StreamState\_t, 1777  
osapi-sockets.h  
    OS SOCKADDR\_MAX\_LEN, 1779  
    OS\_socket\_option, 1780  
    OS\_socket\_option\_IP\_DSCP, 1780  
    OS\_socket\_option\_MAX, 1780  
    OS\_socket\_option\_t, 1779  
    OS\_socket\_option\_UNDEFINED, 1780  
    OS\_socket\_optval\_t, 1779  
    OS\_SocketDomain\_INET, 1780  
    OS\_SocketDomain\_INET6, 1780  
    OS\_SocketDomain\_INVALID, 1780  
    OS\_SocketDomain\_MAX, 1780  
    OS\_SocketDomain\_t, 1780  
    OS\_SocketShutdownMode\_NONE, 1780  
    OS\_SocketShutdownMode\_SHUT\_READ, 1780  
    OS\_SocketShutdownMode\_SHUT\_READWRITE, 1780  
    OS\_SocketShutdownMode\_SHUT\_WRITE, 1780  
    OS\_SocketShutdownMode\_t, 1780  
    OS\_SocketType\_DATAGRAM, 1781  
    OS\_SocketType\_INVALID, 1781  
    OS\_SocketType\_MAX, 1781  
    OS\_SocketType\_STREAM, 1781  
    OS\_SocketType\_t, 1780  
osapi-task.h

OS\_FP\_ENABLED, 1782  
 OS\_MAX\_TASK\_PRIORITY, 1782  
 OSAL\_PRIORITY\_C, 1782  
 osal\_priority\_t, 1782  
 OSAL\_STACKPTR\_C, 1782  
 osal\_stackptr\_t, 1783  
 osal\_task, 1783  
 OSAL\_TASK\_STACK\_ALLOCATE, 1782  
 osapi-timebase.h  
 OS\_TimerSync\_t, 1784  
 osapi-timer.h  
 OS\_TimerCallback\_t, 1785  
 osapi-version.h  
 OS\_BUILD\_BASELINE, 1786  
 OS\_BUILD\_CODENAME, 1786  
 OS\_BUILD\_DEV\_CYCLE, 1786  
 OS\_BUILD\_NUMBER, 1786  
 OS\_CFG\_MAX\_VERSION\_STR\_LEN, 1786  
 OS\_GetBuildNumber, 1787  
 OS\_GetVersionCodeName, 1788  
 OS\_GetVersionNumber, 1788  
 OS\_GetVersionString, 1788  
 OS\_LAST\_OFFICIAL, 1786  
 OS\_MAJOR\_VERSION, 1786  
 OS\_MINOR\_VERSION, 1786  
 OS\_MISSION\_REV, 1786  
 OS\_REVISION, 1787  
 OS\_STR, 1787  
 OS\_STR\_HELPER, 1787  
 OS\_VERSION, 1787  
 OSAL\_API\_VERSION, 1787  
 osconfig.h  
 OS\_ADD\_TASK\_FLAGS, 1263  
 OS\_BUFFER\_MSG\_DEPTH, 1263  
 OS\_BUFFER\_SIZE, 1263  
 OS\_FS\_DEV\_NAME\_LEN, 1263  
 OS\_FS\_PHYS\_NAME\_LEN, 1263  
 OS\_FS\_VOL\_NAME\_LEN, 1263  
 OS\_MAX\_API\_NAME, 1263  
 OS\_MAX\_BIN\_SEMAPHORES, 1263  
 OS\_MAX\_CMD\_LEN, 1263  
 OS\_MAX\_CONDVARS, 1264  
 OS\_MAX\_CONSOLES, 1264  
 OS\_MAX\_COUNT\_SEMAPHORES, 1264  
 OS\_MAX\_FILE\_NAME, 1264  
 OS\_MAX\_FILE\_SYSTEMS, 1264  
 OS\_MAX\_MODULES, 1264  
 OS\_MAX\_MUTEXES, 1264  
 OS\_MAX\_NUM\_OPEN\_DIRS, 1265  
 OS\_MAX\_NUM\_OPEN\_FILES, 1265  
 OS\_MAX\_PATH\_LEN, 1265  
 OS\_MAX\_QUEUES, 1265  
 OS\_MAX\_RWLOCKS, 1265  
 OS\_MAX\_SYM\_LEN, 1265  
 OS\_MAX\_TASKS, 1265  
 OS\_MAX\_TIMEBASES, 1266  
 OS\_MAX\_TIMERS, 1266  
 OS\_MODULE\_FILE\_EXTENSION, 1266  
 OS\_PRINTF\_CONSOLE\_NAME, 1266  
 OS\_QUEUE\_MAX\_DEPTH, 1266  
 OS\_SHELL\_CMD\_INPUT\_FILE\_NAME, 1266  
 OS SOCKADDR\_MAX\_LEN, 1266  
 OS\_UTILITYTASK\_PRIORITY, 1266  
 OS\_UTILITYTASK\_STACK\_SIZE, 1267  
 OSAL\_CONFIG\_CONSOLE\_ASYNC, 1267  
 OSAL\_CONFIG\_INCLUDE\_DYNAMIC\_LOADER, 1267  
 OSAL\_CONFIG\_INCLUDE\_NETWORK, 1267  
 OSAL\_CONFIG\_INCLUDE\_STATIC\_LOADER, 1267  
 out  
 CF\_Engine, 558  
 outgoing\_counter  
 CF\_Channel, 542  
 outgoing\_file\_chunk\_size  
 CF\_ConfigTable, 551  
 OutputPort  
 CFE\_EVS\_HousekeepingTlm\_Payload, 689  
 OwnerAppName  
 CFE\_TBL\_TblRegPacket\_Payload, 753  
 PacketID  
 CFE\_EVS\_LongEventTlm\_Payload, 691  
 CFE\_EVS\_ShortEventTlm\_Payload, 700  
 Parameter  
 CFE\_TBL\_NotifyCmd\_Payload, 748  
 Path  
 OS\_file\_prop\_t, 786  
 Payload  
 CF\_AbandonCmd, 524  
 CF\_CancelCmd, 526  
 CF\_DisableDequeueCmd, 553  
 CF\_DisableDirPollingCmd, 554  
 CF\_EnableDequeueCmd, 555  
 CF\_EnableDirPollingCmd, 556  
 CF\_EotPacket, 559  
 CF\_FreezeCmd, 566  
 CF\_GetParamCmd, 568  
 CF\_HkPacket, 576  
 CF\_PlaybackDirCmd, 602  
 CF\_PurgeQueueCmd, 605  
 CF\_ResetCountersCmd, 605  
 CF\_ResumeCmd, 606  
 CF\_SetParamCmd, 608  
 CF\_SuspendCmd, 611  
 CF\_ThawCmd, 612  
 CF\_TxFileCmd, 624  
 CF\_WriteQueueCmd, 627  
 CFE\_ES\_DeleteCDSCmd, 638

CFE\_ES\_DumpCDSRegistryCmd, 639  
CFE\_ES\_FileNameCmd, 640  
CFE\_ES\_HousekeepingTlm, 641  
CFE\_ES\_MemStatsTlm, 653  
CFE\_ES\_OneAppTlm, 654  
CFE\_ES\_OverWriteSysLogCmd, 655  
CFE\_ES\_QueryAllCmd, 657  
CFE\_ES\_QueryAllTasksCmd, 658  
CFE\_ES\_QueryOneCmd, 658  
CFE\_ES\_ReloadAppCmd, 659  
CFE\_ES\_RestartAppCmd, 661  
CFE\_ES\_RestartCmd, 661  
CFE\_ES\_SendMemPoolStatsCmd, 663  
CFE\_ES\_SetMaxPRCountCmd, 664  
CFE\_ES\_SetPerfFilterMaskCmd, 665  
CFE\_ES\_SetPerfTriggerMaskCmd, 666  
CFE\_ES\_StartApp, 668  
CFE\_ES\_StartPerfDataCmd, 670  
CFE\_ES\_StopAppCmd, 670  
CFE\_ES\_StopPerfDataCmd, 672  
CFE\_ES\_WriteERLogCmd, 674  
CFE\_ES\_WriteSysLogCmd, 674  
CFE\_EVS\_AddEventFilterCmd, 675  
CFE\_EVS\_DeleteEventFilterCmd, 681  
CFE\_EVS\_DisableAppEventsCmd, 682  
CFE\_EVS\_DisableAppEventTypeCmd, 682  
CFE\_EVS\_DisableEventTypeCmd, 683  
CFE\_EVS\_DisablePortsCmd, 684  
CFE\_EVS\_EnableAppEventsCmd, 684  
CFE\_EVS\_EnableAppEventTypeCmd, 685  
CFE\_EVS\_EnableEventTypeCmd, 685  
CFE\_EVS\_EnablePortsCmd, 686  
CFE\_EVS\_HousekeepingTlm, 687  
CFE\_EVS\_LongEventTlm, 691  
CFE\_EVS\_ResetAllFiltersCmd, 694  
CFE\_EVS\_ResetAppCounterCmd, 695  
CFE\_EVS\_ResetFilterCmd, 696  
CFE\_EVS\_SetEventFormatModeCmd, 697  
CFE\_EVS\_SetFilterCmd, 698  
CFE\_EVS\_SetLogModeCmd, 699  
CFE\_EVS\_ShortEventTlm, 700  
CFE\_EVS\_WriteAppDataFileCmd, 701  
CFE\_EVS\_WriteLogDataFileCmd, 701  
CFE\_SB\_AllSubscriptionsTlm, 705  
CFE\_SB\_DisableRouteCmd, 706  
CFE\_SB\_EnableRouteCmd, 707  
CFE\_SB\_HousekeepingTlm, 708  
CFE\_SB\_SingleSubscriptionTlm, 722  
CFE\_SB\_StatsTlm, 723  
CFE\_SB\_WriteMapInfoCmd, 729  
CFE\_SB\_WritePipeInfoCmd, 729  
CFE\_SB\_WriteRoutingInfoCmd, 730  
CFE\_TBL\_AbortLoadCmd, 730  
CFE\_TBL\_ActivateCmd, 731  
CFE\_TBL\_DeleteCDSCmd, 734  
CFE\_TBL\_DumpCmd, 734  
CFE\_TBL\_DumpRegistryCmd, 736  
CFE\_TBL\_HousekeepingTlm, 739  
CFE\_TBL\_LoadCmd, 746  
CFE\_TBL\_NotifyCmd, 747  
CFE\_TBL\_SendRegistryCmd, 749  
CFE\_TBL\_TableRegistryTlm, 750  
CFE\_TBL\_ValidateCmd, 754  
CFE\_TIME\_AddAdjustCmd, 756  
CFE\_TIME\_AddDelayCmd, 756  
CFE\_TIME\_AddOneHzAdjustmentCmd, 757  
CFE\_TIME\_DiagnosticTlm, 757  
CFE\_TIME\_HousekeepingTlm, 766  
CFE\_TIME\_SetLeapSecondsCmd, 772  
CFE\_TIME\_SetMETCmd, 773  
CFE\_TIME\_SetSignalCmd, 773  
CFE\_TIME\_SetSourceCmd, 774  
CFE\_TIME\_SetStateCmd, 774  
CFE\_TIME\_SetSTCFCmd, 775  
CFE\_TIME\_SetTimeCmd, 776  
CFE\_TIME\_SubAdjustCmd, 778  
CFE\_TIME\_SubDelayCmd, 778  
CFE\_TIME\_SubOneHzAdjustmentCmd, 779  
CFE\_TIME\_ToneDataCmd, 781  
pb  
  CF\_Poll, 603  
  CF\_Transaction, 616  
pdec  
  CF\_Logical\_PduBuffer, 585  
pdu  
  CF\_HkRecv, 579  
  CF\_HkSent, 580  
pdu\_header  
  CF\_Logical\_PduBuffer, 585  
pdu\_type  
  CF\_Logical\_PduHeader, 591  
PeakMemInUse  
  CFE\_SB\_StatsTlm\_Payload, 725  
PeakMsgIdsInUse  
  CFE\_SB\_StatsTlm\_Payload, 726  
PeakPipesInUse  
  CFE\_SB\_StatsTlm\_Payload, 726  
PeakQueueDepth  
  CFE\_SB\_PipeDepthStats, 715  
  CFE\_SB\_PipeInfoEntry, 717  
PeakSBBuffersInUse  
  CFE\_SB\_StatsTlm\_Payload, 726  
PeakSubscriptionsInUse  
  CFE\_SB\_StatsTlm\_Payload, 726  
peer\_cc  
  CF\_StateData, 610  
peer\_eid  
  CF\_EotPacket\_Payload, 560

CF\_History, 569  
 penc  
   CF\_Logical\_PduBuffer, 585  
 pending\_file  
   CF\_Playback, 601  
 PerfDataCount  
   CFE\_ES\_HousekeepingTlm\_Payload, 646  
 PerfDataEnd  
   CFE\_ES\_HousekeepingTlm\_Payload, 646  
 PerfDataStart  
   CFE\_ES\_HousekeepingTlm\_Payload, 646  
 PerfDataToWrite  
   CFE\_ES\_HousekeepingTlm\_Payload, 647  
 PerfFilterMask  
   CFE\_ES\_HousekeepingTlm\_Payload, 647  
 PerfMode  
   CFE\_ES\_HousekeepingTlm\_Payload, 647  
 PerfState  
   CFE\_ES\_HousekeepingTlm\_Payload, 647  
 PerfTriggerCount  
   CFE\_ES\_HousekeepingTlm\_Payload, 647  
 PerfTriggerMask  
   CFE\_ES\_HousekeepingTlm\_Payload, 647  
 ph  
   CF\_PduCmdMsg, 599  
   CF\_PduTlmMsg, 600  
 Pipe  
   CFE\_SB\_RouteCmd\_Payload, 719  
   CFE\_SB\_SingleSubscriptionTlm\_Payload, 723  
   CFE\_SB\_SubEntries, 727  
 pipe  
   CF\_Channel, 542  
 pipe\_depth\_input  
   CF\_ChannelConfig, 545  
 PipeDepthStats  
   CFE\_SB\_StatsTlm\_Payload, 726  
 PipeId  
   CFE\_SB\_PipeDepthStats, 715  
   CFE\_SB\_PipeInfoEntry, 717  
   CFE\_SB\_RoutingFileEntry, 720  
 PipeName  
   CFE\_SB\_PipeInfoEntry, 717  
   CFE\_SB\_RoutingFileEntry, 720  
 PipeOptsErrorCounter  
   CFE\_SB\_HousekeepingTlm\_Payload, 711  
 PipeOverflowErrorCounter  
   CFE\_SB\_HousekeepingTlm\_Payload, 711  
 PipesInUse  
   CFE\_SB\_StatsTlm\_Payload, 726  
 PktSegment  
   CFE\_SB\_AllSubscriptionsTlm\_Payload, 706  
 playback  
   CF\_Channel, 543  
 playback\_counter  
   CF\_HkChannel\_Data, 571  
 poll  
   CF\_Channel, 543  
 poll\_counter  
   CF\_HkChannel\_Data, 571  
 polldir  
   CF\_ChannelConfig, 545  
 PoolHandle  
   CFE\_ES\_PoolStatsTlm\_Payload, 657  
   CFE\_ES\_SendMemPoolStatsCmd\_Payload, 663  
 PoolSize  
   CFE\_ES\_MemPoolStats, 652  
 PoolStats  
   CFE\_ES\_PoolStatsTlm\_Payload, 657  
 prev  
   CF\_CListNode, 548  
 Priority  
   CFE\_ES\_AppInfo, 633  
   CFE\_ES\_StartAppCmd\_Payload, 669  
   CFE\_ES\_TaskInfo, 673  
   CFE\_SB\_Qos\_t, 717  
 priority  
   CF\_Playback, 601  
   CF\_PollDir, 604  
   CF\_Transaction, 616  
   CF\_Traverse\_PriorityArg, 618  
   CF\_TxFile\_Payload, 623  
   OS\_task\_prop\_t, 796  
 ProcessorID  
   CFE\_EVS\_PacketID, 693  
   CFE\_FS\_Header, 704  
 ProcessorResets  
   CFE\_ES\_HousekeepingTlm\_Payload, 647  
 psp/fsw/inc/cfe\_psp.h, 1789  
 psp/fsw/inc/cfe\_psp\_cache\_api.h, 1790  
 psp/fsw/inc/cfe\_psp\_cds\_api.h, 1792  
 psp/fsw/inc/cfe\_psp\_eepromaccess\_api.h, 1793  
 psp/fsw/inc/cfe\_psp\_endian.h, 1796  
 psp/fsw/inc/cfe\_psp\_error.h, 1798  
 psp/fsw/inc/cfe\_psp\_exception\_api.h, 1801  
 psp/fsw/inc/cfe\_psp\_id\_api.h, 1803  
 psp/fsw/inc/cfe\_psp\_memaccess\_api.h, 1803  
 psp/fsw/inc/cfe\_psp\_memrange\_api.h, 1807  
 psp/fsw/inc/cfe\_psp\_port\_api.h, 1813  
 psp/fsw/inc/cfe\_psp\_ssr\_api.h, 1815  
 psp/fsw/inc/cfe\_psp\_timertick\_api.h, 1816  
 psp/fsw/inc/cfe\_psp\_version\_api.h, 1818  
 psp/fsw/inc/cfe\_psp\_watchdog\_api.h, 1819  
 PSP\_DEBUG  
   cfe\_psp.h, 1790  
 PSP\_DEBUG\_LEV  
   cfe\_psp.h, 1790  
 PSPMajorVersion  
   CFE\_ES\_HousekeepingTlm\_Payload, 648

PSPMinorVersion  
    CFE\_ES\_HousekeepingTlm\_Payload, 648

PSPMissionRevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 648

PSPRevision  
    CFE\_ES\_HousekeepingTlm\_Payload, 648

Ptr  
    CFE\_ES\_PoolAlign, 656

q\_index  
    CF\_Flags\_Common, 562

q\_size  
    CF\_HkChannel\_Data, 571

Qos  
    CFE\_SB\_SingleSubscriptionTlm\_Payload, 723  
    CFE\_SB\_SubEntries, 727

qs  
    CF\_Channel, 543

queue  
    CF\_WriteQueue\_Payload, 627

recv  
    CF\_HkCounters, 573

RegisteredCoreApps  
    CFE\_ES\_HousekeepingTlm\_Payload, 648

RegisteredExternalApps  
    CFE\_ES\_HousekeepingTlm\_Payload, 648

RegisteredLibs  
    CFE\_ES\_HousekeepingTlm\_Payload, 648

RegisteredTasks  
    CFE\_ES\_HousekeepingTlm\_Payload, 649

Reliability  
    CFE\_SB\_Qos\_t, 718

reliable\_mode  
    CF\_Transaction, 616

Reserved  
    CFE\_TBL\_File\_Hdr, 737

ResetSubtype  
    CFE\_ES\_HousekeepingTlm\_Payload, 649

ResetType  
    CFE\_ES\_HousekeepingTlm\_Payload, 649

Resourceld  
    CFE\_ES\_AppInfo, 633

RestartType  
    CFE\_ES\_RestartCmd\_Payload, 662

result  
    CF\_Crc, 552

resume\_point  
    CF\_CFDP\_Tick\_args, 536

RunStatus  
    CF\_AppData\_t, 526

rx  
    CF\_CFDP\_TxnRecvDispatchTable\_t, 537  
    CF\_StateFlags, 611

rx\_crc\_calc\_bytes\_per\_wakeup

CF\_ConfigTable, 551

rx\_max\_messages\_per\_wakeup  
    CF\_ChannelConfig, 546

rx\_pdudata  
    CF\_Input, 580

same  
    CF\_ChanAction\_SuspResArg, 541

SBBuffersInUse  
    CFE\_SB\_StatsTlm\_Payload, 726

scope\_end  
    CF\_CFDP\_PduNak, 533  
    CF\_Logical\_PduNak, 593

scope\_start  
    CF\_CFDP\_PduNak, 533  
    CF\_Logical\_PduNak, 593

Seconds  
    CFE\_TIME\_OneHzAdjustmentCmd\_Payload, 770  
    CFE\_TIME\_SysTime, 780  
    CFE\_TIME\_TimeCmd\_Payload, 780

segment\_list  
    CF\_Logical\_PduFileDataHeader, 587  
    CF\_Logical\_PduNak, 593

segment\_meta\_flag  
    CF\_Logical\_PduHeader, 591

segmentation\_control  
    CF\_CFDP\_PduMd, 532  
    CF\_Logical\_PduHeader, 591

segments  
    CF\_Logical\_SegmentList, 594

sem\_id  
    CF\_Channel, 543

sem\_name  
    CF\_ChannelConfig, 546

send\_eof  
    CF\_Flags\_Tx, 565

send\_fin  
    CF\_Flags\_Rx, 564

send\_md  
    CF\_Flags\_Tx, 566

send\_nak  
    CF\_Flags\_Rx, 564

SendErrors  
    CFE\_SB\_PipeInfoEntry, 717

sent  
    CF\_HkCounters, 573

seq\_num  
    CF\_Engine, 558  
    CF\_EotPacket\_Payload, 561  
    CF\_History, 570

Sequence  
    CCSDS\_PrimaryHeader, 524

sequence\_num  
    CF\_Logical\_PduHeader, 591

ServerFlyState  
     CFE\_TIME\_DiagnosticTlm\_Payload, 763

shift  
     CF\_Codec\_BitField, 549

Size  
     CFE\_ES\_CDSRegDumpRec, 637  
     CFE\_TBL\_Info, 745  
     CFE\_TBL\_TblRegPacket\_Payload, 753

size  
     CF\_CFDP\_PduEof, 529  
     CF\_CFDP\_PduMd, 532  
     CF\_Chunk, 546  
     CF\_Logical\_PduEof, 586  
     CF\_Logical\_PduMd, 593

source\_eid  
     CF\_Logical\_PduHeader, 591

source\_filename  
     CF\_Logical\_PduMd, 593

SpacecraftID  
     CFE\_EVS\_PacketID, 693  
     CFE\_FS\_Header, 704

Spare  
     CFE\_ES\_TaskInfo, 673  
     CFE\_EVS\_AppNameBitMaskCmd\_Payload, 676  
     CFE\_EVS\_BitMaskCmd\_Payload, 680  
     CFE\_EVS\_SetEventFormatCode\_Payload, 697  
     CFE\_EVS\_SetLogMode\_Payload, 699  
     CFE\_SB\_PipeDepthStats, 715  
     CFE\_SB\_PipeInfoEntry, 717  
     CFE\_SB\_RouteCmd\_Payload, 719

spare  
     CF\_HkChannel\_Data, 572  
     CF\_HkFault, 575  
     CF\_HkPacket\_Payload, 577  
     CF\_SetParam\_Payload, 607  
     CF\_Transaction\_Payload, 617  
     CF\_WriteQueue\_Payload, 627

Spare1  
     CFE\_EVS\_HousekeepingTlm\_Payload, 689  
     CFE\_EVS\_LongEventTlm\_Payload, 692

Spare2  
     CFE\_EVS\_HousekeepingTlm\_Payload, 689  
     CFE\_EVS\_LongEventTlm\_Payload, 692

Spare2Align  
     CFE\_SB\_HousekeepingTlm\_Payload, 711

Spare3  
     CFE\_EVS\_HousekeepingTlm\_Payload, 689

spurious  
     CF\_HkRecv, 579

src\_dir  
     CF\_PollDir, 604

src\_eid  
     CF\_EotPacket\_Payload, 561  
     CF\_History, 570

CF\_Traverse\_TransSeqArg, 619

src\_filename  
     CF\_TxFile\_Payload, 623  
     CF\_TxnFilenames, 624

stack\_size  
     OS\_task\_prop\_t, 796

StackSize  
     CFE\_ES\_AppInfo, 633  
     CFE\_ES\_StartAppCmd\_Payload, 669  
     CFE\_ES\_TaskInfo, 673

start\_time  
     OS\_timer\_prop\_t, 798

StartAddress  
     CFE\_ES\_AppInfo, 633

State  
     CFE\_SB\_RoutingFileEntry, 720

state  
     CF\_CFDP\_R\_SubstateDispatchTable\_t, 533  
     CF\_EotPacket\_Payload, 561  
     CF\_Transaction, 616

state\_data  
     CF\_Transaction, 616

STCF  
     CFE\_TIME\_HousekeepingTlm\_Payload, 768

Std  
     CFE\_TBL\_CombinedFileHdr, 732

StreamId  
     CCSDS\_PrimaryHeader, 524

sub\_state  
     CF\_StateData, 610

SubscribeErrorCounter  
     CFE\_SB\_HousekeepingTlm\_Payload, 711

SubscriptionsInUse  
     CFE\_SB\_StatsTlm\_Payload, 727

Subseconds  
     CFE\_TIME\_OneHzAdjustmentCmd\_Payload, 770  
     CFE\_TIME\_SysTime, 780

substate  
     CF\_CFDP\_S\_SubstateRecvDispatchTable\_t, 534  
     CF\_CFDP\_S\_SubstateSendDispatchTable\_t, 534

Subsystem  
     CCSDS\_ExtendedHeader, 523

SubType  
     CFE\_FS\_Header, 704  
     CFE\_SB\_SingleSubscriptionTlm\_Payload, 723

SuccessValCounter  
     CFE\_TBL\_HousekeepingTlm\_Payload, 743

suspended  
     CF\_Flags\_Common, 563

SysLogBytesUsed  
     CFE\_ES\_HousekeepingTlm\_Payload, 649

SysLogEntries  
     CFE\_ES\_HousekeepingTlm\_Payload, 649

SysLogMode

CFE\_ES\_HousekeepingTlm\_Payload, 649  
SysLogSize  
    CFE\_ES\_HousekeepingTlm\_Payload, 649  
SystemId  
    CCSDS\_ExtendedHeader, 523

Table  
    CFE\_ES\_CDSRegDumpRec, 637

TableLoadedOnce  
    CFE\_TBL\_Info, 745  
    CFE\_TBL\_TblRegPacket\_Payload, 753

TableName  
    CFE\_TBL\_AbortLoadCmd\_Payload, 731  
    CFE\_TBL\_ActivateCmd\_Payload, 732  
    CFE\_TBL\_DelCDSCmd\_Payload, 733  
    CFE\_TBL\_DumpCmd\_Payload, 735  
    CFE\_TBL\_File\_Hdr, 737  
    CFE\_TBL\_FileDef, 738  
    CFE\_TBL\_SendRegistryCmd\_Payload, 750  
    CFE\_TBL\_ValidateCmd\_Payload, 755

TaskId  
    CFE\_ES\_TaskInfo, 673

TaskName  
    CFE\_ES\_TaskInfo, 673

Tbl  
    CFE\_TBL\_CombinedFileHdr, 733

TelemetryHeader  
    CF\_EotPacket, 559  
    CF\_HkPacket, 576  
    CFE\_ES\_HousekeepingTlm, 641  
    CFE\_ES\_MemStatsTlm, 653  
    CFE\_ES\_OneAppTlm, 654  
    CFE\_EVS\_HousekeepingTlm, 687  
    CFE\_EVS\_LongEventTlm, 691  
    CFE\_EVS\_ShortEventTlm, 700  
    CFE\_SB\_AllSubscriptionsTlm, 705  
    CFE\_SB\_HousekeepingTlm, 708  
    CFE\_SB\_SingleSubscriptionTlm, 722  
    CFE\_SB\_StatsTlm, 723  
    CFE\_TBL\_HousekeepingTlm, 739  
    CFE\_TBL\_TableRegistryTlm, 750  
    CFE\_TIME\_DiagnosticTlm, 757  
    CFE\_TIME\_HousekeepingTlm, 766

tempfile\_created  
    CF\_Flags\_Rx, 564

TgtFilename  
    CFE\_TBL\_FileDef, 739

tick  
    CF\_Timer, 613

tick\_resume  
    CF\_Channel, 543

ticks  
    OS\_time\_t, 797

ticks\_per\_second

CF\_ConfigTable, 551  
TimeOfLastUpdate  
    CFE\_TBL\_Info, 745  
    CFE\_TBL\_TblRegPacket\_Payload, 753

timer\_set  
    CF\_Poll, 603

TimeSeconds  
    CFE\_FS\_Header, 704

TimeSinceTone  
    CFE\_TIME\_DiagnosticTlm\_Payload, 763

TimeSource  
    CFE\_TIME\_SourceCmd\_Payload, 777

TimeSubSeconds  
    CFE\_FS\_Header, 704

tlv  
    CF\_Logical\_TlvList, 597

tlv\_list  
    CF\_Logical\_PduEof, 586  
    CF\_Logical\_PduFin, 589

tmp\_dir  
    CF\_ConfigTable, 551

ToneDataCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 764

ToneDataLatch  
    CFE\_TIME\_DiagnosticTlm\_Payload, 764

ToneIntCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 764

ToneIntErrorCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 764

ToneMatchCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 764

ToneMatchErrorCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 764

ToneOverLimit  
    CFE\_TIME\_DiagnosticTlm\_Payload, 764

ToneSignalCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 765

ToneSignalLatch  
    CFE\_TIME\_DiagnosticTlm\_Payload, 765

ToneSource  
    CFE\_TIME\_SignalCmd\_Payload, 776

ToneTaskCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 765

ToneUnderLimit  
    CFE\_TIME\_DiagnosticTlm\_Payload, 765

total\_blocks  
    OS\_statvfs\_t, 795

TotalSegments  
    CFE\_SB\_AllSubscriptionsTlm\_Payload, 706

transaction\_sequence\_number  
    CF\_Traverse\_TransSeqArg, 619

transactions  
    CF\_Engine, 558

TriggerMask

CFE\_ES\_SetPerfTrigMaskCmd\_Payload, 667  
TriggerMaskNum  
    CFE\_ES\_SetPerfTrigMaskCmd\_Payload, 667  
TriggerMode  
    CFE\_ES\_StartPerfCmd\_Payload, 669  
ts  
    CF\_Transaction\_Payload, 618  
tx  
    CF\_CFDP\_TxnSendDispatchTable\_t, 538  
    CF\_StateFlags, 611  
tx\_blocked  
    CF\_Channel, 543  
tx\_pdudata  
    CF\_Output, 598  
txm\_mode  
    CF\_Logical\_PduHeader, 591  
txn  
    CF\_GapComputeArgs\_t, 567  
    CF\_Traverse\_PriorityArg, 618  
    CF\_Traverse\_TransSeqArg, 619  
txn\_seq\_length  
    CF\_Logical\_PduHeader, 592  
txn\_stat  
    CF\_EotPacket\_Payload, 561  
    CF\_History, 570  
txn\_status  
    CF\_Logical\_PduAck, 584  
Type  
    CFE\_ES\_AppInfo, 633  
type  
    CF\_CFDP\_tlv, 537  
    CF\_Logical\_Tlv, 596  
    CF\_WriteQueue\_Payload, 627

uint16  
    common\_types.h, 1750  
uint32  
    common\_types.h, 1750  
uint64  
    common\_types.h, 1750  
uint8  
    common\_types.h, 1750  
UnmarkedMem  
    CFE\_SB\_HousekeepingTlm\_Payload, 712  
UnregisteredAppCounter  
    CFE\_EVS\_HousekeepingTlm\_Payload, 690  
User  
    OS\_file\_prop\_t, 786  
UserDefAddr  
    CFE\_TBL\_Info, 745

valid  
    OS\_module\_address\_t, 789  
ValidationCounter  
    CFE\_TBL\_HousekeepingTlm\_Payload, 743

ValidationFuncPtr  
    CFE\_TBL\_TblRegPacket\_Payload, 754  
Value  
    CFE\_SB\_MsgId\_t, 713  
value  
    CF\_SetParam\_Payload, 607  
    OS\_bin\_sem\_prop\_t, 783  
    OS\_count\_sem\_prop\_t, 784  
version  
    CF\_Logical\_PduHeader, 592  
VersionCounter  
    CFE\_TIME\_DiagnosticTlm\_Payload, 765  
VirtualMET  
    CFE\_TIME\_DiagnosticTlm\_Payload, 765

working  
    CF\_Crc, 552