RTI Connext Cert C API

Version 2.4.16

Contents

1 Main Page 1

2 Module Index 1
2.1 Reference Manual . . . 1

3 Data Structure Index 4
3.1 Data Structures . . . 4

4 File Index 10
4.1 File List . . . 10

5 Module Documentation 13
5.1 Domain Module . . . 13
5.1.1 Detailed Description . . . 14
5.1.2 Typedef Documentation . . . 14
5.2 DomainParticipantFactory . . . 15
5.2.1 Detailed Description . . . 16
5.2.2 Macro Definition Documentation . . . 16
5.2.3 Typedef Documentation . . . 16
5.2.4 Function Documentation . . . 17
5.2.5 Variable Documentation . . . 21
5.3 DomainParticipant . . . 22
5.3.1 Detailed Description . . . 23
5.3.2 Macro Definition Documentation . . . 23
5.3.3 Typedef Documentation . . . 24
5.3.4 Function Documentation . . . 24
5.3.5 Variable Documentation . . . 30
5.4 Discovery . . . 32
5.4.1 Detailed Description . . . 33
5.4.2 Macro Definition Documentation . . . 33
5.4.3 Typedef Documentation . . . 34
5.5 Topic . . . 36
5.5.1 Detailed Description . . . 36
5.6 Topic . . . 37
5.6.1 Detailed Description . . . 38
5.6.2 Macro Definition Documentation . . . 38
5.6.3 Typedef Documentation . . . 38
5.6.4 Function Documentation . . . 39
5.7 User Data Type Support . . . 42
5.7.1 Detailed Description . . . 42
5.7.2 Macro Definition Documentation . . . 43
5.7.3 Typedef Documentation . . . 43
5.7.4 Function Documentation . . . 44
5.7.5 Variable Documentation . . . 45
5.8 Publication Module . . . 46
5.8.1 Detailed Description . . . 46
5.9 Publisher . . . 47
5.9.1 Detailed Description . . . 47
5.9.2 Macro Definition Documentation . . . 48
5.9.3 Typedef Documentation . . . 48
5.9.4 Function Documentation . . . 48
5.9.5 Variable Documentation . . . 51
5.10 DataWriter . . . 52
5.10.1 Detailed Description . . . 54
5.10.2 Macro Definition Documentation . . . 54
5.10.3 Typedef Documentation . . . 56
5.10.4 Function Documentation . . . 57

5.11 Subscription Module . . . 71
5.11.1 Detailed Description . . . 71
5.11.2 Access to data samples . . . 71
5.12 Subscriber . . . 73
5.12.1 Detailed Description . . . 74
5.12.2 Macro Definition Documentation . . . 74
5.12.3 Typedef Documentation . . . 74
5.12.4 Function Documentation . . . 75
5.12.5 Variable Documentation . . . 79
5.13 DataReader . . . 80
5.13.1 Detailed Description . . . 83
5.13.2 Macro Definition Documentation . . . 83
5.13.3 Typedef Documentation . . . 84
5.13.4 Enumeration Type Documentation . . . 86
5.13.5 Function Documentation . . . 87
5.13.6 SEQUENCES USAGE IN TAKE AND READ . . . 89
5.14 Data Sample . . . 101
5.14.1 Detailed Description . . . 101
5.15 Sample States . . . 102
5.15.1 Detailed Description . . . 102
5.15.2 Typedef Documentation . . . 102
5.15.3 Enumeration Type Documentation . . . 102
5.15.4 Variable Documentation . . . 103
5.16 View States . . . 104
5.16.1 Detailed Description . . . 104
5.16.2 Typedef Documentation . . . 104
5.16.3 Enumeration Type Documentation . . . 104
5.16.4 Variable Documentation . . . 105
5.17 Instance States . . . 106
5.17.1 Detailed Description . . . 106
5.17.2 Typedef Documentation . . . 106
5.17.3 Enumeration Type Documentation . . . 106
5.17.4 Variable Documentation . . . 107
5.18 Infrastructure Module . . . 108
5.18.1 Detailed Description . . . 108
5.19 User-Defined Types . . . 109
5.19.1 Detailed Description . . . 110
5.19.2 Typedef Documentation . . . 110
5.19.3 Enumeration Type Documentation . . . 112
5.19.4 Function Documentation . . . 112
5.20 Documentation Guide . . . 113
5.20.1 Unsupported Features . . . 113
5.20.2 API Naming Conventions . . . 113
5.20.3 API Documentation Terms . . . 113
5.20.4 Stereotypes . . . 113
5.21 DDS API . . . 115
5.21.1 Detailed Description . . . 115
5.21.2 Overview . . . 115
5.21.3 Conceptual Model . . . 116
5.21.4 Modules . . . 117
5.21.5 Overview . . . 117
5.21.6 Conceptual Model . . . 118
5.21.7 Modules . . . 120
5.22 DPSE Static Discovery API . . . 121
5.22.1 Detailed Description . . . 121
5.22.2 Macro Definition Documentation . . . 121
5.22.3 Function Documentation . . . 122
5.23 CDR Stream API . . . 125

5.23.1 Detailed Description . . . 125
5.23.2 Function Documentation . . . 125
5.24 RT Run-Time API . . . 128
5.24.1 Detailed Description . . . 128
5.24.2 Typedef Documentation . . . 128
5.24.3 Function Documentation . . . 128
5.25 RHSM Reader History API . . . 130
5.25.1 Detailed Description . . . 130
5.25.2 Function Documentation . . . 130
5.26 WHSM Writer History API . . . 131
5.26.1 Detailed Description . . . 131
5.26.2 Function Documentation . . . 131
5.27 NETIO API . . . 132
5.27.1 Detailed Description . . . 132
5.28 RTPS Transport API . . . 133
5.28.1 Detailed Description . . . 134
5.28.2 Typedef Documentation . . . 134
5.28.3 Enumeration Type Documentation . . . 134
5.28.4 Function Documentation . . . 135
5.28.5 Variable Documentation . . . 135
5.29 NETIO DGRAM API . . . 137
5.29.1 Detailed Description . . . 138
5.29.2 Macro Definition Documentation . . . 139
5.29.3 Typedef Documentation . . . 141
5.29.4 Function Documentation . . . 141
5.29.5 Variable Documentation . . . 143
5.30 Zero Copy v2 API . . . 145
5.30.1 Detailed Description . . . 145
5.30.2 Function Documentation . . . 145
5.30.3 Variable Documentation . . . 146
5.31 OS API . . . 147
5.31.1 Detailed Description . . . 147
5.32 Log API . . . 148
5.32.1 Detailed Description . . . 149
5.32.2 Typedef Documentation . . . 149
5.32.3 Enumeration Type Documentation . . . 149
5.32.4 Function Documentation . . . 149
5.33 REDA . . . 153
5.33.1 Detailed Description . . . 153
5.33.2 Macro Definition Documentation . . . 153
5.34 OSAPI Heap . . . 155
5.34.1 Detailed Description . . . 155
5.34.2 Macro Definition Documentation . . . 155
5.34.3 Typedef Documentation . . . 155
5.34.4 Function Documentation . . . 155
5.35 OSAPI Mutex . . . 156
5.35.1 Detailed Description . . . 156
5.35.2 Typedef Documentation . . . 156
5.35.3 Function Documentation . . . 156
5.36 OSAPI Process . . . 158
5.36.1 Detailed Description . . . 158
5.36.2 Typedef Documentation . . . 158
5.36.3 Function Documentation . . . 158
5.37 OSAPI Semaphore . . . 159
5.37.1 Detailed Description . . . 159
5.37.2 Typedef Documentation . . . 159
5.37.3 Function Documentation . . . 159
5.38 OSAPI_SharedMemoryMonitor . . . 161

5.38.1 Detailed Description . . . 161
5.38.2 Function Documentation . . . 161
5.39 OSAPI_SharedMemorySegment . . . 164
5.39.1 Detailed Description . . . 165
5.39.2 Enumeration Type Documentation . . . 165
5.39.3 Function Documentation . . . 165
5.40 OSAPI Memory API . . . 169
5.40.1 Detailed Description . . . 169
5.40.2 Function Documentation . . . 169
5.41 OSAPI String API . . . 171
5.41.1 Detailed Description . . . 171
5.41.2 Function Documentation . . . 171
5.42 OSAPI System API . . . 173
5.42.1 Detailed Description . . . 174
5.42.2 Macro Definition Documentation . . . 174
5.42.3 Typedef Documentation . . . 174
5.42.4 Function Documentation . . . 176
5.43 OSAPI Thread API . . . 179
5.43.1 Detailed Description . . . 179
5.43.2 Macro Definition Documentation . . . 179
5.43.3 Typedef Documentation . . . 180
5.44 OSAPI Types . . . 181
5.44.1 Detailed Description . . . 181
5.44.2 Macro Definition Documentation . . . 181
5.44.3 Typedef Documentation . . . 182
5.45 DB . . . 183
5.45.1 Detailed Description . . . 184
5.45.2 Macro Definition Documentation . . . 184
5.46 CDR . . . 187
5.46.1 Detailed Description . . . 187
5.46.2 Macro Definition Documentation . . . 187
5.47 RT . . . 188
5.47.1 Detailed Description . . . 188
5.47.2 Macro Definition Documentation . . . 188
5.48 NETIO Packet API . . . 190
5.48.1 Detailed Description . . . 190
5.48.2 Typedef Documentation . . . 190
5.48.3 Function Documentation . . . 190
5.49 NETIO Interface API . . . 192
5.49.1 Detailed Description . . . 192
5.49.2 Macro Definition Documentation . . . 192
5.49.3 Typedef Documentation . . . 193
5.49.4 Variable Documentation . . . 194
5.50 NETIO . . . 196
5.50.1 Detailed Description . . . 199
5.50.2 Macro Definition Documentation . . . 199
5.51 RTPS . . . 206
5.51.1 Detailed Description . . . 211
5.51.2 Macro Definition Documentation . . . 211
5.52 DDS-Specific Primitive Types . . . 222
5.52.1 Detailed Description . . . 222
5.52.2 Macro Definition Documentation . . . 223
5.52.3 Typedef Documentation . . . 223
5.53 Participant Built-in Topic . . . 225
5.53.1 Detailed Description . . . 225
5.53.2 Macro Definition Documentation . . . 225
5.53.3 Function Documentation . . . 225
5.54 Publication Built-in Topic . . . 227

5.54.1 Detailed Description . . . 227
5.54.2 Macro Definition Documentation . . . 227
5.54.3 Function Documentation . . . 227
5.55 Subscription Built-in Topic . . . 229
5.55.1 Detailed Description . . . 229
5.55.2 Macro Definition Documentation . . . 229
5.55.3 Function Documentation . . . 229
5.56 DISCOVERY . . . 231
5.56.1 Detailed Description . . . 231
5.56.2 Variable Documentation . . . 231
5.57 Time Support . . . 232
5.57.1 Detailed Description . . . 232
5.57.2 Macro Definition Documentation . . . 232
5.57.3 Variable Documentation . . . 232
5.58 GUID Support . . . 234
5.58.1 Detailed Description . . . 234
5.58.2 Variable Documentation . . . 234
5.59 Sequence Number Support . . . 235
5.59.1 Detailed Description . . . 235
5.60 Return Codes . . . 236
5.60.1 Detailed Description . . . 236
5.60.2 Standard Return Codes . . . 236
5.60.3 Enumeration Type Documentation . . . 236
5.61 Status Kinds . . . 238
5.61.1 Detailed Description . . . 238
5.61.2 Changes in Status . . . 239
5.61.3 Macro Definition Documentation . . . 242
5.61.4 Typedef Documentation . . . 242
5.61.5 Enumeration Type Documentation . . . 242
5.62 QoS Policies . . . 246
5.62.1 Detailed Description . . . 247
5.62.2 Specifying QoS on entities . . . 248
5.62.3 QoS compatibility . . . 248
5.62.4 Enumeration Type Documentation . . . 249
5.63 DEADLINE . . . 250
5.63.1 Detailed Description . . . 250
5.63.2 Variable Documentation . . . 250
5.64 OWNERSHIP . . . 251
5.64.1 Detailed Description . . . 251
5.64.2 Enumeration Type Documentation . . . 251
5.64.3 Variable Documentation . . . 251
5.65 OWNERSHIP_STRENGTH . . . 252
5.65.1 Detailed Description . . . 252
5.65.2 Variable Documentation . . . 252
5.66 LIVELINESS . . . 253
5.66.1 Detailed Description . . . 253
5.66.2 Enumeration Type Documentation . . . 253
5.66.3 Variable Documentation . . . 254
5.67 RELIABILITY . . . 255
5.67.1 Detailed Description . . . 255
5.67.2 Enumeration Type Documentation . . . 255
5.67.3 Variable Documentation . . . 255
5.68 HISTORY . . . 256
5.68.1 Detailed Description . . . 256
5.68.2 Enumeration Type Documentation . . . 256
5.68.3 Variable Documentation . . . 256
5.69 DURABILITY . . . 257
5.69.1 Detailed Description . . . 257

5.69.2 Enumeration Type Documentation . . . 257
5.69.3 Variable Documentation . . . 257
5.70 RESOURCE_LIMITS . . . 258
5.70.1 Detailed Description . . . 258
5.70.2 Variable Documentation . . . 258
5.71 PRESENTATION . . . 259
5.71.1 Detailed Description . . . 259
5.71.2 Enumeration Type Documentation . . . 259
5.72 DESTINATION_ORDER . . . 260
5.72.1 Detailed Description . . . 260
5.72.2 Enumeration Type Documentation . . . 260
5.73 DataReader Resource Limits . . . 261
5.73.1 Detailed Description . . . 261
5.73.2 Enumeration Type Documentation . . . 261
5.73.3 Variable Documentation . . . 261
5.74 DataWriter Resource Limits . . . 262
5.74.1 Detailed Description . . . 262
5.74.2 Variable Documentation . . . 262
5.75 ENTITY_FACTORY . . . 263
5.75.1 Detailed Description . . . 263
5.75.2 Variable Documentation . . . 263
5.76 Extended Qos Support . . . 264
5.76.1 Detailed Description . . . 264
5.77 SYSTEM_RESOURCE_LIMITS . . . 265
5.77.1 Detailed Description . . . 265
5.77.2 Variable Documentation . . . 265
5.78 WIRE_PROTOCOL . . . 266
5.78.1 Detailed Description . . . 266
5.78.2 Macro Definition Documentation . . . 267
5.78.3 Typedef Documentation . . . 267
5.78.4 Variable Documentation . . . 268
5.79 DATA_READER_PROTOCOL . . . 270
5.79.1 Detailed Description . . . 270
5.79.2 Variable Documentation . . . 270
5.80 DATA_WRITER_PROTOCOL . . . 271
5.80.1 Detailed Description . . . 271
5.80.2 Variable Documentation . . . 271
5.81 TRANSPORT . . . 272
5.81.1 Detailed Description . . . 272
5.81.2 Variable Documentation . . . 272
5.82 DOMAIN_PARTICIPANT_RESOURCE_LIMITS . . . 273
5.82.1 Detailed Description . . . 273
5.82.2 Variable Documentation . . . 273
5.83 Conditions and WaitSets . . . 274
5.83.1 Detailed Description . . . 275
5.83.2 Typedef Documentation . . . 275
5.83.3 Usage . . . 275
5.83.4 Trigger State of a ::DDS_StatusCondition . . . 276
5.83.5 Trigger State of a ::DDS_GuardCondition . . . 276
5.83.6 Function Documentation . . . 276
5.84 Entity Support . . . 281
5.84.1 Detailed Description . . . 281
5.84.2 Macro Definition Documentation . . . 281
5.84.3 Typedef Documentation . . . 282
5.84.4 Abstract operations . . . 282
5.84.5 Function Documentation . . . 283
5.85 ENTITY_NAME . . . 286
5.85.1 Detailed Description . . . 286

5.85.2 Macro Definition Documentation . . . 286
5.85.3 Function Documentation . . . 286
5.85.4 Variable Documentation . . . 287
5.86 DDS_C . . . 288
5.86.1 Detailed Description . . . 300
5.86.2 Macro Definition Documentation . . . 300
5.87 Sequence Support . . . 323
5.87.1 Detailed Description . . . 323
5.87.2 Macro Definition Documentation . . . 323
5.87.3 Function Documentation . . . 324
5.88 String Support . . . 329
5.88.1 Detailed Description . . . 329
5.88.2 Function Documentation . . . 330
5.89 RH . . . 333
5.89.1 Detailed Description . . . 334
5.89.2 Macro Definition Documentation . . . 334
5.90 WH . . . 336
5.90.1 Detailed Description . . . 336
5.90.2 Macro Definition Documentation . . . 336
5.91 DPSE . . . 338
5.91.1 Detailed Description . . . 340
5.91.2 Macro Definition Documentation . . . 340
5.92 Notification Interface . . . 346
5.92.1 Detailed Description . . . 346
5.92.2 Macro Definition Documentation . . . 346
5.92.3 Function Documentation . . . 346
5.93 User notification mechanism interface . . . 348
5.93.1 Detailed Description . . . 348
5.93.2 Typedef Documentation . . . 348
5.94 WHSQ . . . 352
5.94.1 Detailed Description . . . 352
5.94.2 Macro Definition Documentation . . . 352
5.95 Log Codes . . . 354
5.95.1 Detailed Description . . . 354
5.96 NETIO Address API . . . 355
5.96.1 Detailed Description . . . 355
5.96.2 Macro Definition Documentation . . . 355
5.96.3 Function Documentation . . . 356

6 Data Structure Documentation 358
6.1 DDS_BuiltinTopicKey_t Struct Reference . . . 358
6.1.1 Detailed Description . . . 358
6.1.2 Field Documentation . . . 358
6.2 DDS_ChecksumProperty Struct Reference . . . 358
6.2.1 Detailed Description . . . 359
6.2.2 Field Documentation . . . 359
6.3 DDS_ChecksumProperty_t Struct Reference . . . 359
6.3.1 Detailed Description . . . 359
6.3.2 Field Documentation . . . 359
6.4 DDS_ConditionSeq Struct Reference . . . 359
6.4.1 Detailed Description . . . 360
6.5 DDS_DataReaderInstanceReplacedStatus Struct Reference . . . 360
6.5.1 Detailed Description . . . 360
6.5.2 Field Documentation . . . 360
6.6 DDS_DataReaderListener Struct Reference . . . 361
6.6.1 Detailed Description . . . 361
6.6.2 Field Documentation . . . 362
6.7 DDS_DataReaderProtocolQosPolicy Struct Reference . . . 363

6.7.1 Detailed Description . . . 363
6.7.2 Field Documentation . . . 364
6.8 DDS_DataReaderQos Struct Reference . . . 364
6.8.1 Detailed Description . . . 365
6.8.2 Field Documentation . . . 365
6.9 DDS_DataReaderResourceLimitsQosPolicy Struct Reference . . . 366
6.9.1 Detailed Description . . . 366
6.9.2 Field Documentation . . . 366
6.10 DDS_DataWriterListener Struct Reference . . . 368
6.10.1 Detailed Description . . . 368
6.10.2 Field Documentation . . . 368
6.11 DDS_DataWriterProtocolQosPolicy Struct Reference . . . 370
6.11.1 Detailed Description . . . 370
6.11.2 Field Documentation . . . 370
6.12 DDS_DataWriterQos Struct Reference . . . 370
6.12.1 Detailed Description . . . 371
6.12.2 Field Documentation . . . 372
6.13 DDS_DataWriterResourceLimitsQosPolicy Struct Reference . . . 372
6.13.1 Detailed Description . . . 373
6.13.2 Field Documentation . . . 373
6.14 DDS_DeadlineQosPolicy Struct Reference . . . 373
6.14.1 Detailed Description . . . 374
6.14.2 Usage . . . 374
6.14.3 Compatibility . . . 374
6.14.4 Field Documentation . . . 374
6.15 DDS_DestinationOrderQosPolicy Struct Reference . . . 374
6.15.1 Detailed Description . . . 375
6.15.2 Field Documentation . . . 375
6.16 DDS_DiscoveryComponent Struct Reference . . . 375
6.16.1 Detailed Description . . . 375
6.16.2 Field Documentation . . . 375
6.17 DDS_DiscoveryQosPolicy Struct Reference . . . 376
6.17.1 Detailed Description . . . 376
6.17.2 Usage . . . 376
6.17.3 Field Documentation . . . 376
6.18 DDS_DomainParticipantFactoryQos Struct Reference . . . 377
6.18.1 Detailed Description . . . 377
6.18.2 Field Documentation . . . 378
6.19 DDS_DomainParticipantListener Struct Reference . . . 378
6.19.1 Detailed Description . . . 378
6.19.2 Field Documentation . . . 379
6.20 DDS_DomainParticipantQos Struct Reference . . . 379
6.20.1 Detailed Description . . . 379
6.20.2 Field Documentation . . . 379
6.21 DDS_DomainParticipantResourceLimitsQosPolicy Struct Reference . . . 380
6.21.1 Detailed Description . . . 381
6.21.2 Field Documentation . . . 381
6.22 DDS_DurabilityQosPolicy Struct Reference . . . 384
6.22.1 Detailed Description . . . 384
6.22.2 Field Documentation . . . 384
6.23 DDS_Duration_t Struct Reference . . . 384
6.23.1 Detailed Description . . . 384
6.23.2 Field Documentation . . . 384
6.24 DDS_EntityFactoryQosPolicy Struct Reference . . . 385
6.24.1 Detailed Description . . . 385
6.24.2 Usage . . . 385
6.24.3 Field Documentation . . . 385
6.25 DDS_EntityNameQosPolicy Struct Reference . . . 385

6.25.1 Detailed Description . . . 386
6.25.2 Usage . . . 386
6.25.3 Field Documentation . . . 386
6.26 DDS_GUID_t Struct Reference . . . 386
6.26.1 Detailed Description . . . 386
6.26.2 Field Documentation . . . 386
6.27 DDS_HistoryQosPolicy Struct Reference . . . 387
6.27.1 Detailed Description . . . 387
6.27.2 Usage . . . 387
6.27.3 Consistency . . . 387
6.27.4 Field Documentation . . . 388
6.28 DDS_InconsistentTopicStatus Struct Reference . . . 388
6.28.1 Detailed Description . . . 388
6.29 DDS_Listener Struct Reference . . . 388
6.29.1 Detailed Description . . . 388
6.29.2 Access to Plain Communication Status . . . 389
6.29.3 Access to Read Communication Status . . . 391
6.29.4 Field Documentation . . . 392
6.30 DDS_LivelinessChangedStatus Struct Reference . . . 392
6.30.1 Detailed Description . . . 392
6.30.2 Field Documentation . . . 392
6.31 DDS_LivelinessLostStatus Struct Reference . . . 392
6.31.1 Detailed Description . . . 393
6.31.2 Field Documentation . . . 393
6.32 DDS_LivelinessQosPolicy Struct Reference . . . 393
6.32.1 Detailed Description . . . 393
6.32.2 Usage . . . 394
6.32.3 Compatibility . . . 395
6.32.4 Field Documentation . . . 395
6.33 DDS_Locator_t Struct Reference . . . 395
6.33.1 Detailed Description . . . 396
6.33.2 Field Documentation . . . 396
6.34 DDS_LocatorSeq Struct Reference . . . 396
6.34.1 Detailed Description . . . 396
6.35 DDS_OfferedDeadlineMissedStatus Struct Reference . . . 396
6.35.1 Detailed Description . . . 396
6.35.2 Field Documentation . . . 397
6.36 DDS_OfferedIncompatibleQosStatus Struct Reference . . . 397
6.36.1 Detailed Description . . . 397
6.36.2 Field Documentation . . . 398
6.37 DDS_OwnershipQosPolicy Struct Reference . . . 398
6.37.1 Detailed Description . . . 398
6.37.2 Usage . . . 399
6.37.3 Compatibility . . . 399
6.37.4 Field Documentation . . . 401
6.38 DDS_OwnershipStrengthQosPolicy Struct Reference . . . 401
6.38.1 Detailed Description . . . 401
6.38.2 Field Documentation . . . 402
6.39 DDS_ParticipantBuiltinTopicData Struct Reference . . . 402
6.39.1 Detailed Description . . . 402
6.39.2 Field Documentation . . . 403
6.40 DDS_PresentationQosPolicy Struct Reference . . . 403
6.40.1 Detailed Description . . . 404
6.40.2 Compatibility . . . 405
6.40.3 Field Documentation . . . 406
6.41 DDS_ProductVersion Struct Reference . . . 406
6.41.1 Detailed Description . . . 407
6.41.2 Field Documentation . . . 407

6.42 DDS_ProductVersion_t Struct Reference . . . 407
6.42.1 Detailed Description . . . 407
6.42.2 Field Documentation . . . 407
6.43 DDS_ProtocolVersion Struct Reference . . . 408
6.43.1 Detailed Description . . . 408
6.43.2 Field Documentation . . . 408
6.44 DDS_PublicationBuiltinTopicData Struct Reference . . . 408
6.44.1 Detailed Description . . . 409
6.44.2 Field Documentation . . . 409
6.45 DDS_PublicationMatchedStatus Struct Reference . . . 410
6.45.1 Detailed Description . . . 410
6.45.2 Field Documentation . . . 411
6.46 DDS_PublisherListener Struct Reference . . . 411
6.46.1 Detailed Description . . . 411
6.46.2 Field Documentation . . . 412
6.47 DDS_PublisherQos Struct Reference . . . 412
6.47.1 Detailed Description . . . 412
6.47.2 Field Documentation . . . 412
6.48 DDS_ReliabilityQosPolicy Struct Reference . . . 412
6.48.1 Detailed Description . . . 412
6.48.2 Usage . . . 413
6.48.3 Compatibility . . . 413
6.48.4 Field Documentation . . . 413
6.49 DDS_ReliableReaderActivityChangedStatus Struct Reference . . . 413
6.49.1 Detailed Description . . . 414
6.49.2 Field Documentation . . . 414
6.50 DDS_ReliableSampleUnacknowledgedStatus Struct Reference . . . 414
6.50.1 Detailed Description . . . 415
6.50.2 Field Documentation . . . 415
6.51 DDS_RequestedDeadlineMissedStatus Struct Reference . . . 415
6.51.1 Detailed Description . . . 415
6.51.2 Field Documentation . . . 415
6.52 DDS_RequestedIncompatibleQosStatus Struct Reference . . . 416
6.52.1 Detailed Description . . . 416
6.52.2 Field Documentation . . . 416
6.53 DDS_ResourceLimitsQosPolicy Struct Reference . . . 416
6.53.1 Detailed Description . . . 417
6.53.2 Usage . . . 417
6.53.3 Consistency . . . 417
6.53.4 Field Documentation . . . 418
6.54 DDS_RtpsReliableReaderProtocol_t Struct Reference . . . 420
6.54.1 Detailed Description . . . 420
6.54.2 Field Documentation . . . 420
6.55 DDS_RtpsReliableWriterProtocol_t Struct Reference . . . 420
6.55.1 Detailed Description . . . 421
6.55.2 Field Documentation . . . 421
6.56 DDS_RtpsWellKnownPorts_t Struct Reference . . . 422
6.56.1 Detailed Description . . . 422
6.56.2 Field Documentation . . . 423
6.57 DDS_SampleInfo Struct Reference . . . 425
6.57.1 Detailed Description . . . 425
6.57.2 Interpretation of the SampleInfo . . . 425
6.57.3 Field Documentation . . . 426
6.58 DDS_SampleInfoSeq Struct Reference . . . 427
6.58.1 Detailed Description . . . 427
6.59 DDS_SampleLostStatus Struct Reference . . . 428
6.59.1 Detailed Description . . . 428
6.59.2 Field Documentation . . . 428

6.60 DDS_SampleRejectedStatus Struct Reference . . . 428
6.60.1 Detailed Description . . . 429
6.60.2 Field Documentation . . . 429
6.61 DDS_SequenceNumber_t Struct Reference . . . 429
6.61.1 Detailed Description . . . 429
6.61.2 Field Documentation . . . 430
6.62 DDS_SubscriberListener Struct Reference . . . 430
6.62.1 Detailed Description . . . 430
6.62.2 Field Documentation . . . 430
6.63 DDS_SubscriberQos Struct Reference . . . 431
6.63.1 Detailed Description . . . 431
6.63.2 Field Documentation . . . 431
6.64 DDS_SubscriptionBuiltinTopicData Struct Reference . . . 431
6.64.1 Detailed Description . . . 432
6.64.2 Field Documentation . . . 432
6.65 DDS_SubscriptionMatchedStatus Struct Reference . . . 433
6.65.1 Detailed Description . . . 433
6.65.2 Field Documentation . . . 433
6.66 DDS_SystemResourceLimitsQosPolicy Struct Reference . . . 434
6.66.1 Detailed Description . . . 434
6.66.2 Field Documentation . . . 434
6.67 DDS_Time_t Struct Reference . . . 435
6.67.1 Detailed Description . . . 435
6.67.2 Field Documentation . . . 435
6.68 DDS_TopicListener Struct Reference . . . 435
6.68.1 Detailed Description . . . 436
6.68.2 Field Documentation . . . 436
6.69 DDS_TopicQos Struct Reference . . . 436
6.69.1 Detailed Description . . . 436
6.70 DDS_TransportQosPolicy Struct Reference . . . 436
6.70.1 Detailed Description . . . 437
6.70.2 Field Documentation . . . 437
6.71 DDS_UserTrafficQosPolicy Struct Reference . . . 437
6.71.1 Detailed Description . . . 437
6.71.2 Field Documentation . . . 437
6.72 DDS_VendorId Struct Reference . . . 438
6.72.1 Detailed Description . . . 438
6.72.2 Field Documentation . . . 438
6.73 DDS_VendorId_t Struct Reference . . . 438
6.73.1 Detailed Description . . . 438
6.73.2 Field Documentation . . . 438
6.74 DDS_WireProtocolQosPolicy Struct Reference . . . 438
6.74.1 Detailed Description . . . 439
6.74.2 Usage . . . 439
6.74.3 Field Documentation . . . 439
6.75 DPSE_DiscoveryPluginProperty Struct Reference . . . 442
6.75.1 Detailed Description . . . 442
6.75.2 Field Documentation . . . 442
6.76 FooDataReader Struct Reference . . . 443
6.76.1 Detailed Description . . . 443
6.77 FooDataWriter Struct Reference . . . 444
6.77.1 Detailed Description . . . 444
6.78 FooSeq Struct Reference . . . 444
6.78.1 Detailed Description . . . 445
6.78.2 Field Documentation . . . 445
6.79 FooTypeSupport Struct Reference . . . 445
6.79.1 Detailed Description . . . 446
6.80 NDDS_Discovery_Property Struct Reference . . . 446

6.80.1 Detailed Description . . . 446
6.81 NDDS_Type_Plugin Struct Reference . . . 446
6.81.1 Detailed Description . . . 446
6.81.2 Field Documentation . . . 446
6.82 NDDS_Type_PluginVersion Struct Reference . . . 447
6.82.1 Detailed Description . . . 447
6.82.2 Field Documentation . . . 447
6.83 NETIO_Address Struct Reference . . . 448
6.83.1 Detailed Description . . . 448
6.83.2 Field Documentation . . . 448
6.84 NETIO_AddressSeq Struct Reference . . . 448
6.84.1 Detailed Description . . . 448
6.85 NETIO_AddressUInt32 Struct Reference . . . 448
6.85.1 Detailed Description . . . 448
6.85.2 Field Documentation . . . 449
6.86 NETIO_AddressValue Union Reference . . . 449
6.86.1 Detailed Description . . . 449
6.86.2 Field Documentation . . . 449
6.87 NETIO_DGRAM_InterfaceI Struct Reference . . . 449
6.87.1 Detailed Description . . . 450
6.87.2 Field Documentation . . . 450
6.88 NETIO_DGRAM_InterfaceMultiCastGroup Struct Reference . . . 451
6.88.1 Detailed Description . . . 451
6.88.2 Field Documentation . . . 451
6.89 NETIO_DGRAM_InterfaceRouteEntry Struct Reference . . . 451
6.89.1 Detailed Description . . . 452
6.89.2 Field Documentation . . . 452
6.90 NETIO_DGRAM_InterfaceTableEntry Struct Reference . . . 452
6.90.1 Detailed Description . . . 452
6.90.2 Field Documentation . . . 452
6.91 NETIO_DGRAM_InterfaceTableEntrySeq Struct Reference . . . 453
6.91.1 Detailed Description . . . 453
6.92 NETIO_Interface Struct Reference . . . 453
6.92.1 Detailed Description . . . 453
6.93 NETIO_InterfaceI Struct Reference . . . 453
6.93.1 Detailed Description . . . 453
6.94 NETIO_Netmask Struct Reference . . . 453
6.94.1 Detailed Description . . . 453
6.94.2 Field Documentation . . . 454
6.95 NETIO_NetmaskSeq Struct Reference . . . 454
6.95.1 Detailed Description . . . 454
6.96 NETIO_Packet Struct Reference . . . 454
6.96.1 Detailed Description . . . 454
6.97 OSAPI_LogProperty Struct Reference . . . 454
6.97.1 Detailed Description . . . 454
6.97.2 Field Documentation . . . 455
6.98 OSAPI_SharedMemorySegmentHandle Struct Reference . . . 455
6.98.1 Detailed Description . . . 455
6.98.2 Field Documentation . . . 455
6.99 OSAPI_SharedMemorySegmentHeader Struct Reference . . . 455
6.99.1 Detailed Description . . . 455
6.99.2 Field Documentation . . . 456
6.100 OSAPI_SystemI Struct Reference . . . 456
6.100.1 Detailed Description . . . 456
6.100.2 Field Documentation . . . 456
6.101 OSAPI_SystemProperty Struct Reference . . . 457
6.101.1 Detailed Description . . . 457
6.101.2 Field Documentation . . . 457

6.102OSAPI_SystemTime Struct Reference . . . 457
6.102.1 Detailed Description . . . 457
6.102.2 Field Documentation . . . 457
6.103OSAPI_ThreadProperty Struct Reference . . . 458
6.103.1 Detailed Description . . . 458
6.103.2 Field Documentation . . . 458
6.104OSAPI_TimerProperty Struct Reference . . . 458
6.104.1 Detailed Description . . . 458
6.105RTI_ManagementQosPolicy Struct Reference . . . 458
6.105.1 Detailed Description . . . 459
6.105.2 Field Documentation . . . 459
6.106RTPS_Checksum Union Reference . . . 459
6.106.1 Detailed Description . . . 459
6.107RTPS_ChecksumClass Struct Reference . . . 459
6.107.1 Detailed Description . . . 460
6.108RTPS_ChecksumProperty Struct Reference . . . 460
6.108.1 Detailed Description . . . 460
6.109RTPS_InterfaceFactoryProperty Struct Reference . . . 460
6.109.1 Detailed Description . . . 460
6.110UDP_InterfaceFactoryProperty Struct Reference . . . 460
6.110.1 Detailed Description . . . 461
6.110.2 Field Documentation . . . 461
6.111UDP_InterfaceTableEntry Struct Reference . . . 462
6.111.1 Detailed Description . . . 462
6.111.2 Field Documentation . . . 462
6.112UDP_InterfaceTableEntrySeq Struct Reference . . . 463
6.112.1 Detailed Description . . . 463
6.113ZCOPY_NotifInterfaceFactoryProperty Struct Reference . . . 463
6.113.1 Detailed Description . . . 463
6.113.2 Field Documentation . . . 463
6.114ZCOPY_NotifUserInterfaceI Struct Reference . . . 463
6.114.1 Detailed Description . . . 464
6.114.2 Field Documentation . . . 464

7 File Documentation **465**
7.1 cdr_log.h File Reference . . . 465
7.1.1 Detailed Description . . . 465
7.2 db_dll.h File Reference . . . 465
7.2.1 Detailed Description . . . 465
7.3 db_log.h File Reference . . . 465
7.3.1 Detailed Description . . . 466
7.4 dds_c_common.h File Reference . . . 467
7.4.1 Detailed Description . . . 467
7.5 dds_c_common_impl.h File Reference . . . 468
7.5.1 Detailed Description . . . 468
7.6 dds_c_config.h File Reference . . . 468
7.6.1 Detailed Description . . . 468
7.7 dds_c_discovery.h File Reference . . . 468
7.7.1 Detailed Description . . . 469
7.8 dds_c_discovery_plugin.h File Reference . . . 469
7.8.1 Detailed Description . . . 469
7.9 dds_c_dll.h File Reference . . . 469
7.9.1 Detailed Description . . . 469
7.10 dds_c_domain.h File Reference . . . 469
7.10.1 Detailed Description . . . 472
7.10.2 Macro Definition Documentation . . . 472
7.11 dds_c_infrastructure.h File Reference . . . 472
7.11.1 Detailed Description . . . 479

7.11.2 Macro Definition Documentation . . . 479
7.11.3 Enumeration Type Documentation . . . 480
7.11.4 Variable Documentation . . . 480
7.12 dds_c_log.h File Reference . . . 480
7.12.1 Detailed Description . . . 493
7.13 dds_c_publication.h File Reference . . . 493
7.13.1 Detailed Description . . . 496
7.14 dds_c_rh_plugin.h File Reference . . . 496
7.14.1 Detailed Description . . . 496
7.14.2 Variable Documentation . . . 496
7.15 dds_c_sequence.h File Reference . . . 496
7.15.1 Detailed Description . . . 497
7.16 dds_c_string.h File Reference . . . 497
7.16.1 Detailed Description . . . 497
7.17 dds_c_subscription.h File Reference . . . 497
7.17.1 Detailed Description . . . 501
7.18 dds_c_topic.h File Reference . . . 502
7.18.1 Detailed Description . . . 503
7.19 dds_c_type.h File Reference . . . 503
7.19.1 Detailed Description . . . 504
7.20 dds_c_wh_plugin.h File Reference . . . 504
7.20.1 Detailed Description . . . 504
7.20.2 Variable Documentation . . . 504
7.21 disc_dpse_dll.h File Reference . . . 504
7.21.1 Detailed Description . . . 505
7.22 disc_dpse_dpsediscovery.h File Reference . . . 505
7.22.1 Detailed Description . . . 505
7.23 disc_dpse_log.h File Reference . . . 505
7.23.1 Detailed Description . . . 509
7.23.2 Macro Definition Documentation . . . 509
7.24 netio_common.h File Reference . . . 510
7.24.1 Detailed Description . . . 510
7.25 netio_config.h File Reference . . . 510
7.25.1 Detailed Description . . . 510
7.26 netio_dll.h File Reference . . . 510
7.26.1 Detailed Description . . . 510
7.27 netio_interface.h File Reference . . . 510
7.27.1 Detailed Description . . . 512
7.27.2 Typedef Documentation . . . 512
7.27.3 Variable Documentation . . . 512
7.28 netio_log.h File Reference . . . 512
7.28.1 Detailed Description . . . 516
7.29 netio_rtps.h File Reference . . . 516
7.29.1 Detailed Description . . . 516
7.30 netio_zcopy.h File Reference . . . 516
7.30.1 Detailed Description . . . 517
7.31 netio_zcopy_data.h File Reference . . . 517
7.31.1 Detailed Description . . . 517
7.32 netio_zcopy_dll.h File Reference . . . 517
7.32.1 Detailed Description . . . 517
7.33 netio_zcopy_log.h File Reference . . . 517
7.33.1 Detailed Description . . . 517
7.34 netio_zcopy_notif_interface.h File Reference . . . 517
7.34.1 Detailed Description . . . 518
7.35 netio_zcopy_notif_mechanism.h File Reference . . . 518
7.35.1 Detailed Description . . . 519
7.36 netio_zcopy_publication.h File Reference . . . 519
7.36.1 Detailed Description . . . 519

7.37 netio_zcopy_subscription.h File Reference . . . 519
7.37.1 Detailed Description . . . 519
7.38 netio_zcopy_tdatareader_gen.h File Reference . . . 519
7.38.1 Detailed Description . . . 519
7.39 netio_zcopy_tdatawriter_gen.h File Reference . . . 519
7.39.1 Detailed Description . . . 519
7.40 netio_zcopy_whsq.h File Reference . . . 519
7.40.1 Detailed Description . . . 519
7.40.2 Function Documentation . . . 520
7.41 netio_zcopy_whsq_log.h File Reference . . . 520
7.41.1 Detailed Description . . . 520
7.41.2 Macro Definition Documentation . . . 520
7.42 osapi_cc.h File Reference . . . 520
7.42.1 Detailed Description . . . 521
7.42.2 Macro Definition Documentation . . . 521
7.43 osapi_cc_gcc.h File Reference . . . 521
7.43.1 Detailed Description . . . 521
7.44 osapi_cc_qcc.h File Reference . . . 521
7.44.1 Detailed Description . . . 521
7.45 osapi_cc_stdc.h File Reference . . . 521
7.45.1 Detailed Description . . . 521
7.46 osapi_config.h File Reference . . . 521
7.46.1 Detailed Description . . . 521
7.47 osapi_heap_impl.h File Reference . . . 522
7.47.1 Detailed Description . . . 522
7.48 osapi_log_impl.h File Reference . . . 522
7.48.1 Detailed Description . . . 522
7.49 osapi_os.h File Reference . . . 522
7.49.1 Detailed Description . . . 522
7.50 osapi_os_def.h File Reference . . . 522
7.50.1 Detailed Description . . . 522
7.51 osapi_os_posix.h File Reference . . . 522
7.51.1 Detailed Description . . . 522
7.52 osapi_process.h File Reference . . . 522
7.52.1 Detailed Description . . . 523
7.53 osapi_system.h File Reference . . . 523
7.53.1 Detailed Description . . . 524
7.54 osapi_time.h File Reference . . . 524
7.54.1 Detailed Description . . . 524
7.54.2 Typedef Documentation . . . 524
7.55 reda_dll.h File Reference . . . 524
7.55.1 Detailed Description . . . 524
7.56 reda_epoch.h File Reference . . . 524
7.56.1 Detailed Description . . . 525
7.57 reda_log.h File Reference . . . 525
7.57.1 Detailed Description . . . 525
7.58 reda_sequence_defn.h File Reference . . . 525
7.58.1 Detailed Description . . . 525
7.59 rh_sm_dll.h File Reference . . . 525
7.59.1 Detailed Description . . . 526
7.60 rh_sm_history.h File Reference . . . 526
7.60.1 Detailed Description . . . 526
7.61 rh_sm_log.h File Reference . . . 526
7.61.1 Detailed Description . . . 528
7.61.2 Macro Definition Documentation . . . 528
7.62 rt_dll.h File Reference . . . 528
7.62.1 Detailed Description . . . 529
7.63 rt_log.h File Reference . . . 529

7.63.1 Detailed Description . . . 529
7.64 rtps_config.h File Reference . . . 529
7.64.1 Detailed Description . . . 529
7.65 rtps_dll.h File Reference . . . 529
7.65.1 Detailed Description . . . 529
7.66 rtps_log.h File Reference . . . 529
7.66.1 Detailed Description . . . 535
7.66.2 Macro Definition Documentation . . . 535
7.67 rtps_rtps_impl.h File Reference . . . 535
7.67.1 Detailed Description . . . 535
7.68 wh_sm_dll.h File Reference . . . 536
7.68.1 Detailed Description . . . 536
7.69 wh_sm_history.h File Reference . . . 536
7.69.1 Detailed Description . . . 536
7.70 wh_sm_log.h File Reference . . . 536
7.70.1 Detailed Description . . . 537
7.70.2 Macro Definition Documentation . . . 537

1 Main Page

Real-Time Innovations, Inc.

RTI Connext Micro provides a small-footprint, modular messaging solution for resource-limited devices that have limited memory and CPU power, and may not even be running an operating system. It provides the communications services that developers need to distribute time-critical data. Additionally, RTI Connext Micro is designed as a certifiable component in high-assurance systems.

RTI Connext Micro benefits include:

- Accommodations for resource-constrained environments.
- Modular and user extensible architecture.
- Designed to be a certifiable component for safety-critical systems.
- Seamless interoperability with Connext DDS and Connext Messaging.

2 Module Index

2.1 Reference Manual

Here is a list of all modules:

Documentation Guide 113
DDS API 115
Domain Module 13
DomainParticipantFactory 15
DomainParticipant 22
Discovery 32
Participant Built-in Topic 225

Publication Built-in Topic 227
Subscription Built-in Topic 229
Topic 36
Topic 37
User Data Type Support 42
User-Defined Types 109
DDS-Specific Primitive Types 222
Publication Module 46
Publisher 47
DataWriter 52
Subscription Module 71
Subscriber 73
DataReader 80
Data Sample 101
Sample States 102
View States 104
Instance States 106
Infrastructure Module 108
Time Support 232
GUID Support 234
Sequence Number Support 235
Return Codes 236
Status Kinds 238
QoS Policies 246
DISCOVERY 231
DEADLINE 250
OWNERSHIP 251
OWNERSHIP_STRENGTH 252
LIVELINESS 253
RELIABILITY 255
HISTORY 256
DURABILITY 257

RESOURCE_LIMITS 258
PRESENTATION 259
DESTINATION_ORDER 260
ENTITY_FACTORY 263
Extended Qos Support 264
DataReader Resource Limits 261
DataWriter Resource Limits 262
SYSTEM_RESOURCE_LIMITS 265
WIRE_PROTOCOL 266
DATA_READER_PROTOCOL 270
DATA_WRITER_PROTOCOL 271
TRANSPORT 272
DOMAIN_PARTICIPANT_RESOURCE_LIMITS 273
ENTITY_NAME 286
Conditions and WaitSets 274
Entity Support 281
Sequence Support 323
String Support 329
DPSE Static Discovery API 121
CDR Stream API 125
RT Run-Time API 128
RHSM Reader History API 130
WHSM Writer History API 131
NETIO API 132
NETIO Packet API 190
NETIO Interface API 192
NETIO Address API 355
RTPS Transport API 133
NETIO DGRAM API 137
Zero Copy v2 API 145
Notification Interface 346
User notification mechanism interface 348

OS API 147
OSAPI Heap 155
OSAPI Mutex 156
OSAPI Process 158
OSAPI Semaphore 159
OSAPI_SharedMemoryMonitor 161
OSAPI_SharedMemorySegment 164
OSAPI Memory API 169
OSAPI String API 171
OSAPI System API 173
OSAPI Thread API 179
OSAPI Types 181
Log API 148
Log Codes 354
REDA 153
DB 183
CDR 187
RT 188
NETIO 196
RTPS 206
DDS_C 288
RH 333
WH 336
DPSE 338
WHSQ 352

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

DDS_BuiltinTopicKey_t
The key type of the built-in topic types 358

DDS_ChecksumProperty
<<*eXtension*>> Type to define the checksum properties of the participant RTI Connext Micro 358

DDS_ChecksumProperty_t
<<*eXtension*>> Type to define the checksum properties of the participant RTI Connext Micro 359

DDS_ConditionSeq
A sequence object containing references to one or more DDS_Condition objects 359

DDS_DataReaderInstanceReplacedStatus
<<*eXtension*>> <<*cert*>> DDS_INSTANCE_REPLACED_STATUS 360

DDS_DataReaderListener
<<*interface*>> <<*cert*>> DDS_Listener for reader status 361

DDS_DataReaderProtocolQosPolicy
<<*eXtension*>> Protocol that applies only to DDS_DataReader instances 363

DDS_DataReaderQos
<<*cert*>> QoS policies supported by a DDS_DataReader entity 364

DDS_DataReaderResourceLimitsQosPolicy
<<*cert*>> Resource limits that apply only to DDS_DataReader instances 366

DDS_DataWriterListener
<<*interface*>> <<*cert*>> DDS_Listener for writer status 368

DDS_DataWriterProtocolQosPolicy
<<*eXtension*>> <<*cert*>> Protocol that applies only to DDS_DataWriter instances 370

DDS_DataWriterQos
<<*cert*>> QoS policies supported by a DDS_DataWriter entity 370

DDS_DataWriterResourceLimitsQosPolicy
<<*cert*>> Resource limits that apply only to DDS_DataWriter instances 372

DDS_DeadlineQosPolicy
<<*cert*>> Expresses the maximum duration (deadline) within which an instance is expected to be updated 373

DDS_DestinationOrderQosPolicy
Destination order policies that affect the ordering of subsribed data 374

DDS_DiscoveryComponent
Specifies the discovery plugin the DDS_DomainParticipant should use 375

DDS_DiscoveryQosPolicy
<<*cert*>> Specifies the attributes required to discover participants in the domain 376

DDS_DomainParticipantFactoryQos
<<*cert*>> QoS policies supported by a DDS_DomainParticipantFactory 377

DDS_DomainParticipantListener
<<*interface*>> <<*cert*>> Listener for participant status 378

DDS_DomainParticipantQos
<<*cert*>> QoS policies supported by a DDS_DomainParticipant entity 379

DDS_DomainParticipantResourceLimitsQosPolicy
<<*cert*>> Resource limits that apply only to DDS_DomainParticipant instances 380

DDS_DurabilityQosPolicy
Durability properties that affect the life-cycle of published data 384

DDS_Duration_t
Type for *duration* representation 384

DDS_EntityFactoryQosPolicy
<<*cert*>> A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types 385

DDS_EntityNameQosPolicy
<<*cert*>> Policy used to name an entity 385

DDS_GUID_t
Type for *GUID* (Global Unique Identifier) representation 386

DDS_HistoryQosPolicy
<<*cert*>> Specifies the behavior of RTI Connext Micro in the case where the value of a sample changes (one or more times) before it can be successfully communicated to one or more existing subscribers 387

DDS_InconsistentTopicStatus
DDS_INCONSISTENT_TOPIC_STATUS 388

DDS_Listener
<<*interface*>> Abstract base class for all Listener interfaces 388

DDS_LivelinessChangedStatus
<<*cert*>> DDS_LIVELINESS_CHANGED_STATUS 392

DDS_LivelinessLostStatus
<<*cert*>> DDS_LIVELINESS_LOST_STATUS 392

DDS_LivelinessQosPolicy
<<*cert*>> Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive 393

DDS_Locator_t
<<*eXtension*>> <<*cert*>> Type used to represent the addressing information needed to send a message to an RTPS Endpoint using one of the supported transports 395

DDS_LocatorSeq
Declares IDL `sequence` < DDS_Locator_t > 396

DDS_OfferedDeadlineMissedStatus
<<*cert*>> DDS_OFFERED_DEADLINE_MISSED_STATUS 396

DDS_OfferedIncompatibleQosStatus
<<*cert*>> DDS_OFFERED_INCOMPATIBLE_QOS_STATUS 397

DDS_OwnershipQosPolicy
<<*cert*>> Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated 398

DDS_OwnershipStrengthQosPolicy
<<*cert*>> Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key) 401

DDS_ParticipantBuiltinTopicData
<<*cert*>> Object representing a remote DomainParticipant 402

DDS_PresentationQosPolicy
Specifies how the samples representing changes to data instances are presented to a subscribing application 403

DDS_ProductVersion
<<*eXtension*>> <<*cert*>> Type used to represent the current version of RTI Connext Micro 406

DDS_ProductVersion_t
<<*eXtension*>> <<*cert*>> Type used to represent the current version of RTI Connext Micro 407

DDS_ProtocolVersion
<<*eXtension*>> <<*cert*>> Type used to represent the version of the RTPS protocol 408

DDS_PublicationBuiltinTopicData
Object describing a remote Publication 408

DDS_PublicationMatchedStatus
<<*cert*>> DDS_PUBLICATION_MATCHED_STATUS 410

DDS_PublisherListener
<<*interface*>> <<*cert*>> DDS_Listener for DDS_Publisher status 411

DDS_PublisherQos
<<*cert*>> QoS policies supported by a DDS_Publisher entity 412

DDS_ReliabilityQosPolicy
<<*cert*>> Indicates the level of reliability offered/requested by RTI Connext Micro 412

DDS_ReliableReaderActivityChangedStatus
<<*eXtension*>> <<*cert*>> Describes the activity (i.e. are acknowledgements forthcoming) of reliable readers matched to a reliable writer 413

DDS_ReliableSampleUnacknowledgedStatus
<<*cert*>> Status about unacknowledged samples 414

DDS_RequestedDeadlineMissedStatus
<<*cert*>> DDS_REQUESTED_DEADLINE_MISSED_STATUS 415

DDS_RequestedIncompatibleQosStatus
<<*cert*>> DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS 416

DDS_ResourceLimitsQosPolicy
<<*cert*>> Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS 416

DDS_RtpsReliableReaderProtocol_t
Qos related to reliable reader protocol defined in RTPS 420

DDS_RtpsReliableWriterProtocol_t
<<*cert*>> QoS related to the reliable writer protocol defined in RTPS 420

DDS_RtpsWellKnownPorts_t
RTPS well-known port mapping configuration 422

DDS_SampleInfo
Information that accompanies each sample that is `read` or `taken` 425

DDS_SampleInfoSeq
Declares IDL `sequence` < DDS_SampleInfo > 427

DDS_SampleLostStatus
<<*cert*>> DDS_SAMPLE_LOST_STATUS 428

DDS_SampleRejectedStatus
DDS_SAMPLE_REJECTED_STATUS 428

DDS_SequenceNumber_t
Type for *sequence* number representation 429

DDS_SubscriberListener
<<*interface*>> <<*cert*>> DDS_Listener for status about a subscriber 430

DDS_SubscriberQos
<<*cert*>> QoS policies supported by a DDS_Subscriber entity 431

DDS_SubscriptionBuiltinTopicData
Entry created when a DDS_DataReader is discovered in association with its Subscriber 431

DDS_SubscriptionMatchedStatus
<<*cert*>> DDS_SUBSCRIPTION_MATCHED_STATUS 433

DDS_SystemResourceLimitsQosPolicy
<<*cert*>> Resource limits that apply only to DDS_DomainParticipantFactory 434

DDS_Time_t
Type for *time* representation 435

DDS_TopicListener
<<*interface*>> DDS_Listener for DDS_Topic entities 435

DDS_TopicQos
QoS policies supported by a DDS_Topic entity 436

DDS_TransportQosPolicy
<<*eXtension*>> <<*cert*>> Settings for available transports 436

DDS_UserTrafficQosPolicy
A QosPolicy to configure properties of network traffic caused by the exchange of user data 437

DDS_VendorId
<<*eXtension*>> <<*cert*>> Type used to represent the vendor of the service implementing the RTPS protocol 438

DDS_VendorId_t
<<*eXtension*>> <<*cert*>> Type used to represent the vendor of the service implementing the RTPS protocol 438

DDS_WireProtocolQosPolicy
<<*cert*>> Specifies the wire-protocol-related attributes for the DDS_DomainParticipant 438

DPSE_DiscoveryPluginProperty
<<*eXtension*>> <<*cert*>> Properties for the Dynamic Participant/Static Endpoint (DPSE) discovery plugin. This includes all discovery timing properties for participant discovery 442

FooDataReader
<<*interface*>> <<*generic*>> User data type-specific data reader 443

FooDataWriter
<<*interface*>> <<*generic*>> User data type specific data writer 444

FooSeq
<<*interface*>> <<*generic*>> <<*cert*>> A type-safe, ordered collection of elements. The type of these elements is referred to in this documentation as "Foo" 444

FooTypeSupport
<<*interface*>> <<*generic*>> User data type specific interface 445

NDDS_Discovery_Property
Specifies internal information used to store discovery information about a DDS_DomainParticipant in the registry 446

NDDS_Type_Plugin
<<*interface*>> Type plugin interface 446

NDDS_Type_PluginVersion
Data type used to keep track of versioning information about a NDDS_Type_Plugin 447

NETIO_Address 448

NETIO_AddressSeq
<<*eXtension*>> <<*cert*>> The NETIO_AddressSeq sequence type 448

NETIO_AddressUInt32
<<*eXtension*>> <<*cert*>> NETIO_AddressUInt32 448

NETIO_AddressValue
<<*eXtension*>> <<*cert*>> NETIO_AddresValue 449

NETIO_DGRAM_InterfaceI
<<*eXtension*>> <<*cert*>> 449

NETIO_DGRAM_InterfaceMultiCastGroup
<<*eXtension*>> <<*cert*>> 451

NETIO_DGRAM_InterfaceRouteEntry
<<*eXtension*>> <<*cert*>> 451

NETIO_DGRAM_InterfaceTableEntry
<<*eXtension*>> <<*cert*>> 452

NETIO_DGRAM_InterfaceTableEntrySeq
A sequence of NETIO_DGRAMInterfaceTableEntry elements 453

NETIO_Interface
Base-class definition for all NETIO interfaces 453

NETIO_InterfaceI 453

NETIO_Netmask
<<*eXtension*>> <<*cert*>> NETIO_Netmask 453

NETIO_NetmaskSeq
<<*eXtension*>> <<*cert*>> The NETIO_NetmaskSeq sequence type 454

NETIO_Packet
Implementation of the abstract NETIO_Packet type 454

OSAPI_LogProperty
Configuration of logging functionality 454

OSAPI_SharedMemorySegmentHandle 455

OSAPI_SharedMemorySegmentHeader 455

OSAPI_SystemI
<<*cert*>> The System API 456

OSAPI_SystemProperty
<<*cert*>> The system properties 457

OSAPI_SystemTime
System time 457

OSAPI_ThreadProperty
The properties for thread creation 458

OSAPI_TimerProperty
Timer properies 458

RTI_ManagementQosPolicy
<<*cert*>> Policy used determine how DDS_DataReader, DDS_DataWriter, DDS_Publisher, and DDS_Subscriber is managed 458

RTPS_Checksum
<<*eXtension*>> <<*cert*>> RTPS checksum type 459

RTPS_ChecksumClass
<<*eXtension*>> <<*cert*>> Definition of a checksum function 459

RTPS_ChecksumProperty
<<*eXtension*>> <<*cert*>> Checksum properties for the RTPS plugin 460

RTPS_InterfaceFactoryProperty
<<*eXtension*>> <<*cert*>> Properties for the RTPS Transport 460

UDP_InterfaceFactoryProperty
<<*eXtension*>> <<*cert*>> Properties for the UDP Transport 460

UDP_InterfaceTableEntry
Generic structure to describe a network interface 462

UDP_InterfaceTableEntrySeq
<<*eXtension*>> Declares IDL sequence<UDP_InterfaceTableEntry> 463

ZCOPY_NotifInterfaceFactoryProperty
Notification interface factory specific property 463

ZCOPY_NotifUserInterfaceI
Notification mechanism user interface 463

4 File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

cdr_log.h 465

db_dll.h
External linkage definitions for DB API 465

db_log.h
DB module log codes 465

dds_c_common.h
DDS_C common definitions 467

dds_c_common_impl.h
DDS_C common definitions 468

dds_c_config.h
DDS C configuration file 468

dds_c_discovery.h
DDS discovery data definitions 468

dds_c_discovery_plugin.h
DDS discovery plugin interface 469

dds_c_dll.h
DDS_C external linkage definitions 469

dds_c_domain.h
DDS Domain Module definitions 469

dds_c_infrastructure.h
DDS Domain Module definitions 472

dds_c_log.h
DDS_C module log codes 480

dds_c_publication.h
DDS publication module 493

dds_c_rh_plugin.h
DDS reader history interface 496

dds_c_sequence.h
DDS sequence definitions 496

dds_c_string.h
DDS string definitions 497

dds_c_subscription.h
DDS subscription module 497

dds_c_topic.h
DDS topic module 502

dds_c_type.h
DDS type module 503

dds_c_wh_plugin.h
DDS writer history interface 504

disc_dpse_dll.h
DPSE DLL external linkage 504

disc_dpse_dpsediscovery.h
DPSE API Dynamic Participant/Static Endpoint Discovery Plugin 505

disc_dpse_log.h 505

netio_common.h 510

netio_config.h 510

netio_dll.h 510

netio_interface.h
Generic NETIO_Interface functions 510

netio_log.h
NETIO module log codes 512

netio_rtps.h
RTPS API 516

netio_zcopy.h
Zero Copy transport API 516

netio_zcopy_data.h
ZeroCopy extensions for DDS typed interface 517

netio_zcopy_dll.h
ZeroCopy DLL external linkage 517

netio_zcopy_log.h
ZeroCopy module log codes 517

netio_zcopy_notif_interface.h 517

netio_zcopy_notif_mechanism.h 518

netio_zcopy_publication.h
ZeroCopy extensions for DDS publication module 519

netio_zcopy_subscription.h
ZeroCopy extensions for DDS subscription module 519

netio_zcopy_tdatareader_gen.h
ZeroCopy extensions for DDS DataReader 519

netio_zcopy_tdatawriter_gen.h
ZeroCopy extensions for DDS DataWriter 519

netio_zcopy_whsq.h
Shared Queue Writer History includes 519

netio_zcopy_whsq_log.h
WHSQ module log codes 520

osapi_cc.h
OS API Configuration 520

osapi_cc_gcc.h
OS API Configuration 521

osapi_cc_qcc.h
OS API Configuration 521

osapi_cc_stdc.h
OS API Configuration 521

osapi_config.h
OS API Configuration 521

osapi_heap_impl.h
Macro implementing heap interface 522

osapi_log_impl.h
Implementation of log functionality 522

osapi_os.h
OS API Configuration 522

osapi_os_def.h
OS API Configuration, pre-split 522

osapi_os_posix.h
OS API Configuration 522

osapi_process.h
Process interface definition 522

osapi_system.h
System API definition 523

osapi_time.h
Time API definition 524

reda_dll.h 524

reda_epoch.h 524

reda_log.h
REDA module log codes 525

reda_sequence_defn.h 525

rh_sm_dll.h
ReaderHistory DLL external linkage 525

rh_sm_history.h
Reader History API 526

rh_sm_log.h
RH module log codes 526

rt_dll.h 528

rt_log.h 529

rtps_config.h
RTPS Configuration 529

rtps_dll.h
RTPS Configuration 529

rtps_log.h
RTPS module log codes 529

rtps_rtps_impl.h
Implementation of RTPS interface functions and types 535

wh_sm_dll.h
WriterHistory DLL external linkage 536

wh_sm_history.h
Writer History API 536

wh_sm_log.h
WH module log codes 536

5 Module Documentation

5.1 Domain Module

Contains the DDS_DomainParticipant class that acts as an entrypoint of RTI Connext Micro and acts as a factory for many of the classes. The DDS_DomainParticipant also acts as a container for the other objects that make up RTI Connext Micro. Testing RTI Connext Micro command.

Modules

- DomainParticipantFactory

  DDS_DomainParticipantFactory entity and associated elements
- DomainParticipant

  DDS_DomainParticipant entity and associated elements
- Discovery

  Descriptions of remote entities used for discovery.

Typedefs

- typedef DDS_DOMAINID_TYPE_NATIVE DDS_DomainId_t

  <<cert>> An integer that indicates in which domain a DDS_DomainParticipant communicates.

5.1.1 Detailed Description

Contains the DDS_DomainParticipant class that acts as an entrypoint of RTI Connext Micro and acts as a factory for many of the classes. The DDS_DomainParticipant also acts as a container for the other objects that make up RTI Connext Micro. Testing RTI Connext Micro command. Defines the DDS domain package.

This image describes the DDS Domain Package, as described by the DDS standard. Note that not all functions and objects shown in this diagram are supported by RTI Connext Micro. Please read the information on the individual types to determine whether a function is supported.

5.1.2 Typedef Documentation

5.1.2.1 typedef DDS_DOMAINID_TYPE_NATIVE DDS_DomainId_t

<<cert>> An integer that indicates in which domain a DDS_DomainParticipant communicates.

Participants with the same DDS_DomainId_t are said to be in the same domain, and can thus communicate with one another.

The lower limit for a domain ID is 0. The upper limit for a domain ID is determined by the guidelines of the RTPS specification of the well-known ports on which a Participant will listen for discovery metatraffic from other Participants.

5.2 DomainParticipantFactory

DDS_DomainParticipantFactory entity and associated elements

Data Structures

- struct DDS_DomainParticipantFactoryQos

  <<cert>> QoS policies supported by a DDS_DomainParticipantFactory.

Macros

- #define DDS_DomainParticipantFactoryQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_TheParticipantFactory DDS_DomainParticipantFactory_get_instance()

  <<cert>> Alias for singleton participant factory.

Typedefs

- typedef struct DDS_DomainParticipantFactoryImpl DDS_DomainParticipantFactory

  <<singleton>> <<interface>> <<cert>> Allows creation and destruction of DDS_DomainParticipant objects.

Functions

- DDS_ReturnCode_t DDS_DomainParticipantFactoryQos_initialize (struct DDS_DomainParticipantFactoryQos *self)

  <<cert>> Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_DomainParticipantFactoryQos_copy (struct DDS_DomainParticipantFactoryQos *self, const struct DDS_DomainParticipantFactoryQos *source)

  <<cert>> Copy the contents of the given QoS into this QoS. The destination structure must be preallocated and initialized.
- DDS_Boolean DDS_DomainParticipantFactoryQos_is_equal (const struct DDS_DomainParticipantFactoryQos *left, const struct DDS_DomainParticipantFactoryQos *right)

  <<cert>> Compare two DDS_DomainParticipantFactoryQos policies for equality.
- DDS_DomainParticipantFactory * DDS_DomainParticipantFactory_get_instance (void)

  <<cert>> Gets the singleton instance of this class.
- DDS_DomainParticipant * DDS_DomainParticipantFactory_create_participant (DDS_DomainParticipantFactory *self, DDS_DomainId_t domainId, const struct DDS_DomainParticipantQos *qos, const struct DDS_DomainParticipantListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a new DDS_DomainParticipant object.
- DDS_DomainParticipant * DDS_DomainParticipantFactory_lookup_participant (DDS_DomainParticipantFactory *self, DDS_DomainId_t domainId)

  <<cert>> Locates an existing DDS_DomainParticipant.
- DDS_ReturnCode_t DDS_DomainParticipantFactory_get_qos (DDS_DomainParticipantFactory *self, struct DDS_DomainParticipantFactoryQos *qos)

  <<cert>> Gets the value for participant factory QoS.
- DDS_ReturnCode_t DDS_DomainParticipantFactory_set_qos (DDS_DomainParticipantFactory *self, const struct DDS_DomainParticipantFactoryQos *qos)

  <<cert>> Sets the value for a participant factory QoS.
- RT_Registry_T * DDS_DomainParticipantFactory_get_registry (DDS_DomainParticipantFactory *self)

  <<cert>> Retrieves the singleton registry instance associated with the factory.

Variables

- struct DDS_DomainParticipantFactoryQos DDS_PARTICIPANT_FACTORY_QOS_DEFAULT

  <<cert>> Special value for creating domain participant with default QoS.
- struct DDS_DomainParticipantQos DDS_PARTICIPANT_QOS_DEFAULT

  <<cert>> Special value for creating domain participant with default QoS.

5.2.1 Detailed Description

DDS_DomainParticipantFactory entity and associated elements

5.2.2 Macro Definition Documentation

5.2.2.1 #define DDS_DomainParticipantFactoryQos_INITIALIZER

<<cert>> Initializer for new QoS instances.

DDS_DomainParticipantFactoryQos instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a new QoS structure is to initialize it on the stack at the time of its creation.

5.2.2.2 #define DDS_TheParticipantFactory DDS_DomainParticipantFactory_get_instance()

<<cert>> Alias for singleton participant factory.

5.2.3 Typedef Documentation

5.2.3.1 typedef struct DDS_DomainParticipantFactoryImpl DDS_DomainParticipantFactory

<<singleton>> <<interface>> <<cert>> Allows creation and destruction of DDS_DomainParticipant objects.

The sole purpose of this class is to allow the creation and destruction of DDS_DomainParticipant objects. This class itself is a *<<singleton>>*, and accessed via the get_instance() function, and destroyed with finalize_instance() function.

A single application can participate in multiple domains by instantiating multiple DDS_DomainParticipant objects.

An application may even instantiate multiple participants in the same domain. Participants in the same domain exchange data in the same way regardless of whether they are in the same application or different applications or on the same node or different nodes; their location is transparent.

There are two important caveats:

- With multiple participants in the same domain on the same node (in the same application or different applications), each participant must be assigned its own receive ports. The receive ports are calculated based on the domain ID and the participant index set in the DDS_WireProtocolQosPolicy. The default index (-1) causes the participant to automatically search for the next available index, starting at 0, when it is created. If an index larger or equal to 0 is assigned then only that value will be used. If another participant with the same index already exists the participant creation will fail.

  NOTE: An exception to this rule is multicast ports. Multicast ports are shared between between participants in the same domain on the same host). Thus, the index value is ignored.
- You cannot mix entities from different participants. For example, you cannot delete a topic on a different participant than you created it from, and you cannot ask a subscriber to create a reader for a topic created from a participant different than the subscriber's own participant. (Note that it is permissable for an application built on top of RTI Connext Micro to know about entities from different participants. For example, an application

could keep references to a reader from one domain and a writer from another and then bridge the domains by writing the data received in the reader callback.)

See also

DDS_DomainParticipant

5.2.4 Function Documentation

5.2.4.1 DDS_ReturnCode_t DDS_DomainParticipantFactoryQos_initialize (struct DDS_DomainParticipantFactoryQos * *self*)

<<*cert*>> Initializer for new QoS instances.

Parameters

self	<<*in*>> Cannot be NULL.

API Restriction:

This function may be called before DDS_DomainParticipantFactory_get_instance.

5.2.4.2 DDS_ReturnCode_t DDS_DomainParticipantFactoryQos_copy (struct DDS_DomainParticipantFactoryQos * *self*, const struct DDS_DomainParticipantFactoryQos * *source*)

<<*cert*>> Copy the contents of the given QoS into this QoS. The destination structure must be preallocated and initialized.

DDS_DomainParticipantFactoryQos instances can use dynamic memory because of the sequences contained in some QoS policies. A shallow copy by assignment is therefore unsafe. This function performs a deep-copy, allocating memory if necessary.

Parameters

self	<<*in*>> Cannot be NULL.
source	<<*in*>>. QoS to be copied from.

Returns

One of the Standard Return Codes

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

See also

DDS_DomainParticipantFactoryQos_INITIALIZER

5.2.4.3 DDS_Boolean DDS_DomainParticipantFactoryQos_is_equal (const struct DDS_DomainParticipantFactoryQos * *left*, const struct DDS_DomainParticipantFactoryQos * *right*)

<<*cert*>> Compare two DDS_DomainParticipantFactoryQos policies for equality.

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

See also

DDS_DomainParticipantFactoryQos_INITIALIZER
DDS_DomainParticipantFactoryQos_initialize

5.2.4.4 DDS_DomainParticipantFactory* DDS_DomainParticipantFactory_get_instance (void)

<<*cert*>> Gets the singleton instance of this class.

Returns

the DDS_DomainParticipantFactory instance.

MT Safety:

UNSAFE on the FIRST call to DDS_DomainParticipantFactory_get_instance(). It is not safe for two threads to simultaneously make the first call to get an instance of the factory. Subsequent calls are thread safe.

DDS_TheParticipantFactory can be used as an alias for the singleton factory returned by this operation.

DDS_DomainParticipantFactory_get_instance also initializes the RTI Connext Micro library and should be called before any other API, unless the API is explicitly mentioned as safe to call before DDS_DomainParticipantFactory_get_instance.

Returns

the singleton DDS_DomainParticipantFactory instance.

See also

DDS_TheParticipantFactory

5.2.4.5 DDS_DomainParticipant* DDS_DomainParticipantFactory_create_participant (DDS_DomainParticipantFactory * *self*, DDS_DomainId_t *domainId*, const struct DDS_DomainParticipantQos * *qos*, const struct DDS_DomainParticipantListener * *listener*, DDS_StatusMask *mask*)

<<*cert*>> Creates a new DDS_DomainParticipant object.

Precondition

The specified QoS policies must be consistent, or the operation will fail and no DDS_DomainParticipant will be created.

If you want to create multiple participants on a given host in the same domain, make sure each one has a different participant index (set in the DDS_WireProtocolQosPolicy). This in turn will ensure each participant uses a different port number (since the unicast port numbers are calculated from the participant index and the domain ID).

Note that if there is a single participant per host in a given domain, the participant index can be left at the default value (-1).

Precondition

if `listener` is specified, none of the listener callback functions can be NULL.

Parameters

self	<<*in*>> Cannot be NULL.
domainId	<<*in*>> ID of the domain that the application intends to join. **[range]** [>=0], and does not violate guidelines stated in DDS_RtpsWellKnownPorts_t.
qos	<<*in*>> the domain participant's QoS. The special value DDS_PARTICIPANT_QOS_DEFAULT can be used to indicate that the DDS_DomainParticipant should be created with the default DDS_DomainParticipantQos set in the DDS_DomainParticipantFactory. Cannot be NULL.
listener	<<*in*>> the domain participant's listener.
mask	<<*in*>> Changes of communication status to be invoked on the listener.

Returns

domain participant or NULL on failure

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

Specifying QoS on entities for information on setting QoS before entity creation
DDS_DomainParticipantQos for rules on consistency among QoS
DDS_PARTICIPANT_QOS_DEFAULT

5.2.4.6 DDS_DomainParticipant* DDS_DomainParticipantFactory_lookup_participant (DDS_DomainParticipantFactory * *self*, DDS_DomainId_t *domainId*)

<<*cert*>> Locates an existing DDS_DomainParticipant.

If no such DDS_DomainParticipant exists, the operation will return NULL value.

If multiple DDS_DomainParticipant entities belonging to that domainId exist, then the operation will return one of them. It is not specified which one.

Parameters

self	<<*in*>> Cannot be NULL.
domainId	<<*in*>> ID of the domain participant to lookup.

Returns

domain participant if it exists, or NULL

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.2.4.7 DDS_ReturnCode_t DDS_DomainParticipantFactory_get_qos (DDS_DomainParticipantFactory * *self*, struct DDS_DomainParticipantFactoryQos * *qos*)

<<*cert*>> Gets the value for participant factory QoS.

Parameters

self	<<*in*>> Cannot be NULL.
qos	<<*inout*>> QoS to be filled up. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.2.4.8 DDS_ReturnCode_t DDS_DomainParticipantFactory_set_qos (DDS_DomainParticipantFactory * *self*, const struct DDS_DomainParticipantFactoryQos * *qos*)

<<*cert*>> Sets the value for a participant factory QoS.

The DDS_DomainParticipantFactoryQos::entity_factory can be changed. The other policies are immutable.

Note that despite having QoS, the DDS_DomainParticipantFactory is not an DDS_Entity.

Note that this function may cause RTI Connext Micro to free and reallocate memory, depending on the QoS policies that are changed.

Parameters

self	<<*in*>> Cannot be NULL.
qos	<<*in*>> Set of policies to be applied to DDS_DomainParticipantFactory. Policies must be consistent. Immutable policies can only be changed before calling any other RTI Connext Micro functions except for DDS_DomainParticipantFactory_get_qos Cannot be NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_IMMUTABLE_POLICY if immutable policy is changed, or DDS_RETCODE_INCONSISTENT_POLICY if policies are inconsistent

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_DomainParticipantFactoryQos for rules on consistency among QoS

5.2.4.9 RT_Registry_T* DDS_DomainParticipantFactory_get_registry (DDS_DomainParticipantFactory * *self*)

<<*cert*>> Retrieves the singleton registry instance associated with the factory.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

the registry instance or NULL on failure.

MT Safety:

This operation is thread safe.

5.2.5 Variable Documentation

5.2.5.1 struct DDS_DomainParticipantFactoryQos DDS_PARTICIPANT_FACTORY_QOS_DEFAULT

<<*cert*>> Special value for creating domain participant with default QoS.

When used in DDS_DomainParticipantFactory_create_participant, this special value is used to indicate that the DDS_DomainParticipant should be created with the default DDS_DomainParticipant QoS.

See also

DDS_DomainParticipantFactory_create_participant()

5.2.5.2 struct DDS_DomainParticipantQos DDS_PARTICIPANT_QOS_DEFAULT

<<*cert*>> Special value for creating domain participant with default QoS.

When used in DDS_DomainParticipantFactory_create_participant, this special value is used to indicate that the DDS_DomainParticipant should be created with the default DDS_DomainParticipant QoS.

See also

DDS_DomainParticipantFactory_create_participant()

5.3 DomainParticipant

DDS_DomainParticipant entity and associated elements

Data Structures

- struct DDS_DomainParticipantListener

  <<interface>> <<cert>> Listener for participant status.
- struct DDS_DomainParticipantQos

  <<cert>> QoS policies supported by a DDS_DomainParticipant entity.

Macros

- #define DDS_DomainParticipantListener_INITIALIZER

  <<cert>> Initializer for new DDS_DomainParticipantListener.
- #define DDS_DomainParticipantQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.

Typedefs

- typedef struct DDS_DomainParticipantImpl DDS_DomainParticipant

  <<interface>> <<cert>> Container for all DDS_DomainEntity objects.

Functions

- DDS_ReturnCode_t DDS_DomainParticipantQos_initialize (struct DDS_DomainParticipantQos *self)

  <<cert>> Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_DomainParticipantQos_copy (struct DDS_DomainParticipantQos *self, const struct DDS_DomainParticipantQos *source)

  <<cert>> Copy the contents of the given QoS into this QoS. The destination structure must be preallocated and initialized.
- DDS_Boolean DDS_DomainParticipantQos_is_equal (const struct DDS_DomainParticipantQos *left, const struct DDS_DomainParticipantQos *right)

  <<cert>> Compare two DDS_DomainParticipantQos policies for equality.
- DDS_Entity * DDS_DomainParticipant_as_entity (DDS_DomainParticipant *self)

  <<cert>> Access a DDS_DomainParticipant's supertype instance.
- DDS_Publisher * DDS_DomainParticipant_create_publisher (DDS_DomainParticipant *self, const struct DDS_PublisherQos *qos, const struct DDS_PublisherListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_Publisher with the desired QoS policies and attaches to it the specified DDS_PublisherListener.
- DDS_Subscriber * DDS_DomainParticipant_create_subscriber (DDS_DomainParticipant *self, const struct DDS_SubscriberQos *qos, const struct DDS_SubscriberListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_Subscriber with the desired QoS policies and attaches to it the specified DDS_SubscriberListener.
- DDS_Topic * DDS_DomainParticipant_create_topic (DDS_DomainParticipant *self, const char *topic_name, const char *type_name, const struct DDS_TopicQos *qos, const struct DDS_TopicListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_Topic with the desired QoS policies
- DDS_TopicDescription * DDS_DomainParticipant_lookup_topicdescription (DDS_DomainParticipant *self, const char *topic_name)

  <<cert>> Lookup an existing locally-created DDS_TopicDescription, based on its name.

- DDS_ReturnCode_t DDS_DomainParticipant_add_peer (DDS_DomainParticipant *self, const char *peer)

  <<eXtension>> <<cert>> Attempt to contact one or more additional peer participants.
- DDS_DomainId_t DDS_DomainParticipant_get_domain_id (DDS_DomainParticipant *self)

  <<cert>> Get the unique domain identifier.
- DDS_ReturnCode_t DDS_DomainParticipant_get_current_time (DDS_DomainParticipant *self, struct DDS_Time_t *current_time)

  <<cert>> Returns the current value of the time.
- DDS_ReturnCode_t DDS_DomainParticipant_register_type (DDS_DomainParticipant *self, const char *type_name, struct NDDS_Type_Plugin *plugin)

  <<eXtension>> <<cert>> Allows an application to communicate to RTI Connext Micro the existence of a data type.

Variables

- struct DDS_TopicQos DDS_TOPIC_QOS_DEFAULT

  <<cert>> Special value for creating a DDS_Topic with default QoS. Note that this is the only supported parameter in RTI Connext Micro when creating a topic.
- struct DDS_PublisherQos DDS_PUBLISHER_QOS_DEFAULT

  <<cert>> Special value for creating a DDS_Publisher with default QoS.
- struct DDS_SubscriberQos DDS_SUBSCRIBER_QOS_DEFAULT

  <<cert>> Special value for creating a DDS_Subscriber with default QoS.

5.3.1 Detailed Description

DDS_DomainParticipant entity and associated elements

5.3.2 Macro Definition Documentation

5.3.2.1 #define DDS_DomainParticipantListener_INITIALIZER

<<cert>> Initializer for new DDS_DomainParticipantListener.

No memory is allocated. New DDS_DomainParticipantListener instances stored on the stack should be initialized with this value before they are passed to any functions.

```
DDS_DomainParticipantListener listener =
     DDS_DomainParticipantListener_INITIALIZER
     ;
```

See also

DDS_DomainParticipantListener

5.3.2.2 #define DDS_DomainParticipantQos_INITIALIZER

<<cert>> Initializer for new QoS instances.

DDS_DomainParticipantQos instances stored on the stack must be initialized with this value before they are passed to any functions. This step ensures that the contained QoS policies that use dynamic memory are properly initialized. The initializer does not allocate memory.

The simplest way to create a new QoS structure is to initialize it on the stack at the time of its creation:

```
DDS_DomainParticipantQos myQos =
     DDS_DomainParticipantQos_INITIALIZER;
```

5.3.3 Typedef Documentation

5.3.3.1 typedef struct DDS_DomainParticipantImpl DDS_DomainParticipant

<<*interface*>> <<*cert*>> Container for all DDS_DomainEntity objects.

The DomainParticipant object plays several roles:

- It acts as a container for all other DDS_Entity objects.

- It acts as *factory* for the DDS_Publisher, DDS_Subscriber, DDS_Topic and DDS_Entity objects.

- It represents the participation of the application on a communication plane that isolates applications running on the same set of physical computers from each other. A domain establishes a virtual network linking all applications that share the same `domainId` and isolating them from applications running on different domains. In this way, several independent distributed applications can coexist in the same physical network without interfering, or even being aware of each other.

A DDS_DomainParticipant may be created in a disabled or enabled state; refer to DDS_Entity_enable for details. The initial state of the DDS_DomainParticipant is determined by DDS_DomainParticipantFactoryQos::entity_factory.

The documentation for each operation on the DDS_DomainParticipant includes any restrictions with respect to whether the operation is valid only for a disabled or enabled DDS_DomainParticipant.

The following operations may be called even if the DDS_DomainParticipant is not enabled. Other operations will fail with the value DDS_RETCODE_NOT_ENABLED if called on a disabled DomainParticipant:

- Operations defined at the base-class level namely, enable()
- Factory operations: create_topic(), create_publisher(), create_subscriber()

QoS:

DDS_DomainParticipantQos

Status:

Status Kinds

Listener:

DDS_DomainParticipantListener

5.3.4 Function Documentation

5.3.4.1 DDS_ReturnCode_t DDS_DomainParticipantQos_initialize (struct DDS_DomainParticipantQos * *self*)

<<*cert*>> Initializer for new QoS instances.

DDS_DomainParticipantQos instances on heap should be initialized with this function before they are passed to any functions. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This function does not allocate memory.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

One of the Standard Return Codes

5.3.4.2 DDS_ReturnCode_t DDS_DomainParticipantQos_copy (struct DDS_DomainParticipantQos * *self*, const struct DDS_DomainParticipantQos * *source*)

<<*cert*>> Copy the contents of the given QoS into this QoS. The destination structure must be preallocated and

initialized.

DDS_DomainParticipantQos instances can use dynamic memory because of the sequences contained in some QoS policies. A shallow copy by assignment is therefore unsafe. This function performs a deep-copy, allocating memory as necessary.

Parameters

self	<<*in*>> Cannot be NULL.
source	<<*in*>>. QoS to be copied from.

Returns

One of the Standard Return Codes

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

See also

DDS_DomainParticipantQos_INITIALIZER
DDS_DomainParticipantQos_initialize

5.3.4.3 DDS_Boolean DDS_DomainParticipantQos_is_equal (const struct DDS_DomainParticipantQos * *left*, const struct DDS_DomainParticipantQos * *right*)

<<*cert*>> Compare two DDS_DomainParticipantQos policies for equality.

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

See also

DDS_DomainParticipantQos_INITIALIZER
DDS_DomainParticipantQos_initialize

5.3.4.4 DDS_Entity* DDS_DomainParticipant_as_entity (DDS_DomainParticipant * *self*)

<<*cert*>> Access a DDS_DomainParticipant's supertype instance.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

DDS_DomainParticipant's supertype DDS_Entity instance

5.3.4.5 DDS_Publisher* DDS_DomainParticipant_create_publisher (DDS_DomainParticipant * *self*, const struct DDS_PublisherQos * *qos*, const struct DDS_PublisherListener * *listener*, DDS_StatusMask *mask*)

<<*cert*>> Creates a DDS_Publisher with the desired QoS policies and attaches to it the specified DDS_PublisherListener.

Precondition

The specified QoS policies must be consistent, or the operation will fail and no DDS_Publisher will be created. if `listener` is specified, none of the listener callback functions can be NULL.

Parameters

self	<<*in*>> Cannot be NULL.
qos	<<*in*>> QoS to be used for creating the new DDS_Publisher. The special value DDS_PUBLISHER_QOS_DEFAULT can be used to indicate that the DDS_Publisher should be created with the default DDS_PublisherQos set in the DDS_DomainParticipant. Cannot be NULL.
listener	<<*in*>>. Listener to be attached to the newly created DDS_Publisher.
mask	<<*in*>>. Changes of communication status to be invoked on the listener.

Returns

newly created publisher object or NULL on failure.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

Specifying QoS on entities for information on setting QoS before entity creation
DDS_PublisherQos for rules on consistency among QoS
DDS_PUBLISHER_QOS_DEFAULT

5.3.4.6 DDS_Subscriber* DDS_DomainParticipant_create_subscriber (DDS_DomainParticipant * *self*, const struct DDS_SubscriberQos * *qos*, const struct DDS_SubscriberListener * *listener*, DDS_StatusMask *mask*)

<<*cert*>> Creates a DDS_Subscriber with the desired QoS policies and attaches to it the specified DDS_SubscriberListener.

Precondition

The specified QoS policies must be consistent, or the operation will fail and no DDS_Subscriber will be created. if `listener` is specified, none of the listener callback functions can be NULL.

Parameters

self	<<*in*>> Cannot be NULL.
qos	<<*in*>> QoS to be used for creating the new DDS_Subscriber. The special value DDS_SUBSCRIBER_QOS_DEFAULT can be used to indicate that the DDS_Subscriber should be created with the default DDS_SubscriberQos set in the DDS_DomainParticipant. Cannot be NULL.
listener	<<*in*>>. Listener to be attached to the newly created DDS_Subscriber.
mask	<<*in*>>. Changes of communication status to be invoked on the listener.

Returns

newly created subscriber object or NULL on failure.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

Specifying QoS on entities for information on setting QoS before entity creation
DDS_SubscriberQos for rules on consistency among QoS
DDS_SUBSCRIBER_QOS_DEFAULT

5.3.4.7 DDS_Topic* DDS_DomainParticipant_create_topic (DDS_DomainParticipant * *self*, const char * *topic_name*, const char * *type_name*, const struct DDS_TopicQos * *qos*, const struct DDS_TopicListener * *listener*, DDS_StatusMask *mask*)

<<*cert*>> Creates a DDS_Topic with the desired QoS policies

Precondition

The application is not allowed to create two DDS_TopicDescription objects with the same `topic_name` attached to the same DDS_DomainParticipant. If the application attempts this, this function will fail and return NULL.
The specified QoS policies must be consistent, or the operation will fail and no DDS_Topic will be created.
Prior to creating a DDS_Topic, the type must have been registered with RTI Connext Micro. This is done using the DDS_DomainParticipant_register_type operation on a derived class of the NDDS_Type_Plugin interface.

Parameters

self	<<*in*>> Cannot be NULL.
topic_name	<<*in*>> Name for the new topic, must not exceed 255 characters. Cannot be NULL.
type_name	<<*in*>> The type to which the new DDS_Topic will be bound. Cannot be NULL.
qos	<<*in*>> Must be DDS_TOPIC_QOS_DEFAULT.

Cannot be NULL.

Parameters

listener	<<*in*>>. Listener to be attached to the newly created DDS_Topic.
mask	<<*in*>>. Changes of communication status to be invoked on the listener.

Returns

newly created topic, or NULL on failure

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

Specifying QoS on entities for information on setting QoS before entity creation
DDS_TopicQos for rules on consistency among QoS
DDS_TOPIC_QOS_DEFAULT

5.3.4.8 DDS_TopicDescription* DDS_DomainParticipant_lookup_topicdescription (DDS_DomainParticipant * *self*, const char * *topic_name*)

<<cert>> Lookup an existing locally-created DDS_TopicDescription, based on its name.

The returned topic can either be enabled or disabled.

The operation DDS_DomainParticipant_lookup_topicdescription searches among the locally created topics and returns a pointer to a DDS_TopicDescription. It is possible to delete the DDS_TopicDescription returned by DDS_DomainParticipant_lookup_topicdescription, provided it has no readers or writers, but then it is really deleted and subsequent lookups will fail.

Parameters

self	*<<in>>* Cannot be NULL.
topic_name	*<<in>>* Name of DDS_TopicDescription to search for. This string must be no more than 255 characters. Cannot be NULL.

Returns

the topic description if it has already been created locally, or NULL otherwise.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.3.4.9 DDS_ReturnCode_t DDS_DomainParticipant_add_peer (DDS_DomainParticipant * *self*, const char * *peer*)

<<eXtension>> *<<cert>>* Attempt to contact one or more additional peer participants.

Add the given peer description to the list of peers with which this DDS_DomainParticipant will try to communicate.

This function may be called at any time after this DDS_DomainParticipant has been created, before or after it has been enabled. If it is called after DDS_Entity_enable, an attempt will be made to contact the new peer(s) immediately. If it is called before, the peer description will simply be added to the list that was populated by DDS_DiscoveryQosPolicy::initial_peers; the first attempted contact will take place after this DDS_DomainParticipant is enabled.

Adding a peer description with this function does not guarantee that any peer(s) discovered as a result will exactly correspond to those described:

- This DDS_DomainParticipant will attempt to discover peer participants at the given locations, but may not succeed if no such participants are available. Such a situation will not result in an error result from this function, which will not wait for contact attempt(s) to be made.

- If remote participants such as are described by the given peer description are discovered, the distributed application is configured with asymmetric peer lists, and DDS_DiscoveryQosPolicy::accept_unknown_peers is set to DDS_BOOLEAN_TRUE this DDS_DomainParticipant may actually discover *more* peers than are described in the given peer description.

Adding a peer description with this function has no effect on DDS_DiscoveryQosPolicy::initial_peers.

Parameters

self	<<*in*>> Cannot be NULL.
peer	<<*in*>> New peer descriptor to be added. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_DiscoveryQosPolicy::initial_peers
DDS_PARTICIPANT_TOPIC_NAME

5.3.4.10 DDS_DomainId_t DDS_DomainParticipant_get_domain_id (DDS_DomainParticipant * *self*)

<<*cert*>> Get the unique domain identifier.

This operation retrieves the `domain_id` used to create the DDS_DomainParticipant. The `domain_id` identifies the DDS domain to which the DDS_DomainParticipant belongs. Each DDS domain represents a separate data 'communication plane' isolated from other domains.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

the unique `domainId` that was used to create the domain

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_DomainParticipantFactory_create_participant

5.3.4.11 DDS_ReturnCode_t DDS_DomainParticipant_get_current_time (DDS_DomainParticipant * *self*, struct DDS_Time_t * *current_time*)

<<*cert*>> Returns the current value of the time.

The current value of the time that RTI Connext Micro uses to time-stamp DDS_DataWriter and to set the reception-timestamp for the data updates that it receives. Note that the definition of the current time is dependent on the platform and may not be the actual `wall-clock` time. Additional information for specific platforms can be found in the User's Manual.

Parameters

self	<<*in*>> Cannot be NULL.
current_time	<<*inout*>> Current time to be filled up. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.3.4.12 DDS_ReturnCode_t DDS_DomainParticipant_register_type (DDS_DomainParticipant * *self*, const char * *type_name*, struct NDDS_Type_Plugin * *plugin*)

<<*eXtension*>> <<*cert*>> Allows an application to communicate to RTI Connext Micro the existence of a data type.

The *generated* implementation of the operation embeds all the knowledge that has to be communicated to the middleware in order to make it able to manage the contents of data of that type. This includes in particular the key definition that will allow RTI Connext Micro to distinguish different instances of the same type.

The same NDDS_Type_Plugin can be registered multiple times with a DDS_DomainParticipant using the same or different values for the `type_name`. If `register_type` is called multiple times on the same NDDS_Type_Plugin with the same DDS_DomainParticipant and `type_name`, the second (and subsequent) registrations are ignored by the operation fails with DDS_RETCODE_OK.

Precondition

Cannot use the same `type_name` to register two different NDDS_Type_Plugin with the same DDS_DomainParticipant, or else the operation will fail and DDS_RETCODE_PRECONDITION_NOT_MET will be returned.

Parameters

self	<<*in*>> Cannot be NULL.
type_name	<<*in*>> the type name under with the data type `Foo` is registered with the participant; this type name is used when creating a new DDS_Topic. (See DDS_DomainParticipant_create_topic.) The name may not be NULL or longer than 255 characters.
plugin	<<*in*>> the `Foo` type plugin to register the data type with. Cannot be NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_PRECONDITION_NOT_MET or DDS_RETCODE_OUT_OF_RESOURCES.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_DomainParticipant_create_topic

5.3.5 Variable Documentation

5.3.5.1 struct DDS_TopicQos DDS_TOPIC_QOS_DEFAULT

<<*cert*>> Special value for creating a DDS_Topic with default QoS. Note that this is the only supported parameter in RTI Connext Micro when creating a topic.

RTI Connext Micro does not support the Topic QoS. This represents the a placeholder for Topic QoS values. This is the only valid paramater that can be passed into the DDS_DomainParticipant_create_topic function (no other values for QoS are supported).

See also

DDS_DomainParticipant_create_topic

5.3.5.2 struct DDS_PublisherQos DDS_PUBLISHER_QOS_DEFAULT

<<*cert*>> Special value for creating a DDS_Publisher with default QoS.

When used in DDS_DomainParticipant_create_publisher, this special value is used to indicate that the DDS_Publisher should be created with the default DDS_Publisher QoS.

See also

DDS_DomainParticipant_create_publisher

5.3.5.3 struct DDS_SubscriberQos DDS_SUBSCRIBER_QOS_DEFAULT

<<*cert*>> Special value for creating a DDS_Subscriber with default QoS.

When used in DDS_DomainParticipant_create_subscriber, this special value is used to indicate that the DDS_Subscriber should be created with the default DDS_Subscriber QoS.

See also

DDS_DomainParticipant_create_subscriber

5.4 Discovery

Descriptions of remote entities used for discovery.

Modules

- Participant Built-in Topic

  Builtin topic for accessing information about the DomainParticipants discovered by RTI Connext Micro.
- Publication Built-in Topic

  Builtin topic for configuring information about the Publications to be discovered by RTI Connext Micro.
- Subscription Built-in Topic

  Builtin topic for configuring information about the Subscriptions to be discovered by RTI Connext Micro.

Data Structures

- struct DDS_Locator_t

  <<eXtension>> <<cert>> Type used to represent the addressing information needed to send a message to an RTPS Endpoint using one of the supported transports.
- struct DDS_LocatorSeq

  Declares IDL `sequence` < DDS_Locator_t >
- struct DDS_ProtocolVersion

  <<eXtension>> <<cert>> Type used to represent the version of the RTPS protocol.
- struct DDS_VendorId

  <<eXtension>> <<cert>> Type used to represent the vendor of the service implementing the RTPS protocol.
- struct DDS_VendorId_t

  <<eXtension>> <<cert>> Type used to represent the vendor of the service implementing the RTPS protocol.
- struct DDS_BuiltinTopicKey_t

  The key type of the built-in topic types.

Macros

- #define DDS_LOCATOR_INVALID RTPS_LOCATOR_INVALID

  An invalid locator.
- #define DDS_LOCATOR_KIND_UDPv4

  A locator for a UDPv4 address.
- #define DDS_LOCATOR_KIND_UDPv6

  A locator for a UDPv6 address.
- #define DDS_LOCATOR_KIND_RESERVED

  Locator of this kind is reserved.
- #define DDS_LOCATOR_KIND_SHMEM

  A locator for an address acessed via shared memory.
- #define DDS_PROTOCOLVERSION_1_0 { 1, 0 }

  The protocol version 1.0.
- #define DDS_PROTOCOLVERSION_1_1 { 1, 1 }

  The protocol version 1.1.
- #define DDS_PROTOCOLVERSION_1_2 { 1, 2 }

  The protocol version 1.2.
- #define DDS_PROTOCOLVERSION_2_0 { 2, 0 }

  The most recent protocol version. Currently 1.2.
- #define DDS_PROTOCOLVERSION_2_1 { 2, 1 }

  The protocol version 2.1.

- #define DDS_PROTOCOLVERSION { 2, 1 }

  The most recent protocol version. Currently 1.2.
- #define DDS_VENDOR_ID_LENGTH_MAX 2

  Length of vendor id.

Typedefs

- typedef struct DDS_ProtocolVersion DDS_ProtocolVersion_t

  <<eXtension>> <<cert>> Type used to represent the version of the RTPS protocol.
- typedef struct DDS_BuiltinTopicKey_t DDS_BuiltinTopicKey_t

  The key type of the built-in topic types.

5.4.1 Detailed Description

Descriptions of remote entities used for discovery. Given a DDS entity, a remote entity is any other entity that can be discovered and matched. A built-in object is a description of a remote entity, including all the QoS which are important for discovery and matching of remote readers and writers with local readers and writers.

RTI Connext Micro must discover and keep track of the remote entities, such as new participants in the domain. Depending on the type of discovery used, this information about remote entities is gathered in three ways:

1. No data is sent over the network (static discovery)

2. A subset is sent over the network (dynamic participant/static endpoint discovery), while the rest is statically asserted.

3. All data about DomainParticipants DataReaders and DataWriters is over the network (simple discovery)

In the first case, all information about remote DomainParticipants DataReaders and DataWriters is statically configured by the user before starting the middleware. To do this static configuration, the user must fill in the built-in structures for all remote entities it wishes to discover, and must asssert them to the middleware. This type of discovery is not supported in this version of RTI Connext Micro.

In the second case, the participant data is sent over the network, but the endpoint data is statically configured. So, the local application fills in builtin structures representing the QoS and object IDs all the remote readers and writers it wishes to discover. For this type of discovery the user does not configure builtin data for remote participants. This type of discovery is faster than Simple Discovery, and does not require reliable communication. This is supported by RTI Connext Micro and is known as Dynamic Participant, Static Endpoint (DPSE) discovery.

In the last case, the user does not need to fill in any data about remote entities, because it is automatically sent over the network by the middleware. The user can access the data through special listeners for discovery traffic.

This is not supported by RTI Connext Micro and is known as Dynamic Participant, Dynamic Endpoint (DPDE) discovery. Refer to DDS_ParticipantBuiltinTopicData, DDS_SubscriptionBuiltinTopicData and DDS_PublicationBuiltinTopicData for a description of all the built-in topics and their contents.

<<eXtension>> <<cert>> Dynamic Participant, Static Endpoint Discovery plugin mechanism for RTI Connext Micro.

5.4.2 Macro Definition Documentation

5.4.2.1 #define DDS_LOCATOR_INVALID RTPS_LOCATOR_INVALID

An invalid locator.

5.4.2.2 #define DDS_LOCATOR_KIND_UDPv4

A locator for a UDPv4 address.

5.4.2.3 #define DDS_LOCATOR_KIND_UDPv6

A locator for a UDPv6 address.

5.4.2.4 #define DDS_LOCATOR_KIND_RESERVED

Locator of this kind is reserved.

5.4.2.5 #define DDS_LOCATOR_KIND_SHMEM

A locator for an address acessed via shared memory.

5.4.2.6 #define DDS_PROTOCOLVERSION_1_0 { 1, 0 }

The protocol version 1.0.

5.4.2.7 #define DDS_PROTOCOLVERSION_1_1 { 1, 1 }

The protocol version 1.1.

5.4.2.8 #define DDS_PROTOCOLVERSION_1_2 { 1, 2 }

The protocol version 1.2.

5.4.2.9 #define DDS_PROTOCOLVERSION_2_0 { 2, 0 }

The most recent protocol version. Currently 1.2.

5.4.2.10 #define DDS_PROTOCOLVERSION_2_1 { 2, 1 }

The protocol version 2.1.

5.4.2.11 #define DDS_PROTOCOLVERSION { 2, 1 }

The most recent protocol version. Currently 1.2.

5.4.2.12 #define DDS_VENDOR_ID_LENGTH_MAX 2

Length of vendor id.

5.4.3 Typedef Documentation

5.4.3.1 typedef struct DDS_ProtocolVersion DDS_ProtocolVersion_t

<<eXtension>> *<<cert>>* Type used to represent the version of the RTPS protocol.

5.4.3.2 typedef struct DDS_BuiltinTopicKey_t DDS_BuiltinTopicKey_t

The key type of the built-in topic types.

Each remote DDS_Entity to be discovered can be uniquely identified by this key. This is the key of all the built-in topic data types.

The value of element DDS_BUILTIN_TOPIC_KEY_OBJECT_ID is the object ID of the remote entity. In Dynamic Participant/Static Endpoint discovery, this must be set correctly for each remote DDS_DataReader and DDS_DataWriter that an application intends to discover.

For example, to configure a remote DDS_DataWriter with object ID 100 by setting the key in the DDS_PublicationBuiltinTopicData:

```
remote_pub_data.key.value[DDS_BUILTIN_TOPIC_KEY_OBJECT_ID]  = 100;
```

See also

DDS_ParticipantBuiltinTopicData
DDS_PublicationBuiltinTopicData
DDS_SubscriptionBuiltinTopicData

5.5 Topic

Contains the DDS_Topic, the DDS_TopicListener interface, and more generally, all that is needed by an application to define DDS_Topic objects and the data types that are associated with DDS_Topic objects.

Modules

- Topic

  DDS_Topic entity and associated elements
- User Data Type Support

  Defines generic classes and macros to support user data types.
- User-Defined Types

  Generic Type entity and associated elements.
- DDS-Specific Primitive Types

  Basic DDS value types for use in user data types.

5.5.1 Detailed Description

Contains the DDS_Topic, the DDS_TopicListener interface, and more generally, all that is needed by an application to define DDS_Topic objects and the data types that are associated with DDS_Topic objects.

5.6 Topic

DDS_Topic entity and associated elements

Data Structures

- struct DDS_InconsistentTopicStatus

  DDS_INCONSISTENT_TOPIC_STATUS
- struct DDS_TopicQos

  QoS policies supported by a DDS_Topic entity.
- struct DDS_TopicListener

  <<interface>> DDS_Listener for DDS_Topic entities.

Macros

- #define DDS_InconsistentTopicStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_TopicListener_INITIALIZER

  <<cert>> Initializer for new DDS_TopicListener.

Typedefs

- typedef struct DDS_TopicDescriptionImpl DDS_TopicDescription

  <<interface>> Base class for DDS_Topic.
- typedef struct DDS_TopicImpl DDS_Topic

  <<interface>> The most basic description of the data to be published and subscribed.
- typedef void(* DDS_TopicListener_InconsistentTopicCallback)(void *listener_data, DDS_Topic *topic, const struct DDS_InconsistentTopicStatus *status)

  Prototype of DDS_TopicListener::on_inconsistent_topic.

Functions

- const char * DDS_TopicDescription_get_type_name (DDS_TopicDescription *self)

  <<cert>> Get the associated `type_name`.
- const char * DDS_TopicDescription_get_name (DDS_TopicDescription *self)

  <<cert>> Get the name used to create this DDS_TopicDescription.
- DDS_DomainParticipant * DDS_TopicDescription_get_participant (DDS_TopicDescription *self)

  <<cert>> Get the DDS_DomainParticipant to which the DDS_TopicDescription belongs.
- DDS_Entity * DDS_Topic_as_entity (DDS_Topic *self)

  <<cert>> Access a DDS_Topic's DDS_Entity supertype instance.
- DDS_TopicDescription * DDS_Topic_as_topicdescription (DDS_Topic *self)

  <<cert>> Access a DDS_Topic's DDS_TopicDescription supertype instance.
- DDS_Topic * DDS_Topic_narrow (DDS_TopicDescription *self)

  <<cert>> Narrow the given DDS_TopicDescription pointer to a DDS_Topic pointer.
- DDS_ReturnCode_t DDS_Topic_get_inconsistent_topic_status (DDS_Topic *self, struct DDS_InconsistentTopicStatus *status)

  <<cert>> Allows the application to retrieve the DDS_INCONSISTENT_TOPIC_STATUS status of a DDS_Topic.

5.6.1 Detailed Description

DDS_Topic entity and associated elements

5.6.2 Macro Definition Documentation

5.6.2.1 #define DDS_InconsistentTopicStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_InconsistentTopicStatus instances that are stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.6.2.2 #define DDS_TopicListener_INITIALIZER

<<*cert*>> Initializer for new DDS_TopicListener.

All the new DDS_TopicListener instances allocated in the stack should be initialized to this value.No memory is allocated.

```
DDS_TopicListener listener = DDS_TopicListener_INITIALIZER
    ;
 /* initialize listener functions */
 listener.on_inconsistent_topic = ....;
 DDS_Topic_set_listener(myTopic, &listener, mask);
```

See also

DDS_TopicListener

5.6.3 Typedef Documentation

5.6.3.1 typedef struct DDS_TopicDescriptionImpl DDS_TopicDescription

<<*interface*>> Base class for DDS_Topic.

DDS_TopicDescription represents the fact that both publications and subscriptions are tied to a single data-type. Its attribute `type_name` defines a unique resulting type for the publication or the subscription and therefore creates an implicit association with a type.

DDS_TopicDescription has also a `name` that allows it to be retrieved locally.

5.6.3.2 typedef struct DDS_TopicImpl DDS_Topic

<<*interface*>> The most basic description of the data to be published and subscribed.

QoS:

DDS_TopicQos

Status:

DDS_INCONSISTENT_TOPIC_STATUS, DDS_InconsistentTopicStatus

Listener:

DDS_TopicListener

A DDS_Topic is identified by its name, which must be unique in the whole domain. In addition (by virtue of extending DDS_TopicDescription) it fully specifies the type of the data that can be communicated when publishing or subscribing to the DDS_Topic.

DDS_Topic is the only DDS_TopicDescription that can be used for publications and therefore associated with a DDS_DataWriter.

The following operations may be called even if the DDS_Topic is not enabled. Other operations will fail with the value DDS_RETCODE_NOT_ENABLED if called on a disabled DDS_Topic:

- get_inconsistent_topic_status()

5.6.3.3 typedef void(* DDS_TopicListener_InconsistentTopicCallback)(void *listener_data, DDS_Topic *topic, const struct DDS_InconsistentTopicStatus *status)

Prototype of DDS_TopicListener::on_inconsistent_topic.

Parameters

listener_data	<<*out*>> Data associated with the listener when the listener is set
topic	<<*out*>> Locally created DDS_Topic that triggers the listener callback
status	<<*out*>> Current inconsistent status of the locally created DDS_Topic

5.6.4 Function Documentation

5.6.4.1 const char* DDS_TopicDescription_get_type_name (DDS_TopicDescription * *self*)

<<*cert*>> Get the associated `type_name`.

The type name defines a locally unique type for the publication or the subscription.

The `type_name` corresponds to a unique string used to register a type via the DDS_DomainParticipant_register_type function.

Thus, the `type_name` implies an association with a corresponding type and this DDS_TopicDescription.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

the type name. The returned type name is valid until the DDS_TopicDescription is deleted.

MT Safety:

This operation is thread safe.

Postcondition

The result is NULL if self is NULL, otherwise result is non-NULL.

5.6.4.2 const char* DDS_TopicDescription_get_name (DDS_TopicDescription * *self*)

<<*cert*>> Get the name used to create this DDS_TopicDescription.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

the name used to create this DDS_TopicDescription. The returned topic name is valid until the DDS_TopicDescription is deleted.

MT Safety:

This operation is thread safe.

Postcondition

The result is NULL if self is NULL, otherwise result is non-NULL.

5.6.4.3 DDS_DomainParticipant* DDS_TopicDescription_get_participant (DDS_TopicDescription * *self*)

<<*cert*>> Get the DDS_DomainParticipant to which the DDS_TopicDescription belongs.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

The DDS_DomainParticipant to which the DDS_TopicDescription belongs.

MT Safety:

This operation is thread safe.

Postcondition

TThe result is NULL if self is NULL, otherwise result is non-NULL.

5.6.4.4 DDS_Entity* DDS_Topic_as_entity (DDS_Topic * *self*)

<<*cert*>> Access a DDS_Topic's DDS_Entity supertype instance.

Returns

DDS_Topic's supertype DDS_Entity instance

5.6.4.5 DDS_TopicDescription* DDS_Topic_as_topicdescription (DDS_Topic * *self*)

<<*cert*>> Access a DDS_Topic's DDS_TopicDescription supertype instance.

Returns

DDS_Topic's supertype DDS_TopicDescription instance

5.6.4.6 DDS_Topic* DDS_Topic_narrow (DDS_TopicDescription * *self*)

<<*cert*>> Narrow the given DDS_TopicDescription pointer to a DDS_Topic pointer.

Returns

DDS_Topic if this DDS_TopicDescription is a DDS_Topic. Otherwise, return NULL.

5.6.4.7 DDS_ReturnCode_t DDS_Topic_get_inconsistent_topic_status (DDS_Topic * *self,* struct DDS_InconsistentTopicStatus * *status*)

<<*cert*>> Allows the application to retrieve the DDS_INCONSISTENT_TOPIC_STATUS status of a DDS_Topic.

Retrieve the current DDS_InconsistentTopicStatus

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> Status to be retrieved. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_InconsistentTopicStatus

5.7 User Data Type Support

Defines generic classes and macros to support user data types.

Data Structures

- struct FooTypeSupport

  <<interface>> <<generic>> User data type specific interface.

Macros

- #define DDS_TYPESUPPORT_C(TTypeSupport, TData)

  Declares the interface required to support a user data type.

Typedefs

- typedef DDS_HANDLE_TYPE_NATIVE DDS_InstanceHandle_t

  Type definition for an instance handle.

Functions

- Foo * FooTypeSupport_create_data (void)

  <<cert>> <<eXtension>> Create a data type and initialize it.
- DDS_ReturnCode_t FooTypeSupport_register_type (DDS_DomainParticipant *participant, const char *type_name)

  <<cert>> Allows an application to communicate to RTI Connext Micro the existence of a data type.
- DDS_Boolean DDS_InstanceHandle_equals (const DDS_InstanceHandle_t *self, const DDS_InstanceHandle_t *other)

  <<cert>> Compares this instance handle with another handle for equality.
- DDS_Boolean DDS_InstanceHandle_is_nil (const DDS_InstanceHandle_t *self)

  <<cert>> Compare this handle to DDS_HANDLE_NIL

Variables

- const DDS_InstanceHandle_t DDS_HANDLE_NIL

  <<cert>> The NIL instance handle.

5.7.1 Detailed Description

Defines generic classes and macros to support user data types. Defines the DDS user data type support.

DDS specifies strongly typed interfaces to read and write user data. For each data class defined by the application, there is a number of specialised classes that are required to facilitate the type-safe interaction of the application with RTI Connext Micro.

RTI Connext Micro provides an automatic means to generate all these type-specific classes with the RTI IDL Compiler User Manual utility. The complete set of automatic classes created for a hypothetical user data type named `Foo` are shown below.

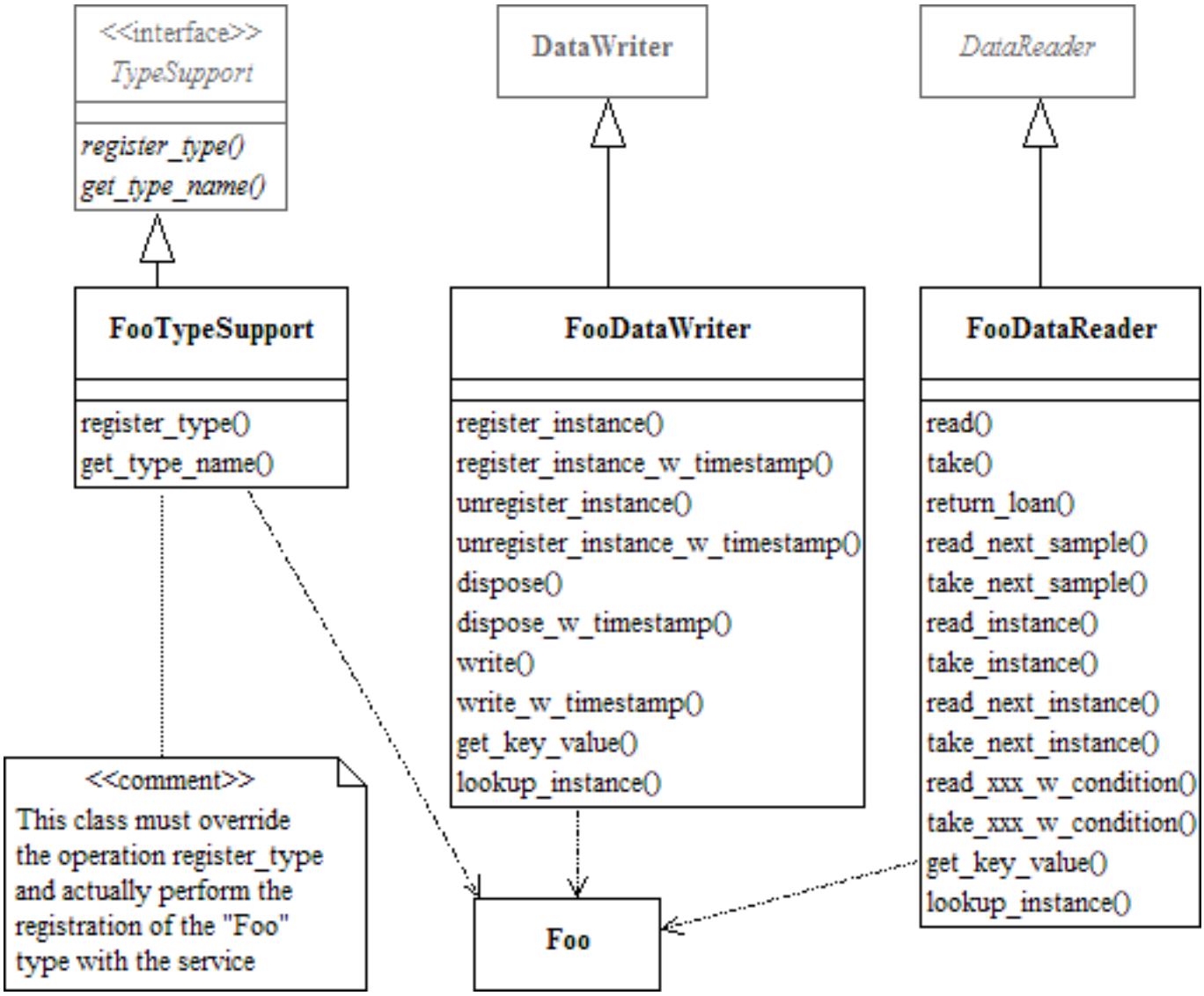

Figure 1: Classes auto-created for an application data type named `Foo`

The macros defined here declare the strongly typed APIs needed to support an arbitrary user defined data of type `Foo`.

5.7.2 Macro Definition Documentation

5.7.2.1 #define DDS_TYPESUPPORT_C(*TTypeSupport, TData*)

Declares the interface required to support a user data type.

Defines:

FooTypeSupport TypeSupport of type `Foo`, i.e. `FooTypeSupport`

5.7.3 Typedef Documentation

5.7.3.1 typedef DDS_HANDLE_TYPE_NATIVE DDS_InstanceHandle_t

Type definition for an instance handle.

Handle to identiy different instances of the same DDS_Topic of a certain type.

See also

FooDataWriter_register_instance
DDS_SampleInfo::instance_handle

5.7.4 Function Documentation

5.7.4.1 Foo* FooTypeSupport_create_data (void)

<<cert>> *<<eXtension>>* Create a data type and initialize it.

Memory for a data type is allocated from memory using internal RTI Connext Micro memory allocation APIs. The implementation of these APIs are specific to a particular platform. For this reason it is of the utmost importance that FooTypeSupport_create_data is not called until DDS_DomainParticipantFactory_get_instance has been called successfully. The DDS_DomainParticipantFactory initializes the RTI Connext Micro system, including any memory management initialization.

The *generated* implementation of the operation knows how to instantiate a data type and initialize it properly.

All memory for the type is deeply allocated.

Returns

newly created data type

MT Safety:

This operation is thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.7.4.2 DDS_ReturnCode_t FooTypeSupport_register_type (DDS_DomainParticipant * *participant*, const char * *type_name*)

<<cert>> Allows an application to communicate to RTI Connext Micro the existence of a data type.

The *generated* implementation of the operation embeds all the knowledge that has to be communicated to the middleware in order to make it able to manage the contents of data of that type. This includes in particular the key definition that will allow RTI Connext Micro to distinguish different instances of the same type.

The same DDS_TYPESUPPORT_C can be registered multiple times with a DDS_DomainParticipant using the same or different values for the `type_name`. If `register_type` is called multiple times on the same DDS_TYPESUPPORT_C with the same DDS_DomainParticipant and `type_name`, the second (and subsequent) registrations are ignored by the operation fails with DDS_RETCODE_OK.

Precondition

Cannot use the same `type_name` to register two different DDS_TYPESUPPORT_C with the same DDS_DomainParticipant, or else the operation will fail and DDS_RETCODE_PRECONDITION_NOT_MET will be returned.

Parameters

participant	*<<in>>* the DDS_DomainParticipant to register the data type `Foo` with. Cannot be NULL.
type_name	*<<in>>* the type name under with the data type `Foo` is registered with the participant; this type name is used when creating a new DDS_Topic. (See DDS_DomainParticipant_create_topic.) The name may not be NULL or longer than 255 characters.

Returns

One of the Standard Return Codes, DDS_RETCODE_PRECONDITION_NOT_MET or DDS_RETCODE_OUT_OF_RESOURCES.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_DomainParticipant_create_topic

5.7.4.3 DDS_Boolean DDS_InstanceHandle_equals (const DDS_InstanceHandle_t * *self*, const DDS_InstanceHandle_t * *other*)

<<cert>> Compares this instance handle with another handle for equality.

Parameters

self	*<<in>>* This handle. Cannot be NULL.
other	*<<in>>* The other handle to be compared with this handle. Cannot be NULL.

Returns

DDS_BOOLEAN_TRUE if the two handles have equal values, or DDS_BOOLEAN_FALSE otherwise.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

See also

DDS_InstanceHandle_is_nil

5.7.4.4 DDS_Boolean DDS_InstanceHandle_is_nil (const DDS_InstanceHandle_t * *self*)

<<cert>> Compare this handle to DDS_HANDLE_NIL

Returns

DDS_BOOLEAN_TRUE if the given instance handle is equal to DDS_HANDLE_NIL or DDS_BOOLEAN_FALSE otherwise.

See also

DDS_InstanceHandle_equals

5.7.5 Variable Documentation

5.7.5.1 const DDS_InstanceHandle_t DDS_HANDLE_NIL

<<cert>> The NIL instance handle.

Special DDS_InstanceHandle_t value

See also

DDS_InstanceHandle_is_nil

5.8 Publication Module

Contains the DDS_Publisher, and DDS_DataWriter classes, and more generally, all that is needed on the publication side.

Modules

- Publisher

  DDS_Publisher entity and associated elements
- DataWriter

  DDS_DataWriter entity and associated elements

5.8.1 Detailed Description

Contains the DDS_Publisher, and DDS_DataWriter classes, and more generally, all that is needed on the publication side. Defines the DDS publication package.

This diagram defines the elements that are defined in the DDS specification, and their relationships to each other. Not all entities and functions shown in the diagram are supported by RTI Connext Micro. Please read the information on the individual types to determine whether a function is supported.

5.9 Publisher

DDS_Publisher entity and associated elements

Data Structures

- struct DDS_PublisherQos

  <<cert>> QoS policies supported by a DDS_Publisher entity.
- struct DDS_PublisherListener

  <<interface>> <<cert>> DDS_Listener for DDS_Publisher status.

Macros

- #define DDS_PublisherQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_PublisherListener_INITIALIZER

  <<cert>> Initializer for new DDS_PublisherListener.

Typedefs

- typedef struct DDS_PublisherImpl DDS_Publisher

  <<interface>> <<cert>> A publisher is the object responsible for the actual dissemination of publications.

Functions

- DDS_ReturnCode_t DDS_PublisherQos_initialize (struct DDS_PublisherQos *self)

  <<cert>> Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_PublisherQos_copy (struct DDS_PublisherQos *self, const struct DDS_PublisherQos *source)

  <<cert>> Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_PublisherQos_is_equal (const struct DDS_PublisherQos *left, const struct DDS_PublisherQos *right)

  <<cert>> Compare two DDS_PublisherQos policies for equality.
- DDS_Entity * DDS_Publisher_as_entity (DDS_Publisher *publisher)

  <<cert>> Access a DDS_Publisher's supertype instance.
- DDS_DataWriter * DDS_Publisher_create_datawriter (DDS_Publisher *self, DDS_Topic *topic, const struct DDS_DataWriterQos *qos, const struct DDS_DataWriterListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_DataWriter that will be attached and belong to the DDS_Publisher.
- DDS_DataWriter * DDS_Publisher_lookup_datawriter (DDS_Publisher *self, const char *topic_name)

  <<cert>> Retrieves the DDS_DataWriter for a specific DDS_Topic.
- DDS_DomainParticipant * DDS_Publisher_get_participant (DDS_Publisher *self)

  <<cert>> Returns the DDS_DomainParticipant to which the DDS_Publisher belongs.

Variables

- struct DDS_DataWriterQos DDS_DATAWRITER_QOS_DEFAULT

  <<cert>> Special value for creating DDS_DataWriter with default QoS

5.9.1 Detailed Description

DDS_Publisher entity and associated elements

5.9.2 Macro Definition Documentation

5.9.2.1 #define DDS_PublisherQos_INITIALIZER

<<cert>> Initializer for new QoS instances.

New DDS_PublisherQos instance stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a new QoS structure is to initialize it on the stack at the time of its creation:

```
DDS_PublisherQos myQos = DDS_PublisherQos_INITIALIZER
    ;
```

5.9.2.2 #define DDS_PublisherListener_INITIALIZER

<<cert>> Initializer for new DDS_PublisherListener.

All the new DDS_PublisherListener instances allocated in the stack should be initialized to this value.No memory is allocated.

```
DDS_PublisherListener listener =
    DDS_PublisherListener_INITIALIZER;
```

See also

DDS_PublisherListener

5.9.3 Typedef Documentation

5.9.3.1 typedef struct DDS_PublisherImpl DDS_Publisher

<<interface>> *<<cert>>* A publisher is the object responsible for the actual dissemination of publications.

QoS:

DDS_PublisherQos

Listener:

DDS_PublisherListener

A publisher acts on the behalf of one or several DDS_DataWriter objects that belong to it. When it is informed of a change to the data associated with one of its DDS_DataWriter objects, it decides when it is appropriate to actually send the data-update message. In making this decision, it considers any extra information that goes with the data (timestamp, writer, etc.) as well as the QoS of the DDS_Publisher and the DDS_DataWriter.

The following operations may be called even if the DDS_Publisher is not enabled. Other operations will fail with the value DDS_RETCODE_NOT_ENABLED if called on a disabled DDS_Publisher:

- The base-class operations DDS_Entity_enable
- DDS_Publisher_create_datawriter

5.9.4 Function Documentation

5.9.4.1 DDS_ReturnCode_t DDS_PublisherQos_initialize (struct DDS_PublisherQos * *self*)

<<cert>> Initializer for new QoS instances.

New DDS_PublisherQos instances on heap should be initialized with this function before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This function does not allocate memory.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

One of the Standard Return Codes

5.9.4.2 DDS_ReturnCode_t DDS_PublisherQos_copy (struct DDS_PublisherQos * *self*, const struct DDS_PublisherQos * *source*)

<<*cert*>> Copy the contents of the given QoS into this QoS.

DDS_PublisherQos instances can use dynamic memory because of the sequences contained in some QoS policies. A shallow copy by assignment is therefore unsafe. This function performs a deep-copy, allocating memory if necessary.

Parameters

self	<<*in*>> Cannot be NULL.
source	<<*in*>> The DDS_PublisherQos to copy from.

Returns

One of the Standard Return Codes

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

See also

DDS_PublisherQos_INITIALIZER
DDS_PublisherQos_initialize

5.9.4.3 DDS_Boolean DDS_PublisherQos_is_equal (const struct DDS_PublisherQos * *left*, const struct DDS_PublisherQos * *right*)

<<*cert*>> Compare two DDS_PublisherQos policies for equality.

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

See also

DDS_PublisherQos_INITIALIZER
DDS_PublisherQos_initialize

5.9.4.4 DDS_Entity* DDS_Publisher_as_entity (DDS_Publisher * *publisher*)

<<*cert*>> Access a DDS_Publisher's supertype instance.

Parameters

publisher	<<*in*>> Cannot be NULL.

Returns

The DDS_Entity that is supertype instance of the publisher.

5.9.4.5 DDS_DataWriter* DDS_Publisher_create_datawriter (DDS_Publisher * *self*, DDS_Topic * *topic*, const struct DDS_DataWriterQos * *qos*, const struct DDS_DataWriterListener * *listener*, DDS_StatusMask *mask*)

<<*cert*>> Creates a DDS_DataWriter that will be attached and belong to the DDS_Publisher.

Precondition

If publisher is enabled, topic must have been enabled. Otherwise, this operation will fail and no DDS_DataWriter will be created.
The given DDS_Topic must have been created from the same participant as this publisher. If it was created from a different participant, this function will fail.

Parameters

self	<<*in*>> Cannot be NULL.
topic	<<*in*>> The DDS_Topic that the DDS_DataWriter will be associated with. Cannot be NULL.
qos	<<*in*>> QoS to be used for creating the new DDS_DataWriter. The special value DDS_DATAWRITER_QOS_DEFAULT can be used to indicate that the DDS_DataWriter should be created with the default DDS_DataWriterQos set in the DDS_Publisher.

Cannot be NULL.

Parameters

listener	<<*in*>> The listener of the DDS_DataWriter.
mask	<<*in*>>. Changes of communication status to be invoked on the listener.

Returns

A DDS_DataWriter of a derived class specific to the data type associated with the DDS_Topic or NULL if an error occurred.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

Specifying QoS on entities for information on setting QoS before entity creation
DDS_DataWriterQos for rules on consistency among QoS
DDS_DATAWRITER_QOS_DEFAULT

5.9.4.6 DDS_DataWriter* DDS_Publisher_lookup_datawriter (DDS_Publisher * *self*, const char * *topic_name*)

<<*cert*>> Retrieves the DDS_DataWriter for a specific DDS_Topic.

This returned DDS_DataWriter is either enabled or disabled.

Parameters

self	<<*in*>> Cannot be NULL.
topic_name	<<*in*>> Name of the DDS_Topic associated with the DDS_DataWriter that is to be looked up. Cannot be NULL.

Returns

A DDS_DataWriter that belongs to the DDS_Publisher attached to the DDS_Topic with `topic_name`. If no such DDS_DataWriter exists, this operation returns NULL.

If more than one DDS_DataWriter is attached to the DDS_Publisher with the same `topic_name`, then this operation may return any one of them.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.9.4.7 DDS_DomainParticipant* DDS_Publisher_get_participant (DDS_Publisher * *self*)

<<*cert*>> Returns the DDS_DomainParticipant to which the DDS_Publisher belongs.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

the DDS_DomainParticipant to which the DDS_Publisher belongs.

MT Safety:

. This operation is thread safe on a valid DDS_DataWriter.

5.9.5 Variable Documentation

5.9.5.1 struct DDS_DataWriterQos DDS_DATAWRITER_QOS_DEFAULT

<<*cert*>> Special value for creating DDS_DataWriter with default QoS

When used in DDS_Publisher_create_datawriter, this special value is used to indicate that the DDS_DataWriter should be created with the default DDS_DataWriter QoS.

See also

DDS_Publisher_create_datawriter

5.10 DataWriter

DDS_DataWriter entity and associated elements

Data Structures

- struct FooDataWriter

  <<interface>> <<generic>> User data type specific data writer.
- struct DDS_OfferedDeadlineMissedStatus

  <<cert>> DDS_OFFERED_DEADLINE_MISSED_STATUS
- struct DDS_LivelinessLostStatus

  <<cert>> DDS_LIVELINESS_LOST_STATUS
- struct DDS_OfferedIncompatibleQosStatus

  <<cert>> DDS_OFFERED_INCOMPATIBLE_QOS_STATUS
- struct DDS_PublicationMatchedStatus

  <<cert>> DDS_PUBLICATION_MATCHED_STATUS
- struct DDS_ReliableReaderActivityChangedStatus

  <<eXtension>> <<cert>> Describes the activity (i.e. are acknowledgements forthcoming) of reliable readers matched to a reliable writer.
- struct DDS_ReliableSampleUnacknowledgedStatus

  <<cert>> Status about unacknowledged samples.
- struct DDS_DataWriterQos

  <<cert>> QoS policies supported by a DDS_DataWriter entity.
- struct DDS_DataWriterListener

  <<interface>> <<cert>> DDS_Listener for writer status.

Macros

- #define DDS_OfferedDeadlineMissedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_LivelinessLostStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_OfferedIncompatibleQosStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_PublicationMatchedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_ReliableReaderActivityChangedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_DataWriterQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_DataWriterListener_INITIALIZER

  <<cert>> Initializer for new DDS_DataWriterListener.

Typedefs

- typedef struct DDS_DataWriterImpl DDS_DataWriter

  <<interface>> <<cert>> Allows an application to set the value of the data to be published under a given DDS_Topic.
- typedef void(* DDS_DataWriterListener_OfferedDeadlineMissedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_OfferedDeadlineMissedStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_offered_deadline_missed function.

- typedef void(* DDS_DataWriterListener_LivelinessLostCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_LivelinessLostStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_liveliness_lost function.
- typedef void(* DDS_DataWriterListener_OfferedIncompatibleQosCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_OfferedIncompatibleQosStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_offered_incompatible_qos function.
- typedef void(* DDS_DataWriterListener_PublicationMatchedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_PublicationMatchedStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_publication_matched function.
- typedef void(* DDS_DataWriterListener_ReliableReaderActivityChangedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_ReliableReaderActivityChangedStatus *status)

  <<eXtension>> <<cert>> A matched reliable reader has become active or become inactive.
- typedef void(* DDS_DataWriterListener_ReliableSampleUnacknowledgedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_ReliableSampleUnacknowledgedStatus *status)

  <<eXtension>> <<cert>> Unacknowledged samples are removed from the writer cache.

Functions

- FooDataWriter * FooDataWriter_narrow (DDS_DataWriter *writer)

  Narrow the given DDS_DataWriter pointer to a FooDataWriter pointer.
- DDS_DataWriter * FooDataWriter_as_datawriter (FooDataWriter *writer)

  Widen the given FooDataWriter pointer to a DDS_DataWriter pointer.
- DDS_InstanceHandle_t FooDataWriter_register_instance (FooDataWriter *self, const void *instance_data)

  <<cert>> Informs RTI Connext Micro that the application will be modifying a particular instance.
- DDS_InstanceHandle_t FooDataWriter_register_instance_w_timestamp (FooDataWriter *self, const void *instance_data, const struct DDS_Time_t *source_timestamp)

  <<cert>> Performs the same functions as register_instance except that the application provides the value for the `source_timestamp`.
- DDS_ReturnCode_t FooDataWriter_unregister_instance (FooDataWriter *self, const void *instance_data, const DDS_InstanceHandle_t *handle)

  <<cert>> Reverses the action of FooDataWriter_register_instance.
- DDS_ReturnCode_t FooDataWriter_unregister_instance_w_timestamp (FooDataWriter *self, const void *instance_data, const DDS_InstanceHandle_t *handle, const struct DDS_Time_t *source_timestamp)

  <<cert>> Performs the same function as FooDataWriter_unregister_instance except that it also provides the value for the `source_timestamp`.
- DDS_ReturnCode_t FooDataWriter_dispose (FooDataWriter *self, const void *instance_data, const DDS_InstanceHandle_t *handle)

  Requests the middleware to delete the data.
- DDS_ReturnCode_t FooDataWriter_dispose_w_timestamp (FooDataWriter *self, const void *instance_data, const DDS_InstanceHandle_t *handle, const struct DDS_Time_t *source_timestamp)

  <<cert>> Performs the same functions as dispose except that the application provides the value for the `source_timestamp` that is made available to DDS_DataReader objects by means of the `source_timestamp` attribute inside the DDS_SampleInfo.
- DDS_ReturnCode_t FooDataWriter_write (FooDataWriter *self, const void *instance_data, const DDS_InstanceHandle_t *handle)

  <<cert>> Modifies the value of a data instance.
- DDS_ReturnCode_t FooDataWriter_write_w_timestamp (FooDataWriter *self, const void *instance_data, const DDS_InstanceHandle_t *handle, const struct DDS_Time_t *source_timestamp)

  <<cert>> Performs the same function as FooDataWriter_write except that it also provides the value for the `source_timestamp`.
- DDS_ReturnCode_t DDS_DataWriterQos_initialize (struct DDS_DataWriterQos *self)

  <<cert>> Initializer for new QoS instances.

- DDS_ReturnCode_t DDS_DataWriterQos_copy (struct DDS_DataWriterQos *self, const struct DDS_DataWriterQos *source)

  Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_DataWriterQos_is_equal (const struct DDS_DataWriterQos *left, const struct DDS_DataWriterQos *right)

  Compare two DDS_DataWriterQos policies for equality.
- DDS_Entity * DDS_DataWriter_as_entity (DDS_DataWriter *dataWriter)

  <<*cert*>> *Access a DDS_DataWriter's supertype instance.*
- DDS_ReturnCode_t DDS_DataWriter_assert_liveliness (DDS_DataWriter *self)

  <<*cert*>> *This operation manually asserts the liveliness of this DDS_DataWriter.*
- DDS_Topic * DDS_DataWriter_get_topic (DDS_DataWriter *self)

  <<*cert*>> *This operation returns the DDS_Topic associated with the DDS_DataWriter.*
- DDS_Publisher * DDS_DataWriter_get_publisher (DDS_DataWriter *self)

  <<*cert*>> *This operation returns the DDS_Publisher to which the DDS_DataWriter belongs.*
- DDS_ReturnCode_t DDS_DataWriter_get_liveliness_lost_status (DDS_DataWriter *self, struct DDS_LivelinessLostStatus *status)

  <<*cert*>> *Accesses the DDS_LIVELINESS_LOST_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataWriter_get_offered_deadline_missed_status (DDS_DataWriter *self, struct DDS_OfferedDeadlineMissedStatus *status)

  <<*cert*>> *Accesses the DDS_OFFERED_DEADLINE_MISSED_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataWriter_get_offered_incompatible_qos_status (DDS_DataWriter *self, struct DDS_OfferedIncompatibleQosStatus *status)

  <<*cert*>> *Accesses the DDS_OFFERED_INCOMPATIBLE_QOS_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataWriter_get_publication_matched_status (DDS_DataWriter *self, struct DDS_PublicationMatchedStatus *status)

  <<*cert*>> *Accesses the DDS_PUBLICATION_MATCHED_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataWriter_get_reliable_reader_activity_changed_status (DDS_DataWriter *self, struct DDS_ReliableReaderActivityChangedStatus *status)

  <<*cert*>> *Get the reliable reader activity changed status for this writer.*
- DDS_ReturnCode_t FooDataWriter_get_loan (FooDataWriter *self, Foo **sample)

  Request a sample from a managed sample pool.
- DDS_ReturnCode_t FooDataWriter_discard_loan (FooDataWriter *self, Foo *sample)

  Return a Sample to a managed Sample pool.

5.10.1 Detailed Description

DDS_DataWriter entity and associated elements

5.10.2 Macro Definition Documentation

5.10.2.1 #define DDS_OfferedDeadlineMissedStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_OfferedDeadlineMissedStatus instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.10.2.2 #define DDS_LivelinessLostStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_LivelinessLostStatus instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.10.2.3 #define DDS_OfferedIncompatibleQosStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_OfferedIncompatibleQosStatus instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.10.2.4 #define DDS_PublicationMatchedStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_PublicationMatchedStatus instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.10.2.5 #define DDS_ReliableReaderActivityChangedStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_ReliableReaderActivityChangedStatus instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.10.2.6 #define DDS_DataWriterQos_INITIALIZER

<<*cert*>> Initializer for new QoS instances.

New DDS_DataWriterQos instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a new QoS structure is to initialize it on the stack at the time of its creation:

```
DDS_DataWriterQos myQos = DDS_DataWriterQos_INITIALIZER
    ;
```

5.10.2.7 #define DDS_DataWriterListener_INITIALIZER

<<*cert*>> Initializer for new DDS_DataWriterListener.

All the new DDS_DataWriterListener instances allocated in the stack should be initialized to this value.No memory is allocated.

```
DDS_DataWriterListener listener =
    DDS_DataWriterListener_INITIALIZER;
```

See also

DDS_DataWriterListener

5.10.3 Typedef Documentation

5.10.3.1 typedef struct DDS_DataWriterImpl DDS_DataWriter

<<*interface*>> <<*cert*>> Allows an application to set the value of the data to be published under a given DDS_Topic.

QoS:

DDS_DataWriterQos

Status:

DDS_LIVELINESS_LOST_STATUS, DDS_LivelinessLostStatus;
DDS_OFFERED_DEADLINE_MISSED_STATUS, DDS_OfferedDeadlineMissedStatus;
DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_OfferedIncompatibleQosStatus;
DDS_PUBLICATION_MATCHED_STATUS, DDS_PublicationMatchedStatus;

Listener:

DDS_DataWriterListener

A DDS_DataWriter is attached to exactly one DDS_Publisher, that acts as a factory for it.

A DDS_DataWriter is bound to exactly one DDS_Topic and therefore to exactly one data type. The DDS_Topic must exist prior to the DDS_DataWriter's creation.

DDS_DataWriter is an abstract class. It must be specialized for each particular application data-type. The additional methods or functions that must be defined in the auto-generated class for a hypothetical application type `Foo` are specified in the example type DDS_DataWriter.

The following operations may be called even if the DDS_DataWriter is not enabled. Other operations will fail with DDS_RETCODE_NOT_ENABLED if called on a disabled DDS_DataWriter:

- The base-class operations DDS_Entity_enable
- DDS_DataWriter_get_liveliness_lost_status DDS_DataWriter_get_offered_deadline_missed_status DDS_DataWriter_get_offered_incompatible_qos_status DDS_DataWriter_get_publication_matched_status

Several DDS_DataWriter may operate in different threads. If they share the same DDS_Publisher, the middleware guarantees that its operations are thread-safe.

5.10.3.2 typedef void(* DDS_DataWriterListener_OfferedDeadlineMissedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_OfferedDeadlineMissedStatus *status)

<<*cert*>> Prototype of a DDS_DataWriterListener on_offered_deadline_missed function.

5.10.3.3 typedef void(* DDS_DataWriterListener_LivelinessLostCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_LivelinessLostStatus *status)

<<*cert*>> Prototype of a DDS_DataWriterListener on_liveliness_lost function.

Parameters

listener_data	<<*out*>> Data associated with the listener when the listener is set
writer	<<*out*>> Locally created DDS_DataWriter that triggers the listener callback
status	<<*out*>> Current liveliness lost status of the locally created DDS_DataWriter

5.10.3.4 typedef void(* DDS_DataWriterListener_OfferedIncompatibleQosCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_OfferedIncompatibleQosStatus *status)

<<*cert*>> Prototype of a DDS_DataWriterListener on_offered_incompatible_qos function.

Parameters

listener_data	<<*out*>> Data associated with the listener when the listener is set
writer	<<*out*>> Locally created DDS_DataWriter that triggers the listener callback
status	<<*out*>> Current offered incompatible qos status of the locally created DDS_DataWriter

5.10.3.5 typedef void(* DDS_DataWriterListener_PublicationMatchedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_PublicationMatchedStatus *status)

<<*cert*>> Prototype of a DDS_DataWriterListener on_publication_matched function.

5.10.3.6 typedef void(* DDS_DataWriterListener_ReliableReaderActivityChangedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_ReliableReaderActivityChangedStatus *status)

<<*eXtension*>> <<*cert*>> A matched reliable reader has become active or become inactive.

Parameters

listener_data	<<*out*>> Data associated with the listener when the listener is set
writer	<<*out*>> Locally created DDS_DataWriter that triggers the listener callback
status	<<*out*>> Current reliable reader activity changed status of the locally created DDS_DataWriter

5.10.3.7 typedef void(* DDS_DataWriterListener_ReliableSampleUnacknowledgedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_ReliableSampleUnacknowledgedStatus *status)

<<*eXtension*>> <<*cert*>> Unacknowledged samples are removed from the writer cache.

Parameters

listener_data	<<*out*>> Data associated with the listener when the listener is set
writer	<<*out*>> Locally created DDS_DataWriter that triggers the listener callback
status	<<*out*>> Unacknowledged sample status for DDS_DataWriter

5.10.4 Function Documentation

5.10.4.1 FooDataWriter* FooDataWriter_narrow (DDS_DataWriter * *writer*)

Narrow the given DDS_DataWriter pointer to a FooDataWriter pointer.

Check if the given `writer` is of type FooDataWriter.

Parameters

writer	<<*in*>> Cannot be NULL.
writer	<<*in*>> Base-class DDS_DataWriter to be converted to the auto-generated class FooDataWriter that extends DDS_DataWriter.

Returns

FooDataWriter if `writer` is of type Foo. Return NULL otherwise.

5.10.4.2 DDS_DataWriter* FooDataWriter_as_datawriter (FooDataWriter * *writer*)

Widen the given FooDataWriter pointer to a DDS_DataWriter pointer.

Parameters

writer	<<*in*>> Cannot be NULL.
writer	<<*in*>> auto-generated FooDataWriter to be converted to the Base-class DDS_DataWriter.

Returns

DDS_DataWriter.

5.10.4.3 DDS_InstanceHandle_t FooDataWriter_register_instance (FooDataWriter * *self,* const void * *instance_data*)

<<cert>> Informs RTI Connext Micro that the application will be modifying a particular instance.

This operation is only useful for keyed data types. Using it for non-keyed types causes no effect and returns DDS_HANDLE_NIL. The operation takes as a parameter an instance (of which only the key value is examined) and returns a `handle` that can be used in successive write() or dispose() operations.

The operation gives RTI Connext Micro an opportunity to pre-configure itself to improve performance.

The use of this operation by an application is optional even for keyed types. If an instance has not been pre-registered, the application can use the special value DDS_HANDLE_NIL as the DDS_InstanceHandle_t paramater to the write or dispose operation and RTI Connext Micro will auto-register the instance.

For best performance, the operation should be invoked prior to calling any operation that modifies the instance, such as FooDataWriter_write, FooDataWriter_write_w_timestamp, FooDataWriter_dispose and FooDataWriter_dispose_w_timestamp and the handle used in conjunction with the data for those calls.

When this operation is used, RTI Connext Micro will automatically supply the value of the `source_timestamp` that is used.

This operation may fail and return DDS_HANDLE_NIL if DDS_ResourceLimitsQosPolicy::max_instances limit has been exceeded.

The operation is **idempotent**. If it is called for an already registered instance, it just returns the already allocated handle. This may be used to lookup and retrieve the handle allocated to a given instance.

This operation can only be called after DDS_DataWriter has been enabled. Otherwise, DDS_HANDLE_NIL will be returned.

Parameters

self	*<<in>>* Cannot be NULL.
instance_data	*<<in>>* The instance that should be registered. Of this instance, only the fields that represent the key are examined by the function. Cannot be NULL. .

Returns

For keyed data type, a handle that can be used in the calls that take a DDS_InstanceHandle_t, such as write, dispose, unregister_instance, or return DDS_HANDLE_NIL on failure. If the instance_data is of a data type that has no keys, this function always return DDS_HANDLE_NIL.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_unregister_instance

5.10.4.4 DDS_InstanceHandle_t FooDataWriter_register_instance_w_timestamp (FooDataWriter * *self,* const void * *instance_data,* const struct DDS_Time_t * *source_timestamp*)

<<cert>> Performs the same functions as register_instance except that the application provides the value for the `source_timestamp`.

The provided `source_timestamp` potentially affects the relative order in which readers observe events from multiple writers.

This operation may fail and return DDS_HANDLE_NIL if DDS_ResourceLimitsQosPolicy::max_instances limit has been exceeded.

This operation can only be called after DDS_DataWriter has been enabled. Otherwise, DDS_HANDLE_NIL will be returned.

Parameters

self	<<*in*>> Cannot be NULL.
instance_data	<<*in*>> The instance that should be registered. Of this instance, only the fields that represent the key are examined by the function. Cannot be NULL.
source_timestamp	<<*in*>> The timestamp value must be greater than or equal to the timestamp value used in the last writer operation (used in a *register*, *unregister*, *dispose*, or *write*, with either the automatically supplied timestamp or the application provided timestamp). This timestamp may potentially affect the order in which readers observe events from multiple writers. Cannot be NULL.

Returns

For keyed data type, return a handle that can be used in the calls that take a DDS_InstanceHandle_t, such as write, dispose, unregister_instance, or return DDS_HANDLE_NIL on failure. If the instance_data is of a data type that has no keys, this function always return DDS_HANDLE_NIL.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_unregister_instance

5.10.4.5 DDS_ReturnCode_t FooDataWriter_unregister_instance (FooDataWriter * *self*, const void * *instance_data*, const DDS_InstanceHandle_t * *handle*)

<<*cert*>> Reverses the action of FooDataWriter_register_instance.

This operation is useful only for keyed data types. Using it for non-keyed types causes no effect and reports no error. The operation takes as a parameter an instance (of which only the key value is examined) and a handle.

This operation should only be called on an instance that is currently registered. This includes instances that have been auto-registered by calling operations such as write or dispose as described in FooDataWriter_register_instance. Otherwise, this operation may fail with DDS_RETCODE_BAD_PARAMETER.

This only need be called just once per instance, regardless of how many times register_instance was called for that instance.

When this operation is used, RTI Connext Micro will automatically supply the value of the `source_timestamp` that is used.

This operation informs RTI Connext Micro that the DDS_DataWriter is no longer going to provide any information about the instance. This operation also indicates that RTI Connext Micro can locally remove all information regarding that instance. The application should not attempt to use the `handle` previously allocated to that instance after calling FooDataWriter_unregister_instance().

The special value DDS_HANDLE_NIL can be used for the parameter `handle`. This indicates that the identity of the instance should be automatically deduced from the `instance_data` (by means of the `key`).

If `handle` is any value other than DDS_HANDLE_NIL, then it must correspond to an instance that has been registered. If there is no correspondence, the operation will fail with DDS_RETCODE_BAD_PARAMETER.

RTI Connext Micro will not detect the error when the `handle` is any value other than DDS_HANDLE_NIL, corresponds to an instance that has been registered, but does not correspond to the instance deduced from the

instance_data (by means of the key). RTI Connext Micro will treat as if the unregister_instance() operation is for the instance as indicated by the handle.

If after a FooDataWriter_unregister_instance, the application wants to modify (FooDataWriter_write or FooDataWriter_dispose) an instance, it has to register it again, or else use the special handle value DDS_HANDLE_NIL.

This operation does not indicate that the instance is deleted (that is the purpose of FooDataWriter_dispose). The operation FooDataWriter_unregister_instance just indicates that the DDS_DataWriter no longer has anything to say about the instance. DDS_DataReader entities that are reading the instance may receive a sample with DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE for the instance, unless there are other DDS_DataWriter objects writing that same instance.

This operation can affect the ownership of the data instance (see OWNERSHIP). If the DDS_DataWriter was the exclusive owner of the instance, then calling unregister_instance() will relinquish that ownership.

If DDS_ReliabilityQosPolicy::kind is set to DDS_RELIABLE_RELIABILITY_QOS and the unregistration would overflow the resource limits of this writer or of a reader, this operation may block for up to DDS_ReliabilityQosPolicy::max_blocking_time; if this writer is still unable to unregister after that period, this function will fail with DDS_RETCODE_TIMEOUT.

Parameters

instance_data	<<*in*>> The instance that should be unregistered. If Foo *has* a key and instance_handle is DDS_HANDLE_NIL, only the fields that represent the key are examined by the function. Otherwise, instance_data is not used. If instance_data is used, it must represent an instance that has been registerd. Otherwise, this function may fail with DDS_RETCODE_BAD_PARAMETER . If Foo *has* a key, instance_data can be NULL only if handle is not DDS_HANDLE_NIL. Otherwise, this function will fail with DDS_RETCODE_BAD_PARAMETER.
handle	<<*in*>> represents the instance to be unregistered. If Foo *has* a key and handle is DDS_HANDLE_NIL, handle is not used and instance is deduced from instance_data. If Foo has no key, handle is not used. If handle is used, it must represent an instance that has been registered. Otherwise, this function may fail with DDS_RETCODE_BAD_PARAMETER. This function will fail with DDS_RETCODE_BAD_PARAMETER if handle is NULL. If Foo *has* a key, handle cannot be DDS_HANDLE_NIL if instance_data is NULL. Otherwise, this function will report the error DDS_RETCODE_BAD_PARAMETER.
self	<<*in*>> Cannot be NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_TIMEOUT or DDS_RETCODE_NOT_ENABLED

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_register_instance
FooDataWriter_unregister_instance_w_timestamp

5.10.4.6 DDS_ReturnCode_t FooDataWriter_unregister_instance_w_timestamp (FooDataWriter * *self*, const void * *instance_data*, const DDS_InstanceHandle_t * *handle*, const struct DDS_Time_t * *source_timestamp*)

<<*cert*>> Performs the same function as FooDataWriter_unregister_instance except that it also provides the value for the source_timestamp.

The provided source_timestamp potentially affects the relative order in which readers observe events from multiple writers.

The constraints on the values of the handle parameter and the corresponding error behavior are the same specified for the FooDataWriter_unregister_instance operation.

This operation may block and may time out (DDS_RETCODE_TIMEOUT) under the same circumtances described for the unregister_instance operation.

Parameters

self	<<*in*>> Cannot be NULL.
instance_data	<<*in*>> The instance that should be unregistered. If Foo *has* a key and `instance_handle` is DDS_HANDLE_NIL, only the fields that represent the key are examined by the function. Otherwise, `instance_data` is not used. If `instance_data` is used, it must represent an instance that has been registerd. Otherwise, this function may fail with DDS_RETCODE_BAD_PARAMETER. If Foo *has* a key, `instance_data` can be NULL only if `handle` is not DDS_HANDLE_NIL. Otherwise, this function will fail with DDS_RETCODE_BAD_PARAMETER.
handle	<<*in*>> represents the `instance` to be unregistered. If Foo *has* a key and `handle` is DDS_HANDLE_NIL, `handle` is not used and `instance` is deduced from `instance_data`. If Foo has no key, `handle` is not used. If `handle` is used, it must represent an instance that has been registered. Otherwise, this function may fail with DDS_RETCODE_BAD_PARAMETER. This function will fail with DDS_RETCODE_BAD_PARAMETER if `handle` is NULL. If Foo *has* a key, `handle` cannot be DDS_HANDLE_NIL if `instance_data` is NULL. Otherwise, this function will fail with DDS_RETCODE_BAD_PARAMETER.
source_timestamp	<<*in*>> The timestamp value must be greater than or equal to the timestamp value used in the last writer operation (used in a *register*, *unregister*, *dispose*, or *write*, with either the automatically supplied timestamp or the application provided timestamp). This timestamp may potentially affect the order in which readers observe events from multiple writers. Cannot be NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_TIMEOUT or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_register_instance
FooDataWriter_unregister_instance

5.10.4.7 DDS_ReturnCode_t FooDataWriter_dispose (FooDataWriter * *self*, const void * *instance_data*, const DDS_InstanceHandle_t * *handle*)

Requests the middleware to delete the data.

This operation is useful only for keyed data types. Using it for non-keyed types has no effect and reports no error.

The actual deletion is postponed until there is no more use for that data in the whole system.

Applications are made aware of the deletion by means of operations on the DDS_DataReader objects that already knew that instance. DDS_DataReader objects that didn't know the instance will never see it.

This operation does not modify the value of the instance. The `instance_data` parameter is passed just for the purposes of identifying the instance.

When this operation is used, RTI Connext Micro will automatically supply the value of the `source_timestamp` that is made available to DDS_DataReader objects by means of the `source_timestamp` attribute inside the DDS_SampleInfo.

The constraints on the values of the handle parameter and the corresponding error behavior are the same specified for the FooDataWriter_unregister_instance operation.

The special value DDS_HANDLE_NIL can be used for the parameter `instance_handle`. This indicates the identity of the instance should be automatically deduced from the `instance_data` (by means of the `key`).

If `handle` is any value other than DDS_HANDLE_NIL, then it must correspond to an instance that has been registered. If there is no correspondence, the operation will fail with DDS_RETCODE_BAD_PARAMETER.

RTI Connext Micro will not detect the error when the `handle` is any value other than DDS_HANDLE_NIL, corresponds to an instance that has been registered, but does not correspond to the instance deduced from the `instance_data` (by means of the `key`). RTI Connext Micro will treat as if the dispose() operation is for the instance as indicated by the `handle`.

This operation may block and time out (DDS_RETCODE_TIMEOUT) under the same circumtances described for FooDataWriter_write().

If there are no instance resources left, this operation may fail with DDS_RETCODE_OUT_OF_RESOURCES. Calling FooDataWriter_unregister_instance may help freeing up some resources.

Parameters

self	*<<in>>* Cannot be NULL.
instance_data	*<<in>>* The data to dispose. If Foo *has* a key and `instance_handle` is DDS_HANDLE_NIL, only the fields that represent the key are examined by the function. Otherwise, `instance_data` is not used. If Foo *has* a key, `instance_data` can be NULL only if `instance_handle` is not DDS_HANDLE_NIL. Otherwise, this function will fail with DDS_RETCODE_BAD_PARAMETER.
handle	*<<in>>* Either the handle returned by a previous call to FooDataWriter_register_instance, or else the special value DDS_HANDLE_NIL. If Foo *has* a key and `instance_handle` is DDS_HANDLE_NIL, `instance_handle` is not used and `instance` is deduced from `instance_data`. If Foo has no key, `instance_handle` is not used. If `handle` is used, it must represent a registered instance of type Foo. Otherwise, this function fail with DDS_RETCODE_BAD_PARAMETER. This function will fail with DDS_RETCODE_BAD_PARAMETER if `handle` is NULL. If Foo *has* a key, `instance_handle` cannot be DDS_HANDLE_NIL if `instance_data` is NULL. Otherwise, this function will fail with DDS_RETCODE_BAD_PARAMETER.

Returns

One of the Standard Return Codes, DDS_RETCODE_TIMEOUT, DDS_RETCODE_OUT_OF_RESOURCES or DDS_RETCODE_NOT_ENABLED.

See also

FooDataWriter_dispose_w_timestamp

5.10.4.8 DDS_ReturnCode_t FooDataWriter_dispose_w_timestamp (FooDataWriter * *self*, const void * *instance_data*, const DDS_InstanceHandle_t * *handle*, const struct DDS_Time_t * *source_timestamp*)

<<cert>> Performs the same functions as dispose except that the application provides the value for the `source_timestamp` that is made available to DDS_DataReader objects by means of the `source_timestamp` attribute inside the DDS_SampleInfo.

The constraints on the values of the `handle` parameter and the corresponding error behavior are the same specified for the FooDataWriter_dispose operation.

This operation may block and time out (DDS_RETCODE_TIMEOUT) under the same circumtances described for FooDataWriter_write.

If there are no instance resources left, this operation may fail with DDS_RETCODE_OUT_OF_RESOURCES. Calling FooDataWriter_unregister_instance may help freeing up some resources.

Parameters

self	<<*in*>> Cannot be NULL.
instance_data	<<*in*>> The data to dispose. If Foo *has* a key and `instance_handle` is DDS_HANDLE_NIL, only the fields that represent the key are examined by the function. Otherwise, `instance_data` is not used. If Foo *has* a key, `instance_data` can be NULL only if `instance_handle` is not DDS_HANDLE_NIL. Otherwise, this function will fail with DDS_RETCODE_BAD_PARAMETER.
handle	<<*in*>> Either the handle returned by a previous call to FooDataWriter_register_instance, or else the special value DDS_HANDLE_NIL. If Foo *has* a key and `instance_handle` is DDS_HANDLE_NIL, `instance_handle` is not used and `instance` is deduced from `instance_data`. If Foo has no key, `instance_handle` is not used. If `handle` is used, it must represent a registered instance of type Foo. Otherwise, this function may fail with DDS_RETCODE_BAD_PARAMETER This function will fail with DDS_RETCODE_BAD_PARAMETER if `handle` is NULL. If Foo *has* a key, `instance_handle` cannot be DDS_HANDLE_NIL if `instance_data` is NULL. Otherwise, this function will fail with DDS_RETCODE_BAD_PARAMETER.
source_timestamp	<<*in*>> The timestamp value must be greater than or equal to the timestamp value used in the last writer operation (used in a *register*, *unregister*, *dispose*, or *write*, with either the automatically supplied timestamp or the application provided timestamp). This timestamp may potentially affect the order in which readers observe events from multiple writers. This timestamp will be available to the DDS_DataReader objects by means of the `source_timestamp` attribute inside the DDS_SampleInfo. Cannot be NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_TIMEOUT, DDS_RETCODE_OUT_OF_RESOURCES or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_dispose

5.10.4.9 DDS_ReturnCode_t FooDataWriter_write (FooDataWriter * *self*, const void * *instance_data*, const DDS_InstanceHandle_t * *handle*)

<<*cert*>> Modifies the value of a data instance.

When this operation is used, RTI Connext Micro will automatically supply the value of the `source_timestamp` that is made available to DDS_DataReader objects by means of the `source_timestamp` attribute inside the DDS_SampleInfo. (Refer to DDS_SampleInfo details).

As a side effect, this operation asserts liveliness on the DDS_DataWriter itself, the DDS_Publisher and the DDS_DomainParticipant.

Note that the special value DDS_HANDLE_NIL can be used for the parameter `handle`. This indicates the identity of the instance should be automatically deduced from the `instance_data` (by means of the `key`).

If `handle` is any value other than DDS_HANDLE_NIL, then it must correspond to an instance that has been registered. If there is no correspondence, the operation will fail with DDS_RETCODE_BAD_PARAMETER.

RTI Connext Micro will not detect the error when the `handle` is any value other than DDS_HANDLE_NIL, corresponds to an instance that has been registered, but does not correspond to the instance deduced from the `instance_data` (by means of the `key`). RTI Connext Micro will treat as if the write() operation is for the instance as indicated by the `handle`.

Note that RCC does not support blocking write calls; this operation is always non-blocking.

If there are no instance resources left, this operation may fail with DDS_RETCODE_OUT_OF_RESOURCES. Calling FooDataWriter_unregister_instance may help freeing up some resources.

Parameters

self	<<*in*>> Cannot be NULL.
instance_data	<<*in*>> The data to write. This function will fail with DDS_RETCODE_BAD_PARAMETER if `instance_data` is NULL.
handle	<<*in*>> Either the handle returned by a previous call to FooDataWriter_register_instance, or else the special value DDS_HANDLE_NIL. If Foo *has* a key and `handle` is not DDS_HANDLE_NIL, `handle` must represent a registered instance of type Foo. Otherwise, this function may fail with DDS_RETCODE_BAD_PARAMETER. This function will fail with DDS_RETCODE_BAD_PARAMETER if `handle` is NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_TIMEOUT, DDS_RETCODE_OUT_OF_RESOURCES or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_DataReader
FooDataWriter_write_w_timestamp

5.10.4.10 DDS_ReturnCode_t FooDataWriter_write_w_timestamp (FooDataWriter * *self*, const void * *instance_data*, const DDS_InstanceHandle_t * *handle*, const struct DDS_Time_t * *source_timestamp*)

<<*cert*>> Performs the same function as FooDataWriter_write except that it also provides the value for the `source_timestamp`.

Explicitly provides the timestamp that will be available to the DDS_DataReader objects by means of the `source_timestamp` attribute inside the DDS_SampleInfo. (Refer to DDS_SampleInfo details)

The constraints on the values of the `handle` parameter and the corresponding error behavior are the same specified for the FooDataWriter_write operation.

This operation may block and time out (DDS_RETCODE_TIMEOUT) under the same circumtances described for FooDataWriter_write.

If there are no instance resources left, this operation may fail with DDS_RETCODE_OUT_OF_RESOURCES. Calling FooDataWriter_unregister_instance may help free up some resources.

This operation may fail with DDS_RETCODE_BAD_PARAMETER under the same circumstances described for the write operation.

Parameters

self	<<*in*>> Cannot be NULL.
instance_data	<<*in*>> The data to write. This function will fail with DDS_RETCODE_BAD_PARAMETER if `instance_data` is NULL.
handle	<<*in*>> Either the handle returned by a previous call to FooDataWriter_register_instance, or else the special value DDS_HANDLE_NIL. If Foo *has* a key and `handle` is not DDS_HANDLE_NIL, `handle` must represent a registered instance of type Foo. Otherwise, this function may fail with DDS_RETCODE_BAD_PARAMETER. This function will fail with DDS_RETCODE_BAD_PARAMETER if `handle` is NULL.

source_timestamp	<<*in*>> The timestamp value must be greater than or equal to the timestamp value used in the last writer operation (used in a *register*, *unregister*, *dispose*, or *write*, with either the automatically supplied timestamp or the application provided timestamp). This timestamp may potentially affect the order in which readers observe events from multiple writers. This timestamp will be available to the DDS_DataReader objects by means of the `source_timestamp` attribute inside the DDS_SampleInfo. Cannot be NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_TIMEOUT, DDS_RETCODE_OUT_OF_RESOURCES or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_write
DDS_DataReader

5.10.4.11 DDS_ReturnCode_t DDS_DataWriterQos_initialize (struct DDS_DataWriterQos * *self*)

<<*cert*>> Initializer for new QoS instances.

New DDS_DataWriterQos instances on heap should be initialized with this function before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This function does not allocate memory.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

One of the Standard Return Codes

5.10.4.12 DDS_ReturnCode_t DDS_DataWriterQos_copy (struct DDS_DataWriterQos * *self*, const struct DDS_DataWriterQos * *source*)

Copy the contents of the given QoS into this QoS.

DDS_DataWriterQos instances can use dynamic memory because of the sequences contained in some QoS policies. A shallow copy by assignment is therefore unsafe. This function performs a deep-copy, allocating memory if necessary.

Parameters

self	<<*in*>> Cannot be NULL.
source	<<*in*>> The DDS_DataWriterQos to copy from.

Returns

One of the Standard Return Codes

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

See also

DDS_DataWriterQos_INITIALIZER
DDS_DataWriterQos_initialize

5.10.4.13 DDS_Boolean DDS_DataWriterQos_is_equal (const struct DDS_DataWriterQos * *left*, const struct DDS_DataWriterQos * *right*)

Compare two DDS_DataWriterQos policies for equality.

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

See also

DDS_DataWriterQos_INITIALIZER
DDS_DataWriterQos_initialize

5.10.4.14 DDS_Entity* DDS_DataWriter_as_entity (DDS_DataWriter * *dataWriter*)

<<cert>> Access a DDS_DataWriter's supertype instance.

Parameters

dataWriter	*<<in>>* Cannot be NULL.

Returns

The DDS_Entity that is supertype instance of the DDS_DataWriter.

5.10.4.15 DDS_ReturnCode_t DDS_DataWriter_assert_liveliness (DDS_DataWriter * *self*)

<<cert>> This operation manually asserts the liveliness of this DDS_DataWriter.

This is used in combination with the LIVELINESS policy to indicate to RTI Connext Micro that the DDS_DataWriter

remains active.

You only need to use this operation if the LIVELINESS setting is DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS. Otherwise, it has no effect.

Note: writing data via the FooDataWriter_write or FooDataWriter_write_w_timestamp operation asserts liveliness on the DDS_DataWriter itself, and its DDS_DomainParticipant. Consequently the use of assert_liveliness() is only needed if the application is not writing data regularly.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

One of the Standard Return Codes or DDS_RETCODE_NOT_ENABLED

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_LivelinessQosPolicy

5.10.4.16 DDS_Topic* DDS_DataWriter_get_topic (DDS_DataWriter * *self*)

<<cert>> This operation returns the DDS_Topic associated with the DDS_DataWriter.

This is the same DDS_Topic that was used to create the DDS_DataWriter.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

DDS_Topic that was used to create the DDS_DataWriter.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.10.4.17 DDS_Publisher* DDS_DataWriter_get_publisher (DDS_DataWriter * *self*)

<<cert>> This operation returns the DDS_Publisher to which the DDS_DataWriter belongs.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

DDS_Publisher to which the DDS_DataWriter belongs.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.10.4.18 DDS_ReturnCode_t DDS_DataWriter_get_liveliness_lost_status (DDS_DataWriter * *self*, struct DDS_LivelinessLostStatus * *status*)

<<*cert*>> Accesses the DDS_LIVELINESS_LOST_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_LivelinessLostStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.10.4.19 DDS_ReturnCode_t DDS_DataWriter_get_offered_deadline_missed_status (DDS_DataWriter * *self*, struct DDS_OfferedDeadlineMissedStatus * *status*)

<<*cert*>> Accesses the DDS_OFFERED_DEADLINE_MISSED_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_OfferedDeadlineMissedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.10.4.20 DDS_ReturnCode_t DDS_DataWriter_get_offered_incompatible_qos_status (DDS_DataWriter * *self*, struct DDS_OfferedIncompatibleQosStatus * *status*)

<<*cert*>> Accesses the DDS_OFFERED_INCOMPATIBLE_QOS_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_OfferedIncompatibleQosStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.10.4.21 DDS_ReturnCode_t DDS_DataWriter_get_publication_matched_status (DDS_DataWriter * *self*, struct DDS_PublicationMatchedStatus * *status*)

<<*cert*>> Accesses the DDS_PUBLICATION_MATCHED_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_PublicationMatchedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.10.4.22 DDS_ReturnCode_t DDS_DataWriter_get_reliable_reader_activity_changed_status (DDS_DataWriter * *self*, struct DDS_ReliableReaderActivityChangedStatus * *status*)

<<*cert*>> Get the reliable reader activity changed status for this writer.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_ReliableReaderActivityChangedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.10.4.23 DDS_ReturnCode_t FooDataWriter_get_loan (FooDataWriter * *self*, Foo ** *sample*)

Request a sample from a managed sample pool.

This function is used to obtain a sample from a managed sample pool; you can then pass the loaned sample to FooDataWriter_write after setting the sample member values as you see fit.

A loaned sample should not be reused to write a new value after the first write operation. Instead, a new sample from FooDataWriter_get_loan should be used to write the new value.

A loaned sample that has not been written can be returned to the DataWriter's sample pool by calling FooDataWriter_discard_loan.

The FooDataWriter_get_loan function is supported while using Zero Copy transfer over shared memory.

Parameters

self	<<*in*>> Cannot be NULL.
self	<<*in*>> the DataWriter managing the sample pool.
sample	<<*out*>> a double pointer to the loaned sample.

Returns

DDS_RETCODE_OK if the loan was successfully obtained, DDS_RETCODE_OUT_OF_RESOURCES if no reusable sample is available in the sample pool, DDS_RETCODE_ERROR otherwise.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_discard_loan
FooDataWriter_write

5.10.4.24 DDS_ReturnCode_t FooDataWriter_discard_loan (FooDataWriter * *self,* Foo * *sample*)

Return a Sample to a managed Sample pool.

Return a Sample that had been previously loaned using FooDataWriter_get_loan to the pool of reusable samples managed by the DataWriter.

Parameters

self	<<*in*>> Cannot be NULL.
self	<<*in*>> the DataWriter managing the Sample pool.
sample	<<*in*>> a pointer to the Sample to be returned.

Returns

RTI_TRUE if the loan was successfully discarded, RTI_FALSE otherwise.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataWriter_get_loan

5.11 Subscription Module

Contains the DDS_Subscriber, and DDS_DataReader classes, and more generally, all that is needed on the subscription side.

Modules

- Subscriber

  DDS_Subscriber entity and associated elements
- DataReader

  DDS_DataReader entity and associated elements
- Data Sample

  DDS_SampleInfo, DDS_SampleStateKind, DDS_ViewStateKind, DDS_InstanceStateKind and associated elements

5.11.1 Detailed Description

Contains the DDS_Subscriber, and DDS_DataReader classes, and more generally, all that is needed on the subscription side. Defines the DDS subscription package.

This diagram defines the elements that are defined in the DDS specification, and their relationships to each other. Not all entities and functions shown in the diagram are supported by RTI Connext Micro. Please read the information on the individual types to determine whether a function is supported.

5.11.2 Access to data samples

Data is made available to the application by the following operations on DDS_DataReader objects: FooDataReader_read_next_sample, FooDataReader_take_next_sample

The general semantics of the `read()` operation is that the application only gets access to the corresponding data (i.e. a precise instance value); the data remains the responsibility of RTI Connext Micro and can be read again.

The semantics of the `take()` operations is that the application takes full responsibility for the data; that data will no longer be available locally to RTI Connext Micro. Consequently, it is possible to access the same information multiple times only if all previous accesses were read() operations, not take().

Each of these operations returns a collection of `Data` values and associated DDS_SampleInfo objects. Each data value represents an atom of data information (i.e., a value for one instance). This collection may contain samples related to the same or different instances (identified by the key). Multiple samples can refer to the same instance if the settings of the HISTORY QoS allow for it.

To return the memory back to the middleware, every read() or take() that retrieves a sequence of samples must be followed with a call to FooDataReader_return_loan.

See also

Interpretation of the SampleInfo

5.11.2.1 Data access patterns

The application accesses data by means of the operations `read` or `take` on the DDS_DataReader. These operations return an ordered collection of `DataSamples` consisting of a DDS_SampleInfo part and a `Data` part.

The way RTI Connext Micro builds the collection depends on QoS policies set on the DDS_DataReader and DDS_Subscriber, as well as the `source_timestamp` of the samples, and the parameters passed to the `read()` / `take()` operations, namely:

- the desired sample states (any combination of DDS_SampleStateKind)
- the desired view states (any combination of DDS_ViewStateKind)

- the desired instance states (any combination of DDS_InstanceStateKind)

The `read()` and `take()` operations are non-blocking and just deliver what is currently available that matches the specified states.

Once the data samples are available to the data readers, they can be read or taken by the application. The basic rule is that the application may do this in any order it wishes. This approach is very flexible and allows the application ultimate control.

5.12 Subscriber

DDS_Subscriber entity and associated elements

Data Structures

- struct DDS_SubscriberQos

  <<cert>> QoS policies supported by a DDS_Subscriber entity.
- struct DDS_SubscriberListener

  <<interface>> <<cert>> DDS_Listener for status about a subscriber.

Macros

- #define DDS_SubscriberQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_SubscriberListener_INITIALIZER

  <<eXtension>> <<cert>> Initializer for new DDS_SubscriberListener.

Typedefs

- typedef struct DDS_SubscriberImpl DDS_Subscriber

  <<interface>> A subscriber is the object responsible for actually receiving data from a subscription.
- typedef void(∗ DDS_SubscriberListener_DataOnReadersCallback)(void ∗listener_data, DDS_Subscriber ∗sub)

  Prototype of a DDS_SubscriberListener data_on_readers function.

Functions

- DDS_ReturnCode_t DDS_SubscriberQos_initialize (struct DDS_SubscriberQos ∗self)

  Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_SubscriberQos_copy (struct DDS_SubscriberQos ∗self, const struct DDS_SubscriberQos ∗source)

  Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_SubscriberQos_is_equal (const struct DDS_SubscriberQos ∗left, const struct DDS_SubscriberQos ∗right)

  Compare two DDS_SubscriberQos policies for equality.
- DDS_Entity ∗ DDS_Subscriber_as_entity (DDS_Subscriber ∗subscriber)

  <<cert>> Access a DDS_Subscriber's supertype instance.
- DDS_ReturnCode_t DDS_Subscriber_get_default_datareader_qos (DDS_Subscriber ∗self, struct DDS_DataReaderQos ∗qos)

  Copies the default DDS_DataReaderQos values into the provided DDS_DataReaderQos instance.
- DDS_ReturnCode_t DDS_Subscriber_set_default_datareader_qos (DDS_Subscriber ∗self, const struct DDS_DataReaderQos ∗qos)

  Sets the default DDS_DataReaderQos values for this subscriber.
- DDS_DataReader ∗ DDS_Subscriber_create_datareader (DDS_Subscriber ∗self, DDS_TopicDescription ∗topic, const struct DDS_DataReaderQos ∗qos, const struct DDS_DataReaderListener ∗listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_DataReader that will be attached and belong to the DDS_Subscriber.
- DDS_DataReader ∗ DDS_Subscriber_lookup_datareader (DDS_Subscriber ∗self, const char ∗topic_name)

  <<cert>> Retrieves an existing DDS_DataReader.
- DDS_DomainParticipant ∗ DDS_Subscriber_get_participant (DDS_Subscriber ∗self)

  <<cert>> Returns the DDS_DomainParticipant to which the DDS_Subscriber belongs.

Variables

- struct DDS_DataReaderQos DDS_DATAREADER_QOS_DEFAULT

  <<cert>> Special value for creating data reader with default QoS.

5.12.1 Detailed Description

DDS_Subscriber entity and associated elements

5.12.2 Macro Definition Documentation

5.12.2.1 #define DDS_SubscriberQos_INITIALIZER

<<cert>> Initializer for new QoS instances.

New DDS_SubscriberQos instances stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a new QoS structure is to initialize it on the stack at the time of its creation:

```
DDS_SubscriberQos myQos = DDS_SubscriberQos_INITIALIZER
    ;
```

5.12.2.2 #define DDS_SubscriberListener_INITIALIZER

<<eXtension>> <<cert>> Initializer for new DDS_SubscriberListener.

All the new instances allocated in the stack should be initialized to this value.No memory is allocated.

5.12.3 Typedef Documentation

5.12.3.1 typedef struct DDS_SubscriberImpl DDS_Subscriber

<<interface>> A subscriber is the object responsible for actually receiving data from a subscription.

QoS:

DDS_SubscriberQos

Status:

DDS_DATA_ON_READERS_STATUS

Listener:

DDS_SubscriberListener

A subscriber acts on the behalf of one or several DDS_DataReader objects that are related to it. When it receives data (from the other parts of the system), it builds the list of concerned DDS_DataReader objects and then indicates to the application that data is available through its listener.

The application can access the list of concerned DDS_DataReader objects through the operation get_datareaders() and then access the data available through operations on the DDS_DataReader.

The following operations may be called even if the DDS_Subscriber is not enabled. Other operations will the value DDS_RETCODE_NOT_ENABLED if called on a disabled DDS_Subscriber:

- The base-class operations DDS_Entity_enable

- DDS_Subscriber_create_datareader

All operations except for the base-class operations enable() and create_datareader() may fail with DDS_RETCODE_NOT_ENABLED.

5.12.3.2 typedef void(* DDS_SubscriberListener_DataOnReadersCallback)(void *listener_data, DDS_Subscriber *sub)

Prototype of a DDS_SubscriberListener data_on_readers function.

5.12.4 Function Documentation

5.12.4.1 DDS_ReturnCode_t DDS_SubscriberQos_initialize (struct DDS_SubscriberQos * *self*)

Initializer for new QoS instances.

New DDS_SubscriberQos instances on the heap should be initialized with this function before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This function does not allocate memory.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_SubscriberQos_INITIALIZER

5.12.4.2 DDS_ReturnCode_t DDS_SubscriberQos_copy (struct DDS_SubscriberQos * *self*, const struct DDS_SubscriberQos * *source*)

Copy the contents of the given QoS into this QoS.

DDS_SubscriberQos instances can use dynamic memory because of the sequences contained in some QoS policies. A shallow copy by assignment is therefore unsafe. This function performs a deep-copy, allocating memory if necessary.

Parameters

self	<<*in*>> Cannot be NULL.
source	<<*in*>> The DDS_PublisherQos to copy from.

Returns

One of the Standard Return Codes

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.12.4.3 DDS_Boolean DDS_SubscriberQos_is_equal (const struct DDS_SubscriberQos * *left*, const struct DDS_SubscriberQos * *right*)

Compare two DDS_SubscriberQos policies for equality.

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

See also

DDS_SubscriberQos_INITIALIZER
DDS_SubscriberQos_initialize

5.12.4.4 DDS_Entity* DDS_Subscriber_as_entity (DDS_Subscriber * *subscriber*)

<<*cert*>> Access a DDS_Subscriber's supertype instance.

Parameters

subscriber	<<*in*>> Cannot be NULL.

Returns

The DDS_Entity that is supertype instance of the subscriber.

5.12.4.5 DDS_ReturnCode_t DDS_Subscriber_get_default_datareader_qos (DDS_Subscriber * *self*, struct DDS_DataReaderQos * *qos*)

Copies the default DDS_DataReaderQos values into the provided DDS_DataReaderQos instance.

The retreived `qos` will match the set of values specified on the last successful call to DDS_Subscriber_set_default_datareader_qos, or else, if the call was never made, the default values from is owning DDS_DomainParticipant.

This function may potentially allocate memory depending on the sequences contained in some QoS policies.

MT Safety:

UNSAFE. It is not safe to retrieve the default QoS value from a subscriber while another thread may be simultaneously calling DDS_Subscriber_get_default_datareader_qos or calling DDS_Subscriber_create_datareader with DDS_DATAREADER_QOS_DEFAULT as the `qos` parameter.

Parameters

self	<<*in*>> Cannot be NULL.
qos	<<*inout*>> DDS_DataReaderQos to be filled-up. Cannot be NULL.

Returns

One of the Standard Return Codes

See also

DDS_DATAREADER_QOS_DEFAULT
DDS_Subscriber_create_datareader

5.12.4.6 DDS_ReturnCode_t DDS_Subscriber_set_default_datareader_qos (DDS_Subscriber * *self*, const struct DDS_DataReaderQos * *qos*)

Sets the default DDS_DataReaderQos values for this subscriber.

This call causes the default values inherited from the owning DDS_DomainParticipant to be overridden.

This default value will be used for newly created DDS_DataReader if DDS_DATAREADER_QOS_DEFAULT is specified as the `qos` parameter when DDS_Subscriber_create_datareader is called.

Precondition

The specified QoS policies must be consistent, or else the operation will have no effect and fail with DDS_RETCODE_INCONSISTENT_POLICY

MT Safety:

UNSAFE. It is not safe to retrieve the default QoS value from a subscriber while another thread may be simultaneously calling DDS_Subscriber_get_default_datareader_qos or calling DDS_Subscriber_create_datareader with DDS_DATAREADER_QOS_DEFAULT as the `qos` parameter.

Parameters

self	<<*in*>> Cannot be NULL.
qos	<<*in*>> The default DDS_DataReaderQos to be set to. The special value DDS_DATAREADER_QOS_DEFAULT may be passed as `qos` to indicate that the default QoS should be reset back to the initial values the factory would have used if DDS_Subscriber_set_default_datareader_qos had never been called. Cannot be NULL.

Returns

One of the Standard Return Codes, or or DDS_RETCODE_INCONSISTENT_POLICY

5.12.4.7 DDS_DataReader* DDS_Subscriber_create_datareader (DDS_Subscriber * *self*, DDS_TopicDescription * *topic*, const struct DDS_DataReaderQos * *qos*, const struct DDS_DataReaderListener * *listener*, DDS_StatusMask *mask*)

<<*cert*>> Creates a DDS_DataReader that will be attached and belong to the DDS_Subscriber.

Precondition

If subscriber is enabled, the topic must be enabled. If it is not, this operation will fail and no DDS_DataReader will be created.
The given DDS_TopicDescription must have been created from the same participant as this subscriber. If it was created from a different participant, this function will return NULL.

Parameters

self	<<*in*>> Cannot be NULL.
topic	<<*in*>> The DDS_TopicDescription that the DDS_DataReader will be associated with. Cannot be NULL.
qos	<<*in*>> The qos of the DDS_DataReader. The special value DDS_DATAREADER_QOS_DEFAULT can be used to indicate that the DDS_DataReader should be created with the default DDS_DataReaderQos set in the DDS_Subscriber. Cannot be NULL.
listener	<<*in*>> The listener of the DDS_DataReader.
mask	<<*in*>>. Changes of communication status to be invoked on the listener.

Returns

A DDS_DataReader of a derived class specific to the data-type associated with the DDS_Topic or NULL if an error occurred.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

Specifying QoS on entities for information on setting QoS before entity creation
DDS_DataReaderQos for rules on consistency among QoS

5.12.4.8 DDS_DataReader* DDS_Subscriber_lookup_datareader (DDS_Subscriber * *self*, const char * *topic_name*)

<<*cert*>> Retrieves an existing DDS_DataReader.

Use this operation on the built-in DDS_Subscriber (Discovery) to access the built-in DDS_DataReader entities for the built-in topics.

The built-in DDS_DataReader is created when this operation is called on a built-in topic for the first time. The built-in DDS_DataReader is deleted automatically when the DDS_DomainParticipant is deleted.

To ensure that builtin DDS_DataReader entities receive all the discovery traffic, it is suggested that you lookup the builtin DDS_DataReader before the DDS_DomainParticipant is enabled. Looking up builtin DDS_DataReader may implicitly register builtin transports due to creation of DDS_DataReader. Therefore, if you are want to modify builtin transport properties, do so *before* using this operation.

Therefore the suggested sequence when looking up builtin DataReaders is:

- Create a disabled DDS_DomainParticipant.
- (optional) Modify builtin transport properties
- Call DDS_DomainParticipant_get_builtin_subscriber().
- Call lookup_datareader().
- Call enable() on the DomainParticipant.

Parameters

self	<<*in*>> Cannot be NULL.
topic_name	<<*in*>> Name of the DDS_TopicDescription that the retrieved DDS_DataReader is attached to. Cannot be NULL.

Returns

A DDS_DataReader that belongs to the DDS_Subscriber attached to the DDS_TopicDescription with `topic_name`. If no such DDS_DataReader exists, this operation returns NULL.

The returned DDS_DataReader may be enabled or disabled.

If more than one DDS_DataReader is attached to the DDS_Subscriber, this operation may return any one of them.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.12.4.9 DDS_DomainParticipant* DDS_Subscriber_get_participant (DDS_Subscriber * *self*)

<<cert>> Returns the DDS_DomainParticipant to which the DDS_Subscriber belongs.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

the DDS_DomainParticipant to which the DDS_Subscriber belongs.

MT Safety:

This operation is thread safe.

5.12.5 Variable Documentation

5.12.5.1 struct DDS_DataReaderQos DDS_DATAREADER_QOS_DEFAULT

<<cert>> Special value for creating data reader with default QoS.

When used in DDS_Subscriber_create_datareader, this special value is used to indicate that the DDS_DataReader should be created with the default DDS_DataReader QoS.

See also

DDS_Subscriber_create_datareader

5.13 DataReader

DDS_DataReader entity and associated elements

Data Structures

- struct FooDataReader

  <<interface>> <<generic>> User data type-specific data reader.
- struct DDS_RequestedDeadlineMissedStatus

  <<cert>> DDS_REQUESTED_DEADLINE_MISSED_STATUS
- struct DDS_LivelinessChangedStatus

  <<cert>> DDS_LIVELINESS_CHANGED_STATUS
- struct DDS_RequestedIncompatibleQosStatus

  <<cert>> DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS
- struct DDS_SampleRejectedStatus

  DDS_SAMPLE_REJECTED_STATUS
- struct DDS_DataReaderInstanceReplacedStatus

  <<eXtension>> <<cert>> DDS_INSTANCE_REPLACED_STATUS.
- struct DDS_SubscriptionMatchedStatus

  <<cert>> DDS_SUBSCRIPTION_MATCHED_STATUS
- struct DDS_SampleLostStatus

  <<cert>> DDS_SAMPLE_LOST_STATUS
- struct DDS_DataReaderQos

  <<cert>> QoS policies supported by a DDS_DataReader entity.
- struct DDS_DataReaderListener

  <<interface>> <<cert>> DDS_Listener for reader status.

Macros

- #define DDS_RequestedDeadlineMissedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_LivelinessChangedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_RequestedIncompatibleQosStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_SampleRejectedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_DataReaderInstanceReplacedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_SubscriptionMatchedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_SampleLostStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_DataReaderQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_DataReaderListener_INITIALIZER

  <<cert>> Initializer for new DDS_DataReaderListener.

Typedefs

- typedef struct DDS_DataReaderImpl DDS_DataReader

  <<interface>> <<cert>> *Allows the application to: (1) declare the data it wishes to receive (i.e. make a subscription) and (2) access the data received by the attached DDS_Subscriber.*
- typedef void(* DDS_DataReaderListener_RequestedDeadlineMissedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_RequestedDeadlineMissedStatus *status)

  <<cert>> *Prototype of a DDS_DataReaderListener requested_deadline_missed function.*
- typedef void(* DDS_DataReaderListener_LivelinessChangedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_LivelinessChangedStatus *status)

  <<cert>> *Prototype of a DDS_DataReaderListener liveliness_changed function.*
- typedef void(* DDS_DataReaderListener_RequestedIncompatibleQosCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_RequestedIncompatibleQosStatus *status)

  <<cert>> *Prototype of a DDS_DataReaderListener requested_incompatible_QoS function.*
- typedef void(* DDS_DataReaderListener_SampleRejectedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SampleRejectedStatus *status)

  <<cert>> *Prototype of a DDS_DataReaderListener sample_rejected function.*
- typedef void(* DDS_DataReaderListener_DataAvailableCallback)(void *listener_data, DDS_DataReader *reader)

  <<cert>> *Prototype of a DDS_DataReaderListener data_available function.*
- typedef void(* DDS_DataReaderListener_SubscriptionMatchedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SubscriptionMatchedStatus *status)

  <<cert>> *Prototype of a DDS_DataReaderListener subscription_match function.*
- typedef void(* DDS_DataReaderListener_SampleLostCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SampleLostStatus *status)

  <<cert>> *Prototype of a DDS_DataReaderListener subscription_lost function.*
- typedef void(* DDS_DataReaderListener_InstanceReplacedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_DataReaderInstanceReplacedStatus *status)

  <<cert>> *Prototype of a DDS_DataReaderListener instance_replaced function.*
- typedef DDS_Boolean(* DDS_DataReaderListener_BeforeSampleDeserializeCallback)(void *listener_data, DDS_DataReader *reader, struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_Boolean *dropped)

  <<cert>> *Prototype of a DDS_DataReaderListener before_sample_deserialize function.*
- typedef DDS_Boolean(* DDS_DataReaderListener_BeforeSampleCommitCallback)(void *listener_data, DDS_DataReader *reader, const void *const sample, DDS_Boolean *dropped)

  <<cert>> *Prototype of a DDS_DataReaderListener before_sample_commit function.*

Enumerations

- enum DDS_SampleRejectedStatusKind { DDS_NOT_REJECTED, DDS_REJECTED_BY_INSTANCES_LIMIT, DDS_REJECTED_BY_SAMPLES_LIMIT, DDS_REJECTED_BY_SAMPLES_PER_INSTANCE_LIMIT, DDS_REJECTED_BY_REMOTE_WRITERS_LIMIT, DDS_REJECTED_BY_SAMPLES_PER_REMOTE_WRITER_LIMIT }

  Kinds of reasons for rejecting a sample.
- enum DDS_SampleLostStatusKind { DDS_SAMPLE_LOST_NOT_LOST, DDS_SAMPLE_LOST_BY_DATAWRITER, DDS_SAMPLE_LOST_BY_MAX_SAMPLES_LIMIT, DDS_SAMPLE_LOST_BY_HISTORY_DEPTH_LIMIT, DDS_SAMPLE_LOST_BY_META_SAMPLE_LIMIT, DDS_SAMPLE_LOST_BY_NOT_READ_ON_CACHE_DELETION, DDS_SAMPLE_LOST_BY_MANAGEDPOOL_OVERWRITE }

  Kinds of reasons a sample was lost.

Functions

- FooDataReader * FooDataReader_narrow (DDS_DataReader *reader)

  Narrow the given DDS_DataReader pointer to a FooDataReader pointer.
- DDS_DataReader * FooDataReader_as_datareader (FooDataReader *reader)

  Widen the given FooDataReader pointer to a DDS_DataReader pointer.
- DDS_ReturnCode_t FooDataReader_read (FooDataReader *self, struct FooSeq *received_data, struct DDS_SampleInfoSeq *info_seq, DDS_Long max_samples, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  Access a collection of data samples from the DDS_DataReader.
- DDS_ReturnCode_t FooDataReader_take (FooDataReader *self, struct FooSeq *received_data, struct DDS_SampleInfoSeq *info_seq, DDS_Long max_samples, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  Access a collection of data-samples from the DDS_DataReader.
- DDS_ReturnCode_t FooDataReader_read_next_sample (FooDataReader *self, struct Foo *received_data, struct DDS_SampleInfo *sample_info)

  Copies the next not-previously-accessed data value from the DDS_DataReader.
- DDS_ReturnCode_t FooDataReader_take_next_sample (FooDataReader *self, struct Foo *received_data, struct DDS_SampleInfo *sample_info)

  Copies the next not-previously-accessed data value from the DDS_DataReader.
- DDS_ReturnCode_t FooDataReader_read_instance (FooDataReader *self, struct FooSeq *received_data, struct DDS_SampleInfoSeq *info_seq, DDS_Long max_samples, const DDS_InstanceHandle_t *a_handle, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  Access a collection of data samples from the DDS_DataReader.
- DDS_ReturnCode_t FooDataReader_take_instance (FooDataReader *self, struct FooSeq *received_data, struct DDS_SampleInfoSeq *info_seq, DDS_Long max_samples, const DDS_InstanceHandle_t *a_handle, DDS_SampleStateMask sample_states, DDS_ViewStateMask view_states, DDS_InstanceStateMask instance_states)

  Access a collection of data samples from the DDS_DataReader.
- DDS_ReturnCode_t FooDataReader_return_loan (FooDataReader *self, FooSeq *received_data, struct DDS_SampleInfoSeq *info_seq)

  Indicates to the DDS_DataReader that the application is done accessing the collection of `received_data` and `info_seq` obtained by some earlier invocation of read or take on the DDS_DataReader.
- DDS_InstanceHandle_t FooDataReader_lookup_instance (FooDataReader *self, const Foo *key_holder)

  Retrieve the instance `handle` that corresponds to an instance `key_holder`.
- DDS_ReturnCode_t DDS_DataReaderQos_initialize (struct DDS_DataReaderQos *self)

  Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_DataReaderQos_copy (struct DDS_DataReaderQos *self, const struct DDS_DataReaderQos *source)

  Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_DataReaderQos_is_equal (const struct DDS_DataReaderQos *left, const struct DDS_DataReaderQos *right)

  Compare two DDS_DataReaderQos policies for equality.
- DDS_Entity * DDS_DataReader_as_entity (DDS_DataReader *self)

  <<cert>> Accesses the DDS_DataReader's supertype instance.
- DDS_TopicDescription * DDS_DataReader_get_topicdescription (DDS_DataReader *self)

  <<cert>> Returns the DDS_TopicDescription associated with the DDS_DataReader.
- DDS_Subscriber * DDS_DataReader_get_subscriber (DDS_DataReader *self)

  <<cert>> Returns the DDS_Subscriber to which the DDS_DataReader belongs.
- DDS_ReturnCode_t DDS_DataReader_get_sample_rejected_status (DDS_DataReader *self, struct DDS_SampleRejectedStatus *status)

  <<cert>> Accesses the DDS_SAMPLE_REJECTED_STATUS communication status.

- DDS_ReturnCode_t DDS_DataReader_get_liveliness_changed_status (DDS_DataReader *self, struct DDS_LivelinessChangedStatus *status)

  <<cert>> Accesses the DDS_LIVELINESS_CHANGED_STATUS communication status.
- DDS_ReturnCode_t DDS_DataReader_get_requested_deadline_missed_status (DDS_DataReader *self, struct DDS_RequestedDeadlineMissedStatus *status)

  <<cert>> Accesses the DDS_REQUESTED_DEADLINE_MISSED_STATUS communication status.
- DDS_ReturnCode_t DDS_DataReader_get_requested_incompatible_qos_status (DDS_DataReader *self, struct DDS_RequestedIncompatibleQosStatus *status)

  <<cert>> Accesses the DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS communication status.
- DDS_ReturnCode_t DDS_DataReader_get_subscription_matched_status (DDS_DataReader *self, struct DDS_SubscriptionMatchedStatus *status)

  <<cert>> Accesses the DDS_SUBSCRIPTION_MATCHED_STATUS communication status.
- DDS_ReturnCode_t DDS_DataReader_get_sample_lost_status (DDS_DataReader *self, struct DDS_SampleLostStatus *status)

  <<cert>> Accesses the DDS_SAMPLE_LOST_STATUS communication status.
- DDS_ReturnCode_t DDS_DataReader_get_instance_replaced_status (DDS_DataReader *self, struct DDS_DataReaderInstanceReplacedStatus *status)

  <<cert>> Accesses the DDS_INSTANCE_REPLACED_STATUS communication status.
- DDS_ReturnCode_t FooDataReader_is_data_consistent (DDS_DataReader *self, DDS_Boolean *is_data_consistent, const Foo *sample, const struct DDS_SampleInfo *sample_info)

  Checks if the sample has been overwritten by the DataWriter.

5.13.1 Detailed Description

DDS_DataReader entity and associated elements

5.13.2 Macro Definition Documentation

5.13.2.1 #define DDS_RequestedDeadlineMissedStatus_INITIALIZER

<<cert>> Initializer for new status instances.

New DDS_RequestedDeadlineMissedStatus instances that are stored on the stack should be initialized with this value before they are passed to any methods. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.13.2.2 #define DDS_LivelinessChangedStatus_INITIALIZER

<<cert>> Initializer for new status instances.

New DDS_LivelinessChangedStatus instances that are stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.13.2.3 #define DDS_RequestedIncompatibleQosStatus_INITIALIZER

<<cert>> Initializer for new status instances.

New DDS_RequestedIncompatibleQosStatus instances that are stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.13.2.4 #define DDS_SampleRejectedStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_SampleRejectedStatus instances that are stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.13.2.5 #define DDS_DataReaderInstanceReplacedStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_DataReaderInstanceReplacedStatus instances that are stored on the stack should be initialized with this value before they are passed to any methods. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.13.2.6 #define DDS_SubscriptionMatchedStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_SubscriptionMatchedStatus instances that are stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.13.2.7 #define DDS_SampleLostStatus_INITIALIZER

<<*cert*>> Initializer for new status instances.

New DDS_SampleLostStatus instances that are stored on the stack should be initialized with this value before they are passed to any function. This step ensures that those fields that use dynamic memory are properly initialized. This does not allocate memory.

The simplest way to create a status structure is to initialize it on the stack at the time of its creation.

5.13.2.8 #define DDS_DataReaderQos_INITIALIZER

<<*cert*>> Initializer for new QoS instances.

New DDS_DataReaderQos instances that are stored in the stack should be initialized with this value before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized.

The simplest way to create a new QoS structure is to initialize it on the stack at the time of its creation:

```
DDS_DataReaderQos myQos = DDS_DataReaderQos_INITIALIZER
    ;
```

5.13.2.9 #define DDS_DataReaderListener_INITIALIZER

<<*cert*>> Initializer for new DDS_DataReaderListener.

All the new instances allocated in the stack should be initialized to this value.No memory is allocated.

5.13.3 Typedef Documentation

5.13.3.1 typedef struct DDS_DataReaderImpl DDS_DataReader

<<*interface*>> <<*cert*>> Allows the application to: (1) declare the data it wishes to receive (i.e. make a subscription) and (2) access the data received by the attached DDS_Subscriber.

QoS:

DDS_DataReaderQos

Status:

DDS_DATA_AVAILABLE_STATUS;
DDS_LIVELINESS_CHANGED_STATUS, DDS_LivelinessChangedStatus;
DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS, DDS_RequestedIncompatibleQosStatus;
DDS_SUBSCRIPTION_MATCHED_STATUS, DDS_SubscriptionMatchedStatus;
DDS_INSTANCE_REPLACED_STATUS, DDS_DataReaderInstanceReplacedStatus;

Listener:

DDS_DataReaderListener

A DDS_DataReader refers to exactly one DDS_TopicDescription (a DDS_Topic) that identifies the data to be read.

The subscription has a unique resulting type. The data-reader may give access to several instances of the resulting type, which can be distinguished from each other by their `key`.

The following operations may be called even if the DDS_DataReader is not enabled. Other operations will fail with the value DDS_RETCODE_NOT_ENABLED if called on a disabled DDS_DataReader:

- The base-class operations DDS_Entity_enable

5.13.3.2 typedef void(* DDS_DataReaderListener_RequestedDeadlineMissedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_RequestedDeadlineMissedStatus *status)

<<*cert*>> Prototype of a DDS_DataReaderListener requested_deadline_missed function.

5.13.3.3 typedef void(* DDS_DataReaderListener_LivelinessChangedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_LivelinessChangedStatus *status)

<<*cert*>> Prototype of a DDS_DataReaderListener liveliness_changed function.

5.13.3.4 typedef void(* DDS_DataReaderListener_RequestedIncompatibleQosCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_RequestedIncompatibleQosStatus *status)

<<*cert*>> Prototype of a DDS_DataReaderListener requested_incompatible_QoS function.

5.13.3.5 typedef void(* DDS_DataReaderListener_SampleRejectedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SampleRejectedStatus *status)

<<*cert*>> Prototype of a DDS_DataReaderListener sample_rejected function.

5.13.3.6 typedef void(* DDS_DataReaderListener_DataAvailableCallback)(void *listener_data, DDS_DataReader *reader)

<<*cert*>> Prototype of a DDS_DataReaderListener data_available function.

5.13.3.7 typedef void(* DDS_DataReaderListener_SubscriptionMatchedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SubscriptionMatchedStatus *status)

<<*cert*>> Prototype of a DDS_DataReaderListener subscription_match function.

5.13.3.8 typedef void(* DDS_DataReaderListener_SampleLostCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SampleLostStatus *status)

<<*cert*>> Prototype of a DDS_DataReaderListener subscription_lost function.

5.13.3.9 typedef void(* DDS_DataReaderListener_InstanceReplacedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_DataReaderInstanceReplacedStatus *status)

<<*cert*>> Prototype of a DDS_DataReaderListener instance_replaced function.

5.13.3.10 typedef DDS_Boolean(* DDS_DataReaderListener_BeforeSampleDeserializeCallback)(void *listener_data, DDS_DataReader *reader, struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_Boolean *dropped)

<<*cert*>> Prototype of a DDS_DataReaderListener before_sample_deserialize function.

5.13.3.11 typedef DDS_Boolean(* DDS_DataReaderListener_BeforeSampleCommitCallback)(void *listener_data, DDS_DataReader *reader, const void *const sample, DDS_Boolean *dropped)

<<*cert*>> Prototype of a DDS_DataReaderListener before_sample_commit function.

5.13.4 Enumeration Type Documentation

5.13.4.1 enum DDS_SampleRejectedStatusKind

Kinds of reasons for rejecting a sample.

Enumerator:

DDS_NOT_REJECTED Samples are never rejected.

See also

DDS_ResourceLimitsQosPolicy

DDS_REJECTED_BY_INSTANCES_LIMIT A resource limit on the number of instances was reached.

See also

DDS_ResourceLimitsQosPolicy

DDS_REJECTED_BY_SAMPLES_LIMIT A resource limit on the number of samples was reached.

See also

DDS_ResourceLimitsQosPolicy

DDS_REJECTED_BY_SAMPLES_PER_INSTANCE_LIMIT A resource limit on the number of samples per instance was reached.

See also

DDS_ResourceLimitsQosPolicy

DDS_REJECTED_BY_REMOTE_WRITERS_LIMIT A resource limit on the number of remote writers from which a DDS_DataReader may read was reached. This constant is an extension to the DDS standard.

See also

DDS_DataReaderResourceLimitsQosPolicy

DDS_REJECTED_BY_SAMPLES_PER_REMOTE_WRITER_LIMIT A resource limit on the number of samples from a given remote writer that a DDS_DataReader may store was reached. This constant is an extension to the DDS standard.

See also

DDS_DataReaderResourceLimitsQosPolicy

5.13.4.2 enum DDS_SampleLostStatusKind

Kinds of reasons a sample was lost.

Enumerator:

DDS_SAMPLE_LOST_NOT_LOST Sample was not lost, normal removal.

See also

DDS_ResourceLimitsQosPolicy

DDS_SAMPLE_LOST_BY_DATAWRITER A sample was lost by the DataWriter, it was never received by the DataReader.

See also

DDS_ResourceLimitsQosPolicy

DDS_SAMPLE_LOST_BY_MAX_SAMPLES_LIMIT A sample was received by the DataReader, but lost because the max_samples resource was exceeded and no sample could be freed to receive it.

See also

DDS_ResourceLimitsQosPolicy

DDS_SAMPLE_LOST_BY_HISTORY_DEPTH_LIMIT A sample was received by the DataReader but lost because the max depth was reached for the instance and no samples could be removed from the history. For example, if all the samples in the DataReader cache are loaned by the application, they cannot be removed.

See also

DDS_ResourceLimitsQosPolicy

DDS_SAMPLE_LOST_BY_META_SAMPLE_LIMIT A meta sample was received by the DataReader but lost because the meta sample for the instance was loaned by the application and could not be reused.

See also

DDS_ResourceLimitsQosPolicy

DDS_SAMPLE_LOST_BY_NOT_READ_ON_CACHE_DELETION The DataReader has samples in the cache that had not been seen by the application and were lost when the DataReader was deleted.

See also

DDS_ResourceLimitsQosPolicy

DDS_SAMPLE_LOST_BY_MANAGEDPOOL_OVERWRITE The DataReader has samples that may have been overwritten.

See also

DDS_ResourceLimitsQosPolicy

5.13.5 Function Documentation

5.13.5.1 FooDataReader* FooDataReader_narrow (DDS_DataReader * *reader*)

Narrow the given DDS_DataReader pointer to a FooDataReader pointer.

Parameters

reader	<<*in*>> Cannot be NULL.

5.13.5.2 DDS_DataReader* FooDataReader_as_datareader (FooDataReader * *reader*)

Widen the given FooDataReader pointer to a DDS_DataReader pointer.

Parameters

reader	<<*in*>> Cannot be NULL.

5.13.5.3 DDS_ReturnCode_t FooDataReader_read (FooDataReader * *self*, struct FooSeq * *received_data*, struct DDS_SampleInfoSeq * *info_seq*, DDS_Long *max_samples*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*)

Access a collection of data samples from the DDS_DataReader.

This operation offers the same functionality and API as FooDataReader_take except that the samples returned remain in the DDS_DataReader such that they can be retrieved again by means of a read or take operation.

Please refer to the documentation of FooDataReader_take() for details on the number of samples returned within the received_data and info_seq as well as the order in which the samples appear in these sequences.

The act of reading a sample changes its `sample_state` to DDS_READ_SAMPLE_STATE. If the sample belongs to the most recent generation of the instance, it will also set the `view_state` of the instance to be DDS_NOT_NEW_VIEW_STATE. It will not affect the `instance_state` of the instance.

Important: If the samples "returned" by this function are loaned from RTI Connext Micro (see FooDataReader_take for more information on memory loaning), it is important that their contents not be changed. Because the memory in which the data is stored belongs to the middleware, any modifications made to the data will be seen the next time the same samples are read or taken; the samples will no longer reflect the state that was received from the network.

Parameters

self	<<*in*>> Cannot be NULL.
received_data	<<*inout*>> User data type-specific FooSeq object where the received data samples will be returned. Must be a valid non-NULL FooSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
info_seq	<<*inout*>> A DDS_SampleInfoSeq object where the received sample info will be returned. Must be a valid non-NULL DDS_SampleInfoSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
max_samples	<<*in*>> The maximum number of samples to be returned. If the special value DDS_LENGTH_UNLIMITED is provided, as many samples will be returned as are available.
sample_states	<<*in*>> Data samples matching one of these `sample_states` are returned.
view_states	<<*in*>> Data samples matching one of these `view_state` are returned.
instance_states	<<*in*>> Data samples matching ones of these `instance_state` are returned.

Returns

One of the Standard Return Codes, DDS_RETCODE_PRECONDITION_NOT_MET, DDS_RETCODE_NO_DATA or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataReader_take,
DDS_LENGTH_UNLIMITED

5.13.5.4 DDS_ReturnCode_t FooDataReader_take (FooDataReader * *self*, struct FooSeq * *received_data*, struct DDS_SampleInfoSeq * *info_seq*, DDS_Long *max_samples*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*)

Access a collection of data-samples from the DDS_DataReader.

The operation will return the list of samples received by the DDS_DataReader since the last FooDataReader_take operation that match the specified DDS_SampleStateMask, DDS_ViewStateMask and DDS_InstanceStateMask.

- In the case where the DDS_HistoryQosPolicy::kind is DDS_KEEP_LAST_HISTORY_QOS (currently the only supported configuration), the call will return at most DDS_HistoryQosPolicy::depth samples per instance.

- The maximum number of samples returned is limited by DDS_ResourceLimitsQosPolicy::max_samples
- For multiple instances, the number of samples returned is additionally limited by the product (DDS_ResourceLimitsQosPolicy::max_samples_per_instance * DDS_ResourceLimitsQosPolicy::max_instances)

If the read or take succeeds and the number of samples returned has been limited (by means of a maximum limit, as listed above, or insufficient DDS_SampleInfo resources), the call will complete successfully and provide those samples the reader is able to return. The user may need to make additional calls, or return outstanding loaned buffers in the case of insuffificient resources, in order to access remaining samples.

Note that in the case where the DDS_Topic associated with the DDS_DataReader is bound to a data-type that has no key definition, then there will be at most one instance in the DDS_DataReader. So the per-sample limits will apply.

The act of *taking* a sample removes it from RTI Connext Micro so it cannot be read or taken again. If the sample belongs to the most recent generation of the instance, it will also set the `view_state` of the sample's instance to DDS_NOT_NEW_VIEW_STATE. It will not affect the `instance_state` of the sample's instance.

After FooDataReader_take completes, `received_data` and `info_seq` will be of the same length and contain the received data.

If the sequences are empty (maximum size equal 0) when the FooDataReader_take is called, the samples returned in the `received_data` and the corresponding `info_seq` are 'loaned' to the application from buffers provided by the DDS_DataReader. The application can use them as desired and has guaranteed exclusive access to them.

Once the application completes its use of the samples it must 'return the loan' to the DDS_DataReader by calling the FooDataReader_return_loan operation.

Note: While you must call FooDataReader_return_loan at some point, you do *not* have to do so before the next FooDataReader_take call. However, failure to return the loan will eventually deplete the DDS_DataReader of the buffers it needs to receive new samples and eventually samples will start to be lost. The total number of buffers available to the DDS_DataReader is specified by the DDS_ResourceLimitsQosPolicy.

If the sequences are not empty (maximum size not equal 0) when the FooDataReader_take is called, samples are copied to received_data and info_seq. The application will not need to call FooDataReader_return_loan.

The samples returned to the caller is a list where samples belonging to the same data instance are consecutive (i.e. instance access scope).

Samples belonging to the same instance will appear in the relative order in which there were received (i.e. reception timestamp destination order, where earlier samples are ahead of later samples).

If the DDS_DataReader has no samples that meet the constraints, the function will fail with DDS_RETCODE_NO_DATA.

In addition to the collection of samples, the read and take operations also use a collection of DDS_SampleInfo structures.

5.13.6 SEQUENCES USAGE IN TAKE AND READ

The initial (input) properties of the `received_data` and `info_seq` collections will determine the precise behavior of the read or take operation. For the purposes of this description, the collections are modeled as having these properties:

- whether the collection container owns the memory of the elements within (`owns`)
- the current-length (`len`, see FooSeq_get_length)
- the maximum length (`max_len`, see FooSeq_get_maximum)

The initial values of the `owns`, `len` and `max_len` properties for the `received_data` and `info_seq` collections govern the behavior of the read and take operations as specified by the following rules:

1. The values of `owns`, `len` and `max_len` properties for the two collections must be identical. Otherwise read/take will fail with DDS_RETCODE_PRECONDITION_NOT_MET.

2. On successful output, the values of `owns`, `len` and `max_len` will be the same for both collections.

3. If the initial `max_len==0`, then the `received_data` and `info_seq` collections will be filled with elements that are loaned by the DDS_DataReader. On output, `owns` will be FALSE, `len` will be set to the number of values returned, and `max_len` will be set to a value verifying `max_len >= len`. The use of this variant allows for zero-copy access to the data and the application will need to return the loan to the DDS_DataWriter using FooDataReader_return_loan.

4. If the initial `max_len>0` and `owns==FALSE`, then the read or take operation will fail with DDS_RETCODE_PRECONDITION_NOT_MET. This avoids the potential hard-to-detect memory leaks caused by an application forgetting to return the loan.

5. If initial `max_len>0` and `owns==TRUE`, then the read or take operation will copy the `received_data` values and DDS_SampleInfo values into the elements already inside the collections. On output, `owns` will be TRUE, `len` will be set to the number of values copied and `max_len` will remain unchanged. The use of this variant forces a copy but the application can control where the copy is placed and the application will not need to return the loan. The number of samples copied depends on the relative values of `max_len` and `max_samples`:

   - If `max_samples` == LENGTH_UNLIMITED, then at most `max_len` values will be copied. The use of this variant lets the application limit the number of samples returned to what the sequence can accommodate.
   - If `max_samples <= max_len`, then at most `max_samples` values will be copied. The use of this variant lets the application limit the number of samples returned to fewer that what the sequence can accommodate.
   - If `max_samples > max_len`, then the read or take operation will fail with DDS_RETCODE_PRECONDITION_NOT_MET. This avoids the potential confusion where the application expects to be able to access up to `max_samples`, but that number can never be returned, even if they are available in the DDS_DataReader, because the output sequence cannot accommodate them.

As described above, upon completion, the `received_data` and `info_seq` collections may contain elements loaned from the DDS_DataReader. If this is the case, the application will need to use FooDataReader_return_loan to return the loan once it is no longer using the `received_data` in the collection. When FooDataReader_return_loan completes, the collection will have `owns=FALSE` and `max_len=0`. The application can determine whether it is necessary to return the loan or not based on how the state of the collections when the read/take operation was called or by accessing the `owns` property. However, in many cases it may be simpler to always call FooDataReader_return_loan, as this operation is harmless (i.e., it leaves all elements unchanged) if the collection does not have a loan.

To avoid potential memory leaks, the implementation of the Foo and DDS_SampleInfo collections should disallow changing the length of a collection for which `owns==FALSE`. Furthermore, deleting a collection for which `owns==FALSE` should be considered an error.

On output, the collection of Foo values and the collection of DDS_SampleInfo structures are of the same length and are in a one-to-one correspondence. Each DDS_SampleInfo provides information, such as the source_timestamp, the sample_state, view_state, and instance_state, etc., about the corresponding sample.

Some elements in the returned collection may not have valid data. If the instance_state in the DDS_SampleInfo is DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE or DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE, then the last sample for that instance in the collection (that is, the one whose DDS_SampleInfo has sample_rank==0) does not contain valid data.

Samples that contain no data do not count towards the limits imposed by the DDS_ResourceLimitsQosPolicy. The act of reading/taking a sample sets its sample_state to DDS_READ_SAMPLE_STATE.

If the sample belongs to the most recent generation of the instance, it will also set the view_state of the instance to DDS_NOT_NEW_VIEW_STATE. It will not affect the instance_state of the instance.

This operation must be provided on the specialized class that is generated for the particular application data-type that is being read (Foo). If the DDS_DataReader has no samples that meet the constraints, the operations fails with DDS_RETCODE_NO_DATA.

Parameters

received_data	<<*inout*>> User data type-specific FooSeq object where the received data samples will be returned. Must be a valid non-NULL FooSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
self	<<*in*>> Cannot be NULL.
info_seq	<<*inout*>> A DDS_SampleInfoSeq object where the received sample info will be returned. Must be a valid non-NULL DDS_SampleInfoSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
max_samples	<<*in*>> The maximum number of samples to be returned. If the special value DDS_LENGTH_UNLIMITED is provided, as many samples will be returned as are available.
sample_states	<<*in*>> Data samples matching one of these `sample_states` are returned.
view_states	<<*in*>> Data samples matching one of these `view_state` are returned.
instance_states	<<*in*>> Data samples matching one of these `instance_state` are returned.

Returns

One of the Standard Return Codes, DDS_RETCODE_PRECONDITION_NOT_MET, DDS_RETCODE_NO_DATA or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataReader_read
DDS_LENGTH_UNLIMITED

5.13.6.1 DDS_ReturnCode_t FooDataReader_read_next_sample (FooDataReader * *self*, struct Foo * *received_data*, struct DDS_SampleInfo * *sample_info*)

Copies the next not-previously-accessed data value from the DDS_DataReader.

This operation copies the next not-previously-accessed data value from the DDS_DataReader. This operation also copies the corresponding DDS_SampleInfo. The implied order among the samples stored in the DDS_DataReader is the same as for the FooDataReader_read operation.

The FooDataReader_read_next_sample operation is semantically equivalent to the FooDataReader_read operation, where the input data sequences has max_len=1, the sample_states=NOT_READ, the view_states=ANY_VIEW_STATE, and the instance_states=ANY_INSTANCE_STATE.

The FooDataReader_read_next_sample operation provides a simplified API to 'read' samples, avoiding the need for the application to manage sequences and specify states.

If there is no unread data in the DDS_DataReader, the operation will fail with DDS_RETCODE_NO_DATA and nothing is copied.

Parameters

self	<<*in*>> Cannot be NULL.
received_data	<<*inout*>> user data type-specific Foo object where the next received data sample will be returned. The received_data must have been fully allocated. Otherwise, this operation may fail. Must be a valid non-NULL Foo. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
sample_info	<<*inout*>> a DDS_SampleInfo object where the next received sample info will be returned. Must be a valid non-NULL DDS_SampleInfo. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_NO_DATA or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataReader_read

5.13.6.2 DDS_ReturnCode_t FooDataReader_take_next_sample (FooDataReader * *self*, struct Foo * *received_data*, struct DDS_SampleInfo * *sample_info*)

Copies the next not-previously-accessed data value from the DDS_DataReader.

This operation copies the next not-previously-accessed data value from the DDS_DataReader and 'removes' it from the DDS_DataReader so that it is no longer accessible. This operation also copies the corresponding DDS_SampleInfo. This operation is analogous to the FooDataReader_read_next_sample except for the fact that the sample is removed from the DDS_DataReader.

The FooDataReader_take_next_sample operation is semantically equivalent to the FooDataReader_take operation, where the input data sequences has max_len=1, the sample_states=NOT_READ, the view_states=ANY_VIEW_STATE, and the instance_states=ANY_INSTANCE_STATE.

The FooDataReader_read_next_sample operation provides a simplified API to 'take' samples, avoiding the need for the application to manage sequences and specify states.

If there is no unread data in the DDS_DataReader, the operation will fail with DDS_RETCODE_NO_DATA and nothing is copied.

Parameters

self	*<<in>>* Cannot be NULL.
received_data	*<<inout>>* user data type-specific Foo object where the next received data sample will be returned. The received_data must have been fully allocated. Otherwise, this operation may fail. Must be a valid non-NULL Foo. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
sample_info	*<<inout>>* a DDS_SampleInfo object where the next received sample info will be returned. Must be a valid non-NULL DDS_SampleInfo. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_NO_DATA or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataReader_take

5.13.6.3 DDS_ReturnCode_t FooDataReader_read_instance (FooDataReader * *self*, struct FooSeq * *received_data*, struct DDS_SampleInfoSeq * *info_seq*, DDS_Long *max_samples*, const DDS_InstanceHandle_t * *a_handle*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*)

Access a collection of data samples from the DDS_DataReader.

This operation accesses a collection of data values from the DDS_DataReader. The behavior is identical to FooDataReader_read, except that all samples returned belong to the single specified instance whose handle is `a_handle`.

Upon successful completion, the data collection will contain samples all belonging to the same instance. The corresponding DDS_SampleInfo verifies DDS_SampleInfo::instance_handle == `a_handle`.

The FooDataReader_read_instance operation is semantically equivalent to the FooDataReader_read operation, except in building the collection, the DDS_DataReader will check that the sample belongs to the specified instance and otherwise it will not place the sample in the returned collection.

The behavior of the FooDataReader_read_instance operation follows the same rules as the FooDataReader_read operation regarding the pre-conditions and post-conditions for the `received_data` and `sample_info`. Similar to the FooDataReader_read, the FooDataReader_read_instance operation may 'loan' elements to the output collections, which must then be returned by means of FooDataReader_return_loan.

Similar to the FooDataReader_read, this operation must be provided on the specialized class that is generated for the particular application data-type that is being taken.

If the DDS_DataReader has no samples that meet the constraints, the function will fail with DDS_RETCODE_NO_DATA.

This operation may fail with DDS_RETCODE_BAD_PARAMETER if the DDS_InstanceHandle_t `a_handle` does not correspond to an existing data-object known to the DDS_DataReader.

Parameters

self	*<<in>>* Cannot be NULL.
received_data	*<<inout>>* user data type-specific FooSeq object where the received data samples will be returned. Must be a valid non-NULL FooSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
info_seq	*<<inout>>* a DDS_SampleInfoSeq object where the received sample info will be returned. Must be a valid non-NULL DDS_SampleInfoSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
max_samples	*<<in>>* The maximum number of samples to be returned. If the special value DDS_LENGTH_UNLIMITED is provided, as many samples will be returned as are available.
a_handle	*<<in>>* The specified instance to return samples for. Must be a valid non-NULL DDS_InstanceHandle_t. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL The function will fail with DDS_RETCODE_BAD_PARAMETER if the `handle` does not correspond to an existing data-object known to the DDS_DataReader.
sample_states	*<<in>>* data samples matching ones of these `sample_states` are returned
view_states	*<<in>>* data samples matching ones of these `view_state` are returned
instance_states	*<<in>>* data samples matching ones of these `instance_state` are returned

Returns

One of the Standard Return Codes, DDS_RETCODE_PRECONDITION_NOT_MET, DDS_RETCODE_NO_DATA or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataReader_read
DDS_LENGTH_UNLIMITED

5.13.6.4 DDS_ReturnCode_t FooDataReader_take_instance (FooDataReader * *self*, struct FooSeq * *received_data*, struct DDS_SampleInfoSeq * *info_seq*, DDS_Long *max_samples*, const DDS_InstanceHandle_t * *a_handle*, DDS_SampleStateMask *sample_states*, DDS_ViewStateMask *view_states*, DDS_InstanceStateMask *instance_states*)

Access a collection of data samples from the DDS_DataReader.

This operation accesses a collection of data values from the DDS_DataReader. The behavior is identical to FooDataReader_take, except for that all samples returned belong to the single specified instance whose handle is `a_handle`.

The semantics are the same for the FooDataReader_take operation, except in building the collection, the DDS_DataReader will check that the sample belongs to the specified instance, and otherwise it will not place the sample in the returned collection.

The behavior of the FooDataReader_take_instance operation follows the same rules as the FooDataReader_read operation regarding the pre-conditions and post-conditions for the `received_data` and `sample_info`. Similar to the FooDataReader_read, the FooDataReader_take_instance operation may 'loan' elements to the output collections, which must then be returned by means of FooDataReader_return_loan.

Similar to the FooDataReader_read, this operation must be provided on the specialized class that is generated for the particular application data-type that is being taken.

If the DDS_DataReader has no samples that meet the constraints, the function fails with DDS_RETCODE_NO_DATA.

This operation may fail with DDS_RETCODE_BAD_PARAMETER if the DDS_InstanceHandle_t `a_handle` does not correspond to an existing data-object known to the DDS_DataReader.

Parameters

self	<<*in*>> Cannot be NULL.
received_data	<<*inout*>> user data type-specific FooSeq object where the received data samples will be returned. Must be a valid non-NULL FooSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
info_seq	<<*inout*>> a DDS_SampleInfoSeq object where the received sample info will be returned. Must be a valid non-NULL DDS_SampleInfoSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
max_samples	<<*in*>> The maximum number of samples to be returned. If the special value DDS_LENGTH_UNLIMITED is provided, as many samples will be returned as are available.
a_handle	<<*in*>> The specified instance to return samples for. Must be a valid non-NULL DDS_InstanceHandle_t. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL. The function will fail with DDS_RETCODE_BAD_PARAMETER if the `handle` does not correspond to an existing data-object known to the DDS_DataReader.
sample_states	<<*in*>> data samples matching ones of these `sample_states` are returned
view_states	<<*in*>> data samples matching ones of these `view_state` are returned
instance_states	<<*in*>> data samples matching ones of these `instance_state` are returned

Returns

One of the Standard Return Codes, DDS_RETCODE_PRECONDITION_NOT_MET, DDS_RETCODE_NO_DATA or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

FooDataReader_take
DDS_LENGTH_UNLIMITED

5.13.6.5 DDS_ReturnCode_t FooDataReader_return_loan (FooDataReader * *self,* FooSeq * *received_data,* struct DDS_SampleInfoSeq * *info_seq*)

Indicates to the DDS_DataReader that the application is done accessing the collection of `received_data` and `info_seq` obtained by some earlier invocation of read or take on the DDS_DataReader.

This operation indicates to the DDS_DataReader that the application is done accessing the collection of `received_data` and `info_seq` obtained by some earlier invocation of read or take on the DDS_DataReader.

The `received_data` and `info_seq` must belong to a single related "pair"; that is, they should correspond to a pair returned from a single call to read or take. The `received_data` and `info_seq` must also have been obtained from the same DDS_DataReader to which they are returned. If either of these conditions is not met, the operation will fail with DDS_RETCODE_PRECONDITION_NOT_MET.

The operation FooDataReader_return_loan allows implementations of the read and take operations to "loan" buffers from the DDS_DataReader to the application and in this manner provide "zerocopy" access to the data. During the loan, the DDS_DataReader will guarantee that the data and sample-information are not modified.

It is not necessary for an application to return the loans immediately after the read or take calls. However, as these buffers correspond to internal resources inside the DDS_DataReader, the application should not retain them indefinitely.

The use of FooDataReader_return_loan is only necessary if the read or take calls "loaned" buffers to the application. This only occurs if the `received_data` and `info_Seq` collections had `max_len=0` at the time read or take was called.

The application may also examine the "owns" property of the collection to determine where there is an outstanding loan. However, calling FooDataReader_return_loan on a collection that does not have a loan is safe and has no side effects.

If the collections had a loan, upon completion of FooDataReader_return_loan, the collections will have `max_len=0`.

Similar to read, this operation must be provided on the specialized class that is generated for the particular application data-type that is being taken.

Parameters

received_data	<<*in*>> user data type-specific FooSeq object where the received data samples was obtained from earlier invocation of read or take on the DDS_DataReader. Must be a valid non--NULL FooSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.
self	<<*in*>> Cannot be NULL.
info_seq	<<*in*>> a DDS_SampleInfoSeq object where the received sample info was obtained from earlier invocation of read or take on the DDS_DataReader. Must be a valid non-NULL DDS_-SampleInfoSeq. The function will fail with DDS_RETCODE_BAD_PARAMETER if it is NULL.

Returns

One of the Standard Return Codes, DDS_RETCODE_PRECONDITION_NOT_MET or DDS_RETCODE_NOT_ENABLED.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.6 DDS_InstanceHandle_t FooDataReader_lookup_instance (FooDataReader * *self,* const Foo * *key_holder*)

Retrieve the instance `handle` that corresponds to an instance `key_holder`.

Useful for keyed data types.

This operation takes as a parameter an instance and returns a handle that can be used in subsequent operations that accept an instance handle as an argument. The instance parameter is only used for the purpose of examining the fields that define the key. This operation does not register the instance in question. If the instance has not been previously registered, or if for any other reason the Service is unable to provide an instance handle, the Service will return the special value HANDLE_NIL.

Parameters

self	<<*in*>> Cannot be NULL.
key_holder	<<*in*>> a user data type specific key holder.

Returns

the instance handle associated with this instance. If Foo has no key, this function has no effect and returns DDS_HANDLE_NIL

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.7 DDS_ReturnCode_t DDS_DataReaderQos_initialize (struct DDS_DataReaderQos * *self*)

Initializer for new QoS instances.

New DDS_DataReaderQos instances on the heap should be initialized with this function before they are passed to any function. This step ensures that those contained QoS policies that use dynamic memory are properly initialized. This function does not allocate memory.

Parameters

self	<<*in*>> Cannot be NULL.

See also

DDS_DataReaderQos_INITIALIZER

5.13.6.8 DDS_ReturnCode_t DDS_DataReaderQos_copy (struct DDS_DataReaderQos * *self*, const struct DDS_DataReaderQos * *source*)

Copy the contents of the given QoS into this QoS.

DDS_DataReaderQos instances can use dynamic memory because of the sequences contained in some QoS policies. A shallow copy by assignment is therefore unsafe. This function performs a deep-copy, allocating memory if necessary.

Parameters

self	<<*in*>> Cannot be NULL.
source	<<*in*>>. Status to be copied from.

Returns

One of the Standard Return Codes

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

5.13.6.9 DDS_Boolean DDS_DataReaderQos_is_equal (const struct DDS_DataReaderQos * *left*, const struct DDS_DataReaderQos * *right*)

Compare two DDS_DataReaderQos policies for equality.

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

See also

DDS_DataReaderQos_INITIALIZER
DDS_DataReaderQos_initialize

5.13.6.10 DDS_Entity* DDS_DataReader_as_entity (DDS_DataReader * *self*)

<<*cert*>> Accesses the DDS_DataReader's supertype instance.

Parameters

self	<<*in*>> Cannot be NULL.

5.13.6.11 DDS_TopicDescription* DDS_DataReader_get_topicdescription (DDS_DataReader * *self*)

<<*cert*>> Returns the DDS_TopicDescription associated with the DDS_DataReader.

Returns that same DDS_TopicDescription that was used to create the DDS_DataReader.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

DDS_TopicDescription associated with the DDS_DataReader.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.12 DDS_Subscriber* DDS_DataReader_get_subscriber (DDS_DataReader * *self*)

<<*cert*>> Returns the DDS_Subscriber to which the DDS_DataReader belongs.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

DDS_Subscriber to which the DDS_DataReader belongs.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.13 DDS_ReturnCode_t DDS_DataReader_get_sample_rejected_status (DDS_DataReader * *self*, struct DDS_SampleRejectedStatus * *status*)

<<*cert*>> Accesses the DDS_SAMPLE_REJECTED_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_SampleRejectedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.14 DDS_ReturnCode_t DDS_DataReader_get_liveliness_changed_status (DDS_DataReader * *self*, struct DDS_LivelinessChangedStatus * *status*)

<<*cert*>> Accesses the DDS_LIVELINESS_CHANGED_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_LivelinessChangedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.15 DDS_ReturnCode_t DDS_DataReader_get_requested_deadline_missed_status (DDS_DataReader * *self*, struct DDS_RequestedDeadlineMissedStatus * *status*)

<<*cert*>> Accesses the DDS_REQUESTED_DEADLINE_MISSED_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_RequestedDeadlineMissedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.16 DDS_ReturnCode_t DDS_DataReader_get_requested_incompatible_qos_status (DDS_DataReader * *self*, struct DDS_RequestedIncompatibleQosStatus * *status*)

<<cert>> Accesses the DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS communication status.

Parameters

self	*<<in>>* Cannot be NULL.
status	*<<inout>>* DDS_RequestedIncompatibleQosStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.17 DDS_ReturnCode_t DDS_DataReader_get_subscription_matched_status (DDS_DataReader * *self*, struct DDS_SubscriptionMatchedStatus * *status*)

<<cert>> Accesses the DDS_SUBSCRIPTION_MATCHED_STATUS communication status.

Parameters

self	*<<in>>* Cannot be NULL.
status	*<<inout>>* DDS_SubscriptionMatchedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.18 DDS_ReturnCode_t DDS_DataReader_get_sample_lost_status (DDS_DataReader * *self*, struct DDS_SampleLostStatus * *status*)

<<cert>> Accesses the DDS_SAMPLE_LOST_STATUS communication status.

Parameters

self	*<<in>>* Cannot be NULL.
status	*<<inout>>* DDS_SampleLostStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.19 DDS_ReturnCode_t DDS_DataReader_get_instance_replaced_status (DDS_DataReader * *self*, struct DDS_DataReaderInstanceReplacedStatus * *status*)

<<*cert*>> Accesses the DDS_INSTANCE_REPLACED_STATUS communication status.

Parameters

self	<<*in*>> Cannot be NULL.
status	<<*inout*>> DDS_DataReaderInstanceReplacedStatus to be filled in. Cannot be NULL.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.13.6.20 DDS_ReturnCode_t FooDataReader_is_data_consistent (DDS_DataReader * *self*, DDS_Boolean * *is_data_consistent*, const Foo * *sample*, const struct DDS_SampleInfo * *sample_info*)

Checks if the sample has been overwritten by the DataWriter.

Due to the design of the Zero Copy transfer feature, this function returns with DDS_RETCODE_OK and a TRUE value returned through its output parameter. The function will only fail with DDS_RETCODE_BAD_PARAMETER if the output parameter is NULL.

Parameters

self	<<*in*>> Cannot be NULL.
is_data_consistent	<<*out*>> Output parameter for holding the consistency of the sample, which is always TRUE
sample	<<*in*>> Sample to be validated
sample_info	<<*in*>> DDS_SampleInfo object received with the sample

Returns

One of the Standard Return Codes, DDS_RETCODE_BAD_PARAMETER.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.14 Data Sample

DDS_SampleInfo, DDS_SampleStateKind, DDS_ViewStateKind, DDS_InstanceStateKind and associated elements

Modules

- Sample States

  DDS_SampleStateKind and associated elements
- View States

  DDS_ViewStateKind and associated elements
- Instance States

  DDS_InstanceStateKind and associated elements

Data Structures

- struct DDS_SampleInfo

  Information that accompanies each sample that is `read` *or* `taken`*.*
- struct DDS_SampleInfoSeq

  Declares IDL `sequence` < *DDS_SampleInfo* > *.*

5.14.1 Detailed Description

DDS_SampleInfo, DDS_SampleStateKind, DDS_ViewStateKind, DDS_InstanceStateKind and associated elements

5.15 Sample States

DDS_SampleStateKind and associated elements

Typedefs

- typedef DDS_UnsignedLong DDS_SampleStateMask

  A bit-mask (list) of sample states, i.e. DDS_SampleStateKind.

Enumerations

- enum DDS_SampleStateKind { DDS_READ_SAMPLE_STATE = 0x0001 << 0, DDS_NOT_READ_SAMPLE_STATE = 0x0001 << 1 }

  Indicates whether or not a sample has ever been read.

Variables

- const DDS_SampleStateMask DDS_ANY_SAMPLE_STATE

  Any sample state DDS_READ_SAMPLE_STATE | DDS_NOT_READ_SAMPLE_STATE.

5.15.1 Detailed Description

DDS_SampleStateKind and associated elements

5.15.2 Typedef Documentation

5.15.2.1 typedef DDS_UnsignedLong DDS_SampleStateMask

A bit-mask (list) of sample states, i.e. DDS_SampleStateKind.

5.15.3 Enumeration Type Documentation

5.15.3.1 enum DDS_SampleStateKind

Indicates whether or not a sample has ever been read.

For each sample received, the middleware internally maintains a sample_state relative to each DDS_DataReader. The sample state can be either:

- DDS_READ_SAMPLE_STATE indicates that the DDS_DataReader has already accessed that sample by means of a read or take operation.
- DDS_NOT_READ_SAMPLE_STATE indicates that the DDS_DataReader has not accessed that sample before.

The sample state will, in general, be different for each sample in the collection returned by read or take.

Enumerator:

DDS_READ_SAMPLE_STATE Sample has been read.

DDS_NOT_READ_SAMPLE_STATE Sample has not been read.

5.15.4 Variable Documentation

5.15.4.1 const DDS_SampleStateMask DDS_ANY_SAMPLE_STATE

Any sample state DDS_READ_SAMPLE_STATE | DDS_NOT_READ_SAMPLE_STATE.

5.16 View States

DDS_ViewStateKind and associated elements

Typedefs

- typedef DDS_UnsignedLong DDS_ViewStateMask

  A bit-mask (list) of view states, i.e. DDS_ViewStateKind.

Enumerations

- enum DDS_ViewStateKind { DDS_NEW_VIEW_STATE = 0x0001 << 0, DDS_NOT_NEW_VIEW_STATE = 0x0001 << 1 }

  Indicates whether or not an instance is new.

Variables

- const DDS_ViewStateMask DDS_ANY_VIEW_STATE

  Any view state DDS_NEW_VIEW_STATE | DDS_NOT_NEW_VIEW_STATE.

5.16.1 Detailed Description

DDS_ViewStateKind and associated elements

5.16.2 Typedef Documentation

5.16.2.1 typedef DDS_UnsignedLong DDS_ViewStateMask

A bit-mask (list) of view states, i.e. DDS_ViewStateKind.

5.16.3 Enumeration Type Documentation

5.16.3.1 enum DDS_ViewStateKind

Indicates whether or not an instance is new.

For each instance (identified by the key), the middleware internally maintains a view state relative to each DDS_DataReader. The view state can be either:

- DDS_NEW_VIEW_STATE indicates that either this is the first time that the DDS_DataReader has ever accessed samples of that instance, or else that the DDS_DataReader has accessed previous samples of the instance, but the instance has since been reborn (i.e. become not-alive and then alive again).

- DDS_NOT_NEW_VIEW_STATE indicates that the DDS_DataReader has already accessed samples of the same instance and that the instance has not been reborn since.

The view_state available in the DDS_SampleInfo is a snapshot of the view state of the instance relative to the DDS_DataReader used to access the samples at the time the collection was obtained (i.e. at the time read or take was called). The view_state is therefore the same for all samples in the returned collection that refer to the same instance.

Once an instance has been detected as not having any "live" writers and all the samples associated with the instance are "taken" from the DDS_DataReader, the middleware can reclaim all local resources regarding the instance. Future samples will be treated as "never seen."

Enumerator:

DDS_NEW_VIEW_STATE New instance. This latest generation of the instance has not previously been accessed.

DDS_NOT_NEW_VIEW_STATE Not a new instance. This latest generation of the instance has previously been accessed.

5.16.4 Variable Documentation

5.16.4.1 const DDS_ViewStateMask DDS_ANY_VIEW_STATE

Any view state DDS_NEW_VIEW_STATE | DDS_NOT_NEW_VIEW_STATE.

5.17 Instance States

DDS_InstanceStateKind and associated elements

Typedefs

- typedef DDS_UnsignedLong DDS_InstanceStateMask

  A bit-mask (list) of instance states, i.e. DDS_InstanceStateKind.

Enumerations

- enum DDS_InstanceStateKind { DDS_ALIVE_INSTANCE_STATE = 0x0001 << 0, DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE = 0x0001 << 1, DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE = 0x0001 << 2 }

  Indicates is the samples are from a live DDS_DataWriter or not.

Variables

- const DDS_InstanceStateMask DDS_ANY_INSTANCE_STATE

  Any instance state ALIVE_INSTANCE_STATE | NOT_ALIVE_DISPOSED_INSTANCE_STATE | NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.
- const DDS_InstanceStateMask DDS_NOT_ALIVE_INSTANCE_STATE

  Not alive instance state NOT_ALIVE_DISPOSED_INSTANCE_STATE | NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.

5.17.1 Detailed Description

DDS_InstanceStateKind and associated elements

5.17.2 Typedef Documentation

5.17.2.1 typedef DDS_UnsignedLong DDS_InstanceStateMask

A bit-mask (list) of instance states, i.e. DDS_InstanceStateKind.

5.17.3 Enumeration Type Documentation

5.17.3.1 enum DDS_InstanceStateKind

Indicates is the samples are from a live DDS_DataWriter or not.

For each instance, the middleware internally maintains an instance state. The instance state can be:

- DDS_ALIVE_INSTANCE_STATE indicates that (a) samples have been received for the instance, (b) there are live DDS_DataWriter entities writing the instance, and (c) the instance has not been explicitly disposed (or else more samples have been received after it was disposed).

- DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE indicates the instance was explicitly disposed by a DDS_DataWriter by means of the dispose operation.

- DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE indicates the instance has been declared as not-alive by the DDS_DataReader because it detected that there are no live DDS_DataWriter entities writing that instance.

The precise behavior events that cause the instance state to change depends on the setting of the OWNERSHIP QoS:

- If OWNERSHIP is set to DDS_EXCLUSIVE_OWNERSHIP_QOS, then the instance state becomes DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE only if the DDS_DataWriter that "owns" the instance explicitly disposes it. The instance state becomes DDS_ALIVE_INSTANCE_STATE again only if the DDS_DataWriter that owns the instance writes it.

- If OWNERSHIP is set to DDS_SHARED_OWNERSHIP_QOS, then the instance state becomes DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE if any DDS_DataWriter explicitly disposes the instance. The instance state becomes DDS_ALIVE_INSTANCE_STATE as soon as any DDS_DataWriter writes the instance again.

The instance state available in the DDS_SampleInfo is a snapshot of the instance state of the instance at the time the collection was obtained (i.e. at the time read or take was called). The instance state is therefore the same for all samples in the returned collection that refer to the same instance.

Enumerator:

DDS_ALIVE_INSTANCE_STATE Instance is currently in existence.

DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE Not alive disposed instance. The instance has been disposed by a DataWriter.

DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE Not alive no writers for instance. None of the DDS_DataWriter objects are currently alive (according to the LIVELINESS) are writing the instance.

5.17.4 Variable Documentation

5.17.4.1 const DDS_InstanceStateMask DDS_ANY_INSTANCE_STATE

Any instance state ALIVE_INSTANCE_STATE | NOT_ALIVE_DISPOSED_INSTANCE_STATE | NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.

5.17.4.2 const DDS_InstanceStateMask DDS_NOT_ALIVE_INSTANCE_STATE

Not alive instance state NOT_ALIVE_DISPOSED_INSTANCE_STATE | NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.

5.18 Infrastructure Module

Defines the DDS infrastructure package.

Modules

- Time Support

  Time and duration types and defines.
- GUID Support

  GUID type and defines.
- Sequence Number Support

  Sequence number type and defines.
- Return Codes

  Types of return codes.
- Status Kinds

  Kinds of communication status.
- QoS Policies

  Quality of Service (QoS) policies.
- Conditions and WaitSets

  DDS_Condition and DDS_WaitSet and related items.
- Entity Support

  DDS_Entity, DDS_Listener and related items.
- Sequence Support

  Defines sequence interface and primitive data types sequences.
- String Support

  String creation, cloning, assignment, and deletion.

5.18.1 Detailed Description

Defines the DDS infrastructure package. Defines the abstract classes and the interfaces that are refined by the other modules. Contains common definitions such as return codes, status values, and QoS policies.

This diagram defines the elements that are defined in the DDS specification, and their relationships to each other. Not all entities and functions shown in the diagram are supported by RTI Connext Micro. Please read the information on the individual types to determine whether a function is supported.

5.19 User-Defined Types

Generic Type entity and associated elements.

Data Structures

- struct NDDS_Type_PluginVersion

  Data type used to keep track of versioning information about a NDDS_Type_Plugin.
- struct NDDS_Type_Plugin

  <<interface>> Type plugin interface.

Typedefs

- typedef struct DDS_BuiltinTopicKey_t DDS_InstanceId_t

  The data structure by which RTI Connext Micro identifies keys in user data types.
- typedef struct NDDS_Type_PluginVersion NDDS_Type_PluginVersion

  Data type used to keep track of versioning information about a NDDS_Type_Plugin.
- typedef RTI_UINT32(* NDDS_Type_GetSerializedSizeFunc)(struct NDDS_Type_Plugin *plugin, RTI_UINT32 current_alignment, void *param)

  Prototype of the function to determine the size of a data sample in serialized form.
- typedef RTI_BOOL(* NDDS_Type_CreateSampleFunc)(struct NDDS_Type_Plugin *plugin, void **sample, void *param)

  Prototype of the data sample creation function.
- typedef RTI_BOOL(* NDDS_Type_DeleteSampleFunc)(struct NDDS_Type_Plugin *plugin, void *sample, void *param)

  Prototype of the data sample delete function.
- typedef RTI_BOOL(* NDDS_Type_CopySampleFunc)(struct NDDS_Type_Plugin *plugin, void *dst, const void *src, void *param)

  Prototype of the data sample copy function.
- typedef NDDS_TypePluginKeyKind(* NDDS_PluginHelper_GetKeyKindFunc)(struct NDDS_Type_Plugin *plugin, void *param)

  Prototype of the keyed data check function.
- typedef RTI_BOOL(* NDDS_PluginHelper_InstanceToKeyHashFunc)(struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_KeyHash_t *keyHash, const void *instance, void *param)

  Prototype of the function for converting an instance of data into a 16-byte key hash.
- typedef void *(* NDDS_PluginHelper_CreateTypedDataWriterFunc)(void *writer)

  Prototype of the function for creating a language-specific wrapper for a DDS_DataWriter created by the C core layer.
- typedef void(* NDDS_PluginHelper_DeleteTypedDataWriterFunc)(void *wrapper)

  Prototype of the function for deleting a language-specific wrapper of a DDS_DataReader created by the C core layer.
- typedef void *(* NDDS_PluginHelper_CreateTypedDataReaderFunc)(void *reader)

  Prototype of the function for creating a language-specific wrapper for a DDS_DataReader created by the C core layer.
- typedef void(* NDDS_PluginHelper_DeleteTypedDataReaderFunc)(void *wrapper)

  Prototype of the function for deleting a language-specific wrapper of a DDS_DataReader created by the C core layer.
- typedef DDS_ReturnCode_t(* NDDS_PluginHelper_ManagedPoolGetLoanFunc)(void *managed_pool, void **sample_out)

  Prototype of the function for getting a loan from a pool of managed samples.
- typedef RTI_BOOL(* NDDS_PluginHelper_ManagedPoolDiscardLoanFunc)(void *managed_pool, RTI_UINT32 sample_id)

  Prototype of the function for discarding a loan and returning a it back to a pool of managed samples.

- typedef RTI_BOOL(* NDDS_PluginHelper_CreateManagedPoolFunc)(struct NDDS_Type_Plugin *plugin, void *datawriter, void *param, void **managed_pool_out)

  Prototype of the function for creating a pool of samples.
- typedef void(* NDDS_PluginHelper_DeleteManagedPoolFunc)(void *managed_pool)

  Prototype of the function for deleting a pool of samples.
- typedef RTI_BOOL(* NDDS_PluginHelper_ManagedPoolGetSampleIDFunc)(void *managed_pool, const void *user_data, RTI_UINT32 *sample_id_out)

  Prototype of the function to getting an internal id associated with a sample loaned from the managed pool.
- typedef RTI_BOOL(* NDDS_PluginHelper_NeedsOpaqueSamples)(void *reader, void *param)

  Prototype of the function to be called by the DDS_DataReader to figure out if it needs to create opaque samples.

Enumerations

- enum NDDS_TypePluginKeyKind { NDDS_TYPEPLUGIN_NO_KEY , NDDS_TYPEPLUGIN_USER_KEY }

  Type key kind: either keyed or unkeyed.

Functions

- RTI_BOOL PluginHelper_instance_to_keyhash (struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_KeyHash_t *keyHash, const void *instance, void *param)

  Function to use a NDDS_Type_Plugin to create a 16-byte keyhash from an instance of data.
- NDDS_TypePluginKeyKind PluginHelper_get_key_kind (struct NDDS_Type_Plugin *plugin, void *param)

  Function to determine the type of key used by a NDDS_Type_Plugin's data type.

5.19.1 Detailed Description

Generic Type entity and associated elements.

5.19.2 Typedef Documentation

5.19.2.1 typedef struct DDS_BuiltinTopicKey_t DDS_InstanceId_t

The data structure by which RTI Connext Micro identifies keys in user data types.

RTI Connext Micro propagates and differentiates between instances of keyed types using DDS_InstanceId_t . Before a sample of a keyed type can be written, its key must be translated into one of these.

Users who do not edit the code generated by `rtiddsgen` do not need to deal with this type.

This type is structurally identical to the key type of the built-in topic types. A one-to-one mapping must exist from the key fields of a user data sample to the values of this type's fields.

See also

DDS_BuiltinTopicKey_t

5.19.2.2 typedef struct NDDS_Type_PluginVersion NDDS_Type_PluginVersion

Data type used to keep track of versioning information about a NDDS_Type_Plugin.

5.19.2.3 typedef RTI_UINT32(* NDDS_Type_GetSerializedSizeFunc)(struct NDDS_Type_Plugin *plugin, RTI_UINT32 current_alignment, void *param)

Prototype of the function to determine the size of a data sample in serialized form.

5.19.2.4 typedef RTI_BOOL(* NDDS_Type_CreateSampleFunc)(struct NDDS_Type_Plugin *plugin, void **sample, void *param)

Prototype of the data sample creation function.

5.19.2.5 typedef RTI_BOOL(* NDDS_Type_DeleteSampleFunc)(struct NDDS_Type_Plugin *plugin, void *sample, void *param)

Prototype of the data sample delete function.

5.19.2.6 typedef RTI_BOOL(* NDDS_Type_CopySampleFunc)(struct NDDS_Type_Plugin *plugin, void *dst, const void *src, void *param)

Prototype of the data sample copy function.

5.19.2.7 typedef NDDS_TypePluginKeyKind(* NDDS_PluginHelper_GetKeyKindFunc)(struct NDDS_Type_Plugin *plugin, void *param)

Prototype of the keyed data check function.

5.19.2.8 typedef RTI_BOOL(* NDDS_PluginHelper_InstanceToKeyHashFunc)(struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_KeyHash_t *keyHash, const void *instance, void *param)

Prototype of the function for converting an instance of data into a 16-byte key hash.

5.19.2.9 typedef void*(* NDDS_PluginHelper_CreateTypedDataWriterFunc)(void *writer)

Prototype of the function for creating a language-specific wrapper for a DDS_DataWriter created by the C core layer.

5.19.2.10 typedef void(* NDDS_PluginHelper_DeleteTypedDataWriterFunc)(void *wrapper)

Prototype of the function for deleting a language-specific wrapper of a DDS_DataReader created by the C core layer.

5.19.2.11 typedef void*(* NDDS_PluginHelper_CreateTypedDataReaderFunc)(void *reader)

Prototype of the function for creating a language-specific wrapper for a DDS_DataReader created by the C core layer.

5.19.2.12 typedef void(* NDDS_PluginHelper_DeleteTypedDataReaderFunc)(void *wrapper)

Prototype of the function for deleting a language-specific wrapper of a DDS_DataReader created by the C core layer.

5.19.2.13 typedef DDS_ReturnCode_t(* NDDS_PluginHelper_ManagedPoolGetLoanFunc)(void *managed_pool, void **sample_out)

Prototype of the function for getting a loan from a pool of managed samples.

5.19.2.14 typedef RTI_BOOL(* NDDS_PluginHelper_ManagedPoolDiscardLoanFunc)(void *managed_pool, RTI_UINT32 sample_id)

Prototype of the function for discarding a loan and returning a it back to a pool of managed samples.

5.19.2.15 typedef RTI_BOOL(* NDDS_PluginHelper_CreateManagedPoolFunc)(struct NDDS_Type_Plugin *plugin, void *datawriter, void *param, void **managed_pool_out)

Prototype of the function for creating a pool of samples.

5.19.2.16 typedef void(* NDDS_PluginHelper_DeleteManagedPoolFunc)(void *managed_pool)

Prototype of the function for deleting a pool of samples.

5.19.2.17 typedef RTI_BOOL(* NDDS_PluginHelper_ManagedPoolGetSampleIDFunc)(void *managed_pool, const void *user_data, RTI_UINT32 *sample_id_out)

Prototype of the function to getting an internal id associated with a sample loaned from the managed pool.

5.19.2.18 typedef RTI_BOOL(* NDDS_PluginHelper_NeedsOpaqueSamples)(void *reader, void *param)

Prototype of the function to be called by the DDS_DataReader to figure out if it needs to create opaque samples.

5.19.3 Enumeration Type Documentation

5.19.3.1 enum NDDS_TypePluginKeyKind

Type key kind: either keyed or unkeyed.

This type defines whether a user data type is keyed or unkeyed.

Enumerator:

NDDS_TYPEPLUGIN_NO_KEY Data is not keyed.

NDDS_TYPEPLUGIN_USER_KEY Data is keyed.

5.19.4 Function Documentation

5.19.4.1 RTI_BOOL PluginHelper_instance_to_keyhash (struct NDDS_Type_Plugin * *plugin,* struct CDR_Stream_t * *stream,* DDS_KeyHash_t * *keyHash,* const void * *instance,* void * *param*)

Function to use a NDDS_Type_Plugin to create a 16-byte keyhash from an instance of data.

5.19.4.2 NDDS_TypePluginKeyKind PluginHelper_get_key_kind (struct NDDS_Type_Plugin * *plugin,* void * *param*)

Function to determine the type of key used by a NDDS_Type_Plugin's data type.

5.20 Documentation Guide

This section describes the conventions used in the API documentation.

5.20.1 Unsupported Features

[**Not supported.**] This note means that the feature from the DDS specification is not supported in the current release.

5.20.2 API Naming Conventions

5.20.2.1 Structure & Class Names

RTI Connext Micro makes a distinction between *value types* and *interface types*. Value types are types such as primitives, enumerations, strings, and structures whose identity and equality are determined solely by explicit state. Interface types are those abstract opaque data types that conceptually have an identity apart from their explicit state. Examples include all of the DDS_Entity subtypes. Instances of value types are frequently transitory and are declared on the stack. Instances of interface types typically have longer lifecycles, are accessible by pointer only, and may be managed by a factory object.

Value type structures are made more explicit through the use of C's structure tag syntax. For example, a DDS_Duration_t object must be declared as being of type "struct DDS_Duration_t," not simply "DDS_Duration_t." Interface types, by contrast, are always of typedef'ed types; their underlying representations are opaque.

5.20.3 API Documentation Terms

In the API documentation, the term module refers to a logical grouping of documentation and elements in the API.

At this time, typedefs that occur in the API, such as DDS_ReturnCode_t do not show up in the compound list or indices. This is a known limitation in the generated HTML.

5.20.4 Stereotypes

Commonly used stereotypes in the API documentation include the following.

5.20.4.1 Extensions

- *<<eXtension>>*
  - An RTI Connext Micro product extension to the DDS standard specification.
  - The extension APIs complement the standard APIs specified by the OMG DDS specification. They are provided to improve product usability and enable access to product-specific features.

5.20.4.2 Types

- *<<interface>>*
  - Pure interface type with *no state*.
- *<<generic>>*
  - A *generic* type is a *skeleton* class written in terms of generic parameters. Type-specific instantiations of such types are conventionally referred to in this documentation in terms of the hypothetical type "Foo"; for example: FooSeq, FooDataType, FooDataWriter, and FooDataReader.
- *<<singleton>>*

  - Singleton class. There is a single instance of the class.
  - Generally acccessed via a `get_instance()` static function.

5.20.4.3 Method Parameters

- *<<in>>*
  - An *input* parameter. The input parameter is a value parameter and is expected to be a valid value. If the parameter is a pointer, e.g. a pointer to an *interface types*, it is also an *<<in>>*, even if the content being pointed to has changed on return. *<<in>>* means that the pointer argument is not modified, not that the object pointed to is not modified.

- *<<out>>*
  - An *output* parameter. Only possible with references. The input value is not used, but the value may be changed on return, e.g a field in a structure may have been updated or a pointer may point to a new object. In general, a parameter is designated with *<<out>>* only if the caller is expected to access the output directly.

- *<<inout>>*
  - An *input* and *output* parameter. Only possible with references. The input value must be valid, including the value of a pointer, and the parameter may have a new value on output; for example, the address of a pointer to a buffer is updated on return. In general, a parameter is designated with *<<inout>>* only if the caller is expected to access the output directly.

5.20.4.4 Cert API

- *<<cert>>*
  - Supported in safety certified version of RTI Connext Micro. Note, specific fields of *<<cert>>* data structures may not be supported and will be described in documentation as unsupported.

5.21 DDS API

RTI Connext Micro modules following the DDS module definitions.

Modules

- Domain Module

  Contains the DDS_DomainParticipant class that acts as an entrypoint of RTI Connext Micro and acts as a factory for many of the classes. The DDS_DomainParticipant also acts as a container for the other objects that make up RTI Connext Micro. Testing RTI Connext Micro command.
- Topic

  Contains the DDS_Topic, the DDS_TopicListener interface, and more generally, all that is needed by an application to define DDS_Topic objects and the data types that are associated with DDS_Topic objects.
- Publication Module

  Contains the DDS_Publisher, and DDS_DataWriter classes, and more generally, all that is needed on the publication side.
- Subscription Module

  Contains the DDS_Subscriber, and DDS_DataReader classes, and more generally, all that is needed on the subscription side.
- Infrastructure Module

  Defines the DDS infrastructure package.

5.21.1 Detailed Description

RTI Connext Micro modules following the DDS module definitions.

5.21.2 Overview

Information flows with the aid of the following constructs: DDS_Publisher and DDS_DataWriter on the sending side, DDS_Subscriber and DDS_DataReader on the receiving side.

- A DDS_Publisher is an object responsible for data distribution. It may publish data of different data types. A DDS_DataWriter acts as an accessor to a publisher - each DDS_DataWriter object is dedicated to one application data type. A DDS_DataWriter is the object the application must use to communicate to a publisher the existence and value of data-objects of a given type. When data-object values have been communicated to the publisher through the appropriate data-writer, it is the publisher's responsibility to perform the distribution (the publisher will do this according to its own QoS, or the QoS attached to the corresponding data-writer). A *publication* is defined by the association of a data-writer to a publisher. This association expresses the intent of the application to publish the data described by the data-writer in the context provided by the publisher.

- A DDS_Subscriber is an object responsible for receiving published data and making it available (according to the Subscriber's QoS) to the receiving application. It may receive and dispatch data of different specified types. To access the received data, the application must use a DDS_DataReader attached to the subscriber. Thus, a *subscription* is defined by the association of a data-reader with a subscriber. This association expresses the intent of the application to subscribe to the data described by the data-reader in the context provided by the subscriber.

DDS_Topic objects conceptually fit between publications and subscriptions. Publications must be known in such a way that subscriptions can refer to them unambiguously. A DDS_Topic is meant to fulfill that purpose: it associates a name (unique in the domain i.e. the set of applications that are communicating with each other), a data type, and QoS related to the data itself. In addition to the topic QoS, the QoS of the DDS_DataWriter associated with that Topic and the QoS of the DDS_Publisher associated to the DDS_DataWriter control the behavior on the publisher's side, while the corresponding DDS_Topic, DDS_DataReader and DDS_Subscriber QoS control the behavior on the subscriber's side.

When an application wishes to publish data of a given type, it must create a DDS_Publisher (or reuse an already created one) and a DDS_DataWriter with all the characteristics of the desired publication. Similarly, when an application wishes to receive data, it must create a DDS_Subscriber (or reuse an already created one) and a DDS_DataReader to define the subscription.

5.21.3 Conceptual Model

The overall conceptual model is shown below.

Notice that all the main communication objects (the specializations of Entity) follow unified patterns of:

- Supporting QoS (made up of several QosPolicy); QoS provides a generic mechanism for the application to control the behavior of the Service and tailor it to its needs. Each DDS_Entity supports its own specialized kind of QoS policies (see QoS Policies).

- Accepting a DDS_Listener; listeners provide a generic mechanism for the middleware to notify the application of relevant asynchronous events, such as arrival of data corresponding to a subscription, violation of a QoS setting, etc. Each DDS_Entity supports its own specialized kind of listener. Listeners are related to changes in status conditions (see Status Kinds).

  Note that only one Listener per entity is allowed (instead of a list of them). The reason for that choice is that this allows a much simpler (and, thus, more efficient) implementation as far as the middleware is concerned. Moreover, if it were required, the application could easily implement a listener that, when triggered, triggers in return attached 'sub-listeners'.

All DCPS entities are attached to a DDS_DomainParticipant. A domain participant represents the local membership of the application in a domain. A *domain* is a distributed concept that links all the applications able to communicate with each other. It represents a communication plane: only the publishers and the subscribers attached to the same domain may interact.

DDS_DomainEntity is an intermediate object whose only purpose is to state that a DomainParticipant cannot contain other domain participants.

At the DCPS level, data types represent information that is sent atomically. For performance reasons, only plain data structures are handled by this level.

By definition, a DDS_Topic corresponds to a single data type. However, several topics may refer to the same data type. Therefore, a DDS_Topic identifies data of a single type, ranging from one single instance to a whole collection of instances of that given type. This is shown below for the hypothetical data type `Foo`.

In case a set of instances is gathered under the same topic, different instances must be distinguishable. This is achieved by means of the values of some data fields that form the **key** to that data set. The *key description* (i.e., the list of data fields whose value forms the key) has to be indicated to the middleware. The rule is simple: *different data samples with the same key value represent successive values for the same instance, while different data samples with different key values represent different instances*. If no key is provided, the data set associated with the DDS_Topic is restricted to a *single instance*.

Topics need to be known by the middleware and potentially propagated. Topic objects are created using the create operations provided by DDS_DomainParticipant.

The interaction style is straightforward on the publisher's side: when the application decides that it wants to make data available for publication, it calls the appropriate operation on the related DDS_DataWriter (this, in turn, will trigger its DDS_Publisher).

On the subscriber's side however, there are more choices: relevant information may arrive when the application is busy doing something else or when the application is just waiting for that information. Therefore, depending on the way the application is designed, asynchronous notifications or synchronous access may be more appropriate. Both interaction modes are allowed, a DDS_Listener is used to provide a callback for asynchronous access, and polling allows for synchronous access.

5.21.4 Modules

DCPS consists of five modules:

- Infrastructure module defines the abstract classes and the interfaces that are refined by the other modules. It also provides support for the notification-based interaction style with the middleware.
- Domain module contains the DDS_DomainParticipant class that acts as an entrypoint of the Service and acts as a factory for many of the classes. The DDS_DomainParticipant also acts as a container for the other objects that make up the Service.
- Topic module contains the DDS_Topic class, and more generally, all that is needed by the application to define DDS_Topic objects and attach QoS policies to them.
- Publication module contains the DDS_Publisher and DDS_DataWriter classes, and more generally, all that is needed on the publication side.
- Subscription module contains the DDS_Subscriber, and DDS_DataReader, and more generally, all that is needed on the subscription side.

5.21.5 Overview

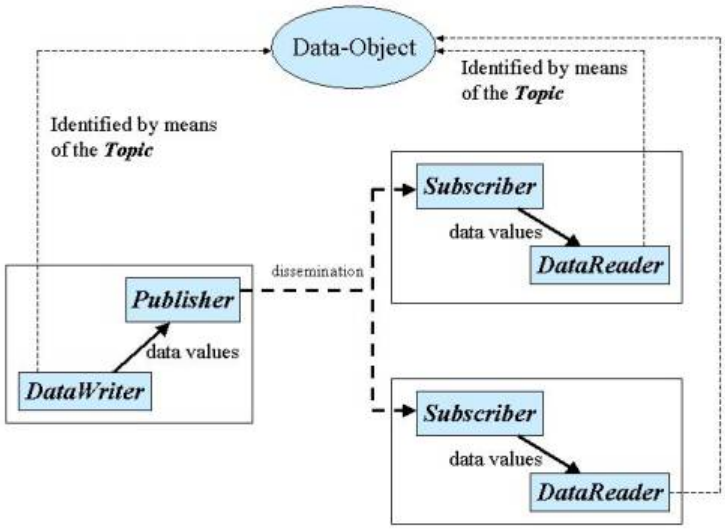

Figure 2: DDS Dissemination Overview

Information flows with the aid of the following constructs: DDS_Publisher and DDS_DataWriter on the sending side, DDS_Subscriber and DDS_DataReader on the receiving side.

- A DDS_Publisher is an object responsible for data distribution. It may publish data of different data types. A DDS_DataWriter acts as an accessor to a publisher - each DDS_DataWriter object is dedicated to one application data type. A DDS_DataWriter is the object the application must use to communicate to a publisher the existence and value of data-objects of a given type. When data-object values have been communicated to the publisher through the appropriate data-writer, it is the publisher's responsibility to perform the distribution (the publisher will do this according to its own QoS, or the QoS attached to the corresponding data-writer). A *publication* is defined by the association of a data-writer to a publisher. This association expresses the intent of the application to publish the data described by the data-writer in the context provided by the publisher.

- A DDS_Subscriber is an object responsible for receiving published data and making it available (according to the Subscriber's QoS) to the receiving application. It may receive and dispatch data of different specified types. To access the received data, the application must use a DDS_DataReader attached to the subscriber. Thus, a *subscription* is defined by the association of a data-reader with a subscriber. This association expresses the intent of the application to subscribe to the data described by the data-reader in the context provided by the subscriber.

DDS_Topic objects conceptually fit between publications and subscriptions. Publications must be known in such a way that subscriptions can refer to them unambiguously. A DDS_Topic is meant to fulfill that purpose: it associates a name (unique in the domain i.e. the set of applications that are communicating with each other), a data type, and QoS related to the data itself. In addition to the topic QoS, the QoS of the DDS_DataWriter associated with that Topic and the QoS of the DDS_Publisher associated to the DDS_DataWriter control the behavior on the publisher's side, while the corresponding DDS_Topic, DDS_DataReader and DDS_Subscriber QoS control the behavior on the subscriber's side.

When an application wishes to publish data of a given type, it must create a DDS_Publisher (or reuse an already created one) and a DDS_DataWriter with all the characteristics of the desired publication. Similarly, when an application wishes to receive data, it must create a DDS_Subscriber (or reuse an already created one) and a DDS_DataReader to define the subscription.

5.21.6 Conceptual Model

The overall conceptual model is shown below.

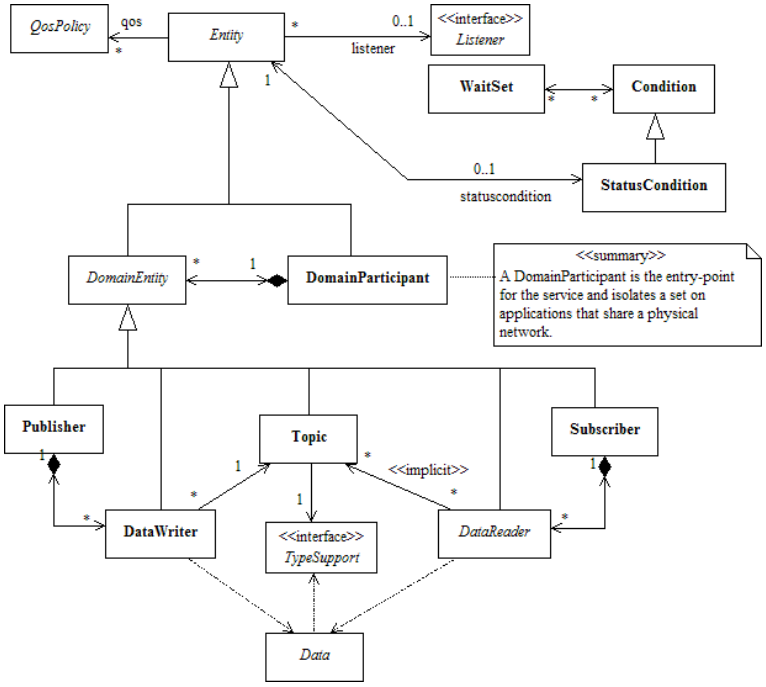

Figure 3: DCPS conceptual model

Notice that all the main communication objects (the specializations of Entity) follow unified patterns of:

- Supporting QoS (made up of several QosPolicy); QoS provides a generic mechanism for the application to control the behavior of the Service and tailor it to its needs. Each DDS_Entity supports its own specialized kind of QoS policies (see QoS Policies).

- Accepting a DDS_Listener; listeners provide a generic mechanism for the middleware to notify the application of relevant asynchronous events, such as arrival of data corresponding to a subscription, violation of a QoS

setting, etc. Each DDS_Entity supports its own specialized kind of listener. Listeners are related to changes in status conditions (see Status Kinds).

Note that only one Listener per entity is allowed (instead of a list of them). The reason for that choice is that this allows a much simpler (and, thus, more efficient) implementation as far as the middleware is concerned. Moreover, if it were required, the application could easily implement a listener that, when triggered, triggers in return attached 'sub-listeners'.

All DCPS entities are attached to a DDS_DomainParticipant. A domain participant represents the local membership of the application in a domain. A *domain* is a distributed concept that links all the applications able to communicate with each other. It represents a communication plane: only the publishers and the subscribers attached to the same domain may interact.

DDS_DomainEntity is an intermediate object whose only purpose is to state that a DomainParticipant cannot contain other domain participants.

At the DCPS level, data types represent information that is sent atomically. For performance reasons, only plain data structures are handled by this level.

By definition, a DDS_Topic corresponds to a single data type. However, several topics may refer to the same data type. Therefore, a DDS_Topic identifies data of a single type, ranging from one single instance to a whole collection of instances of that given type. This is shown below for the hypothetical data type `Foo`.

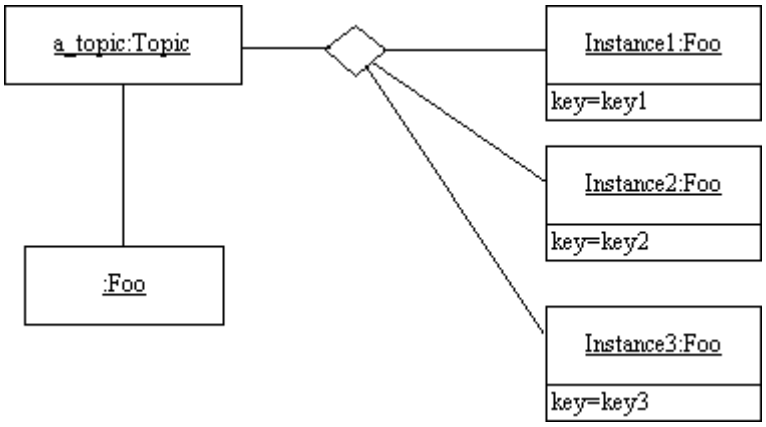

Figure 4: A collection of object instances, identified by their `keys`, can be disseminated on a topic.

In case a set of instances is gathered under the same topic, different instances must be distinguishable. This is achieved by means of the values of some data fields that form the **key** to that data set. The *key description* (i.e., the list of data fields whose value forms the key) has to be indicated to the middleware. The rule is simple: *different data samples with the same key value represent successive values for the same instance, while different data samples with different key values represent different instances*. If no key is provided, the data set associated with the DDS_Topic is restricted to a *single instance*.

Topics need to be known by the middleware and potentially propagated. Topic objects are created using the create operations provided by DDS_DomainParticipant.

The interaction style is straightforward on the publisher's side: when the application decides that it wants to make data available for publication, it calls the appropriate operation on the related DDS_DataWriter (this, in turn, will trigger its DDS_Publisher).

On the subscriber's side however, there are more choices: relevant information may arrive when the application is busy doing something else or when the application is just waiting for that information. Therefore, depending on the way the application is designed, asynchronous notifications or synchronous access may be more appropriate. Both interaction modes are allowed, a DDS_Listener is used to provide a callback for asynchronous access, and polling allows for synchronous access.

5.21.7 Modules

DCPS consists of five modules:

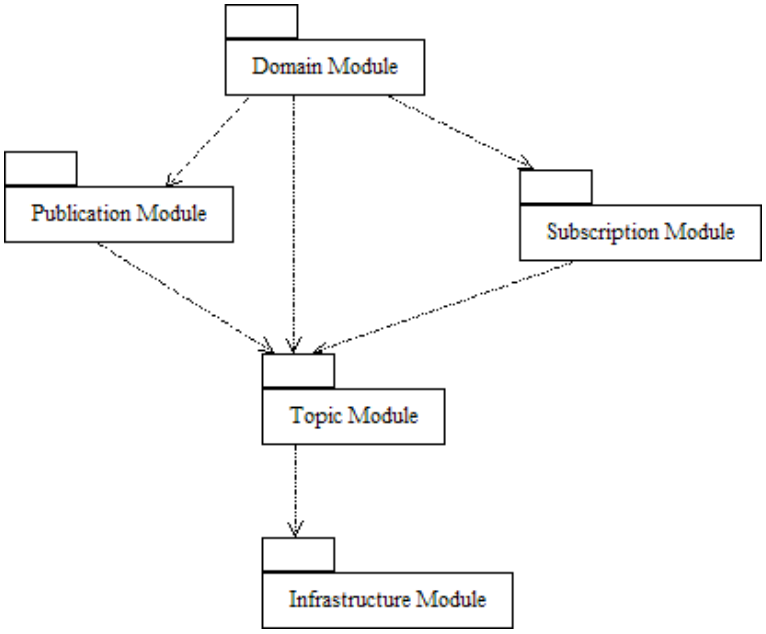

Figure 5: DCPS modules and their dependencies

- Infrastructure module defines the abstract classes and the interfaces that are refined by the other modules. It also provides support for the notification-based interaction style with the middleware.

- Domain module contains the DDS_DomainParticipant class that acts as an entrypoint of the Service and acts as a factory for many of the classes. The DDS_DomainParticipant also acts as a container for the other objects that make up the Service.

- Topic module contains the DDS_Topic class, and more generally, all that is needed by the application to define DDS_Topic objects and attach QoS policies to them.

- Publication module contains the DDS_Publisher and DDS_DataWriter classes, and more generally, all that is needed on the publication side.

- Subscription module contains the DDS_Subscriber, and DDS_DataReader, and more generally, all that is needed on the subscription side.

5.22 DPSE Static Discovery API

Data Structures

- struct DPSE_DiscoveryPluginProperty

  <<eXtension>> <<cert>> Properties for the Dynamic Participant/Static Endpoint (DPSE) discovery plugin. This includes all discovery timing properties for participant discovery.

Macros

- #define DPSE_DiscoveryPluginProperty_INITIALIZER

  Initializer for the DPSE_DiscoveryPluginProperty.

Functions

- DDS_ReturnCode_t DPSE_DiscoveryPluginProperty_initialize (struct DPSE_DiscoveryPluginProperty *self)

  Initialize a DiscoveryPluginProperty structure.
- DDS_ReturnCode_t DPSE_RemoteParticipant_assert (DDS_DomainParticipant *const participant, const char *rem_participant_name)

  <<cert>> Assert remote participant by passing in the name of that participant.
- DDS_ReturnCode_t DPSE_RemotePublication_assert (DDS_DomainParticipant *const participant, const char *const rem_participant_name, const struct DDS_PublicationBuiltinTopicData *const data, NDDS_TypePluginKeyKind key_kind)

  <<cert>> Assert remote publication by passing in the builtin data that describes the type, topic, and QoS of that remote publication.
- DDS_ReturnCode_t DPSE_RemoteSubscription_assert (DDS_DomainParticipant *const participant, const char *const rem_participant_name, const struct DDS_SubscriptionBuiltinTopicData *const data, NDDS_TypePluginKeyKind key_kind)

  <<cert>> Assert remote subscription by passing in the builtin data that describes the type, topic, and QoS of that remote subscription.
- struct RT_ComponentFactoryI * DPSE_DiscoveryFactory_get_interface (void)

  Gets the singleton instance of the DPSE discovery plugin factory. The singleton instance of the factory is used when creating a domain participant.

5.22.1 Detailed Description

<<eXtension>> <<cert>> The RTI Connext Micro support for DPSE discovery of remote entities. Please refer to the Discovery chapter in the User's Manual, and the Getting Started Guide, for an example of how to register and use the DPSE plugin.

5.22.2 Macro Definition Documentation

5.22.2.1 #define DPSE_DiscoveryPluginProperty_INITIALIZER

Value:

```
{ \
    RT_ComponentFactoryProperty_INITIALIZER,\
    {30,0},  \
    {100,0}, \
    5,       \
    {1,0},   \
    DDS_LENGTH_AUTO,\
    4 \
}
```

Initializer for the DPSE_DiscoveryPluginProperty.

5.22.3 Function Documentation

5.22.3.1 DDS_ReturnCode_t DPSE_DiscoveryPluginProperty_initialize (struct DPSE_DiscoveryPluginProperty * *self*)

Initialize a DiscoveryPluginProperty structure.

Parameters

self	<<*in*>> Cannot be NULL.

5.22.3.2 DDS_ReturnCode_t DPSE_RemoteParticipant_assert (DDS_DomainParticipant *const *participant,* const char * *rem_participant_name*)

<<*cert*>> Assert remote participant by passing in the name of that participant.

DPSE discovery requires that you pre-configure your application with information about any remote participants that you intend to discover. The only information that needs to be pre-configured is the name of the remote DDS_DomainParticipant that your application intends to discover.

Parameters

participant	<<*in*>> The DDS_DomainParticipant to assert the remote participant rem_participant_name into.
rem_participant_name	<<*in*>> The name of the remote participant as specified in DDS_ParticipantBuiltinTopicData::participant_name.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.22.3.3 DDS_ReturnCode_t DPSE_RemotePublication_assert (DDS_DomainParticipant *const *participant,* const char *const *rem_participant_name,* const struct DDS_PublicationBuiltinTopicData *const *data,* NDDS_TypePluginKeyKind *key_kind*)

<<*cert*>> Assert remote publication by passing in the builtin data that describes the type, topic, and QoS of that remote publication.

DPSE discovery requires that you pre-configure your application with information about any remote publications that you intend to discover and receive data from. The DDS_PublicationBuiltinTopicData field must include the topic name, type name, and QoS of the remote DDS_DataWriter that you intend to receive communications from.

Additionally, this method takes a key_kind argument that must match the key kind returned by the type plugin.

Parameters

participant	<<*in*>> The DDS_DomainParticipant to assert the remote publication defined by DDS_PublicationBuiltinTopicData into.
rem_participant_name	<<*in*>> The name of the remote participant as specified in DDS_ParticipantBuiltinTopicData::participant_name.
data	<<*in*>> The DDS_PublicationBuiltinTopicData defining the remote publication.
key_kind	<<*in*>> The IDL key type.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.22.3.4 DDS_ReturnCode_t DPSE_RemoteSubscription_assert (DDS_DomainParticipant *const *participant, const char *const *rem_participant_name, const struct DDS_SubscriptionBuiltinTopicData *const *data, NDDS_TypePluginKeyKind *key_kind*)

<<cert>> Assert remote subscription by passing in the builtin data that describes the type, topic, and QoS of that remote subscription.

DPSE discovery requires that you pre-configure your application with information about any remote subscriptions that you intend to discover and send data to. The DDS_SubscriptionBuiltinTopicData parameter must include the topic name, type name, and QoS of the remote DDS_DataReader that your application intends to send data to. Additionally, if the remote DDS_DataReader is listening on a non-default port, you must set the remote locator (including the non-default port) in the DDS_SubscriptionBuiltinTopicData that you intend to use to contact the remote DDS_DataReader.

Additionally, this method takes a key_kind argument that must match the key kind returned by the type plugin.

Parameters

participant	*<<in>>* The DDS_DomainParticipant to assert the remote subscription defined by DDS_SubscriptionBuiltinTopicData into.
rem_participant_name	*<<in>>* The name of the remote participant as specified in DDS_ParticipantBuiltinTopicData::participant_name.
data	*<<in>>* The DDS_SubscriptionBuiltinTopicData defining the remote subscription.
key_kind	*<<in>>* The IDL key type.

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.22.3.5 struct RT_ComponentFactoryI* DPSE_DiscoveryFactory_get_interface (void) [read]

Gets the singleton instance of the DPSE discovery plugin factory. The singleton instance of the factory is used when creating a domain participant.

This function gets the single singleton instance of the DPSE discovery plugin factory that is used by the middleware to create a DPSE plugin. In the future, as more discovery plug-ins are supported, they will be registered with the middleware by installing different discovery factory instances with the DDS_DomainParticipantFactory. The singleton instance of the factory is registered with a name, and that name is specified in the DDS_DomainParticipant DDS_DiscoveryQosPolicy. This is used to create a discovery plug-in instance when creating a domain participant.

Returns

Pointer to DPSE discovery plugin factory

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

DDS_DiscoveryQosPolicy

5.23 CDR Stream API

Functions

- RTI_UINT32 CDR_Stream_get_current_position_offset (struct CDR_Stream_t *cdrs)
  <<cert>> Return current offset of stream pointer relative to buffer
- RTI_BOOL CDR_Stream_set_current_position_offset (struct CDR_Stream_t *cdrs, RTI_UINT32 num)
  <<cert>> Set current offset of stream pointer relative to buffer
- RTI_BOOL CDR_Stream_increment_current_position (struct CDR_Stream_t *me, RTI_INT32 amount)
  <<cert>> Increase stream pointer by specified number of bytes
- RTI_BOOL CDR_Stream_check_size (struct CDR_Stream_t *me, RTI_UINT32 size)
  <<cert>> Verifies whether remaining space in stream buffer is larger than specified size
- char * CDR_Stream_get_current_position_ptr (struct CDR_Stream_t *me)
  <<cert>> Returns a pointer to the current offset in the stream
- RTI_BOOL CDR_Stream_is_byte_swapped (struct CDR_Stream_t *me)
  <<cert>> Whether a stream is byte-swapped

5.23.1 Detailed Description

<<eXtension>> *<<cert>>* CDR Stream

5.23.2 Function Documentation

5.23.2.1 RTI_UINT32 CDR_Stream_get_current_position_offset (struct CDR_Stream_t * *cdrs*)

<<cert>> Return current offset of stream pointer relative to buffer

Parameters

in	*cdrs*	Self

Returns

The number of bytes between the stream buffer and the current stream pointer.

MT Safety:

This operation is not thread safe.

5.23.2.2 RTI_BOOL CDR_Stream_set_current_position_offset (struct CDR_Stream_t * *cdrs*, RTI_UINT32 *num*)

<<cert>> Set current offset of stream pointer relative to buffer

Parameters

in	*cdrs*	Self
in	*num*	Offset in bytes to set stream pointer ahead of buffer

Returns

RTI_TRUE on success with stream pointer updated. Otherwise, RTI_FALSE on failure with stream pointer unchanged.

MT Safety:

This operation is not thread safe.

5.23.2.3 RTI_BOOL CDR_Stream_increment_current_position (struct CDR_Stream_t * *me*, RTI_INT32 *amount*)

<<*cert*>> Increase stream pointer by specified number of bytes

Parameters

in	*me*	Self
in	*amount*	Number of bytes to increase stream pointer

Returns

RTI_TRUE on success with stream pointer incremented. RTI_FALSE on failure with stream pointer unchanged.

MT Safety:

This operation is not thread safe.

5.23.2.4 RTI_BOOL CDR_Stream_check_size (struct CDR_Stream_t * *me*, RTI_UINT32 *size*)

<<*cert*>> Verifies whether remaining space in stream buffer is larger than specified size

Parameters

in	*me*	Self
in	*size*	Number of bytes to check whether fits in stream

Returns

RTI_TRUE if size fits within stream, otherwise RTI_FALSE.

MT Safety:

This operation is not thread safe.

5.23.2.5 char* CDR_Stream_get_current_position_ptr (struct CDR_Stream_t * *me*)

<<*cert*>> Returns a pointer to the current offset in the stream

Parameters

in	*me*	Self

Returns

Pointer to current offset in stream on success, NULL on failure.

MT Safety:

This operation is not thread safe.

5.23.2.6 RTI_BOOL CDR_Stream_is_byte_swapped (struct CDR_Stream_t * *me*)

<<*cert*>> Whether a stream is byte-swapped

This function verifies whether a stream is set to serialize or deserialize in byte-swapped order, relative to its host endianess.

Parameters

in	*me*	Self

Returns

RTI_TRUE if stream is byte-swapped

MT Safety:

This operation is not thread safe.

5.24 RT Run-Time API

Typedefs

- typedef union RT_ComponentFactoryId RT_ComponentFactoryId_T

  Abstract data-type to store the ID of a RT_ComponentFactory. Refer to RT_ComponentFactoryId_set_name to set a variable of this type.

Functions

- RTI_BOOL RT_Registry_register (RT_Registry_T *self, const char *name, struct RT_ComponentFactoryI *intf, struct RT_ComponentFactoryProperty *property, struct RT_ComponentFactoryListener *listener)

  <<cert>> Register a factory of RT components
- RTI_BOOL RT_Registry_unregister (RT_Registry_T *self, const char *name, struct RT_ComponentFactory-Property **property, struct RT_ComponentFactoryListener **listener)

  <<cert>> Remove a component factory from the registry
- RTI_BOOL RT_ComponentFactoryId_set_name (RT_ComponentFactoryId_T *id, const char *const name)

  Set the component ID from a string name.

5.24.1 Detailed Description

The run-time module (RT) manages RTI Connext Micro modules. It provides a common interface to add, remove, and lookup components.

5.24.2 Typedef Documentation

5.24.2.1 typedef union RT_ComponentFactoryId RT_ComponentFactoryId_T

Abstract data-type to store the ID of a RT_ComponentFactory. Refer to RT_ComponentFactoryId_set_name to set a variable of this type.

5.24.3 Function Documentation

5.24.3.1 RTI_BOOL RT_Registry_register (RT_Registry_T * *self,* const char * *name,* struct RT_ComponentFactoryI * *intf,* struct RT_ComponentFactoryProperty * *property,* struct RT_ComponentFactoryListener * *listener*)

<<cert>> Register a factory of RT components

In order to make a component available, it is necessary to register a factory for the components. The factory is responsible for creating and deleting components.

The name of the factory must be unique. Otherwise, if a factory already exists with the the same name, the registration will fail.

Parameters

self	*<<in>>* Cannot be NULL.
name	*<<in>>* Name of the component factory
intf	*<<in>>* Pointer to the implementation of the factory interface
property	*<<in>>* Properties with which to create the factory
listener	*<<in>>* The listener to install with the factory

Returns

RTI_TRUE on success, RTI_FALSE on failure

MT Safety:

This operation is not thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

See also

RT_Registry_unregister

5.24.3.2 RTI_BOOL RT_Registry_unregister (RT_Registry_T * *self,* const char * *name,* struct RT_ComponentFactoryProperty ** *property,* struct RT_ComponentFactoryListener ** *listener*)

<<*cert*>> Remove a component factory from the registry

After a component factory is unregistered, it will no longer be possible to create or delete components with the factory.

Parameters

self	<<*in*>> Cannot be NULL.
name	<<*in*>> Name of the component factory
property	<<*in*>> Properties with which the factory was created
listener	<<*in*>> The listener installed with the factory

Returns

RTI_TRUE on success, RTI_FALSE on failure

MT Safety:

This operation is not thread safe.

See also

RT_Registry_register

5.24.3.3 RTI_BOOL RT_ComponentFactoryId_set_name (RT_ComponentFactoryId_T * *id,* const char *const *name*)

Set the component ID from a string name.

Set the component ID from a string name. Cannot exceed 8 8-bit values, including the NUL string termination character.

Parameters

id	<<*inout*>> The ID to set.
name	<<*in*>> The name to set.

5.25 RHSM Reader History API

Functions

- struct RT_ComponentFactoryI * RHSM_HistoryFactory_get_interface (void)

  Gets the singleton instance of the RHSM reader history factory.

5.25.1 Detailed Description

The RHSM plugin is an implementation of the RTI Connext Micro ReaderHistory plugin interface and implements the DataReader reader cache as supported by RTI Connext Micro. RTI Connext Micro requires that a ReaderHistory plugin is registered with the name "rh" in the DomainParticipantFactory before a DataReader can be created. Each instance of a DataReader creates its own instance of the ReaderHistory plugin. Please refer to User Manual for an example on how to register the writer history factory.

5.25.2 Function Documentation

5.25.2.1 struct RT_ComponentFactoryI* RHSM_HistoryFactory_get_interface (void) [read]

Gets the singleton instance of the RHSM reader history factory.

This function gets the singleton instance of the RHSM history plugin factory that is used by the middleware to create a ReaderHistory plugin. The singleton instance of the factory must be registered with the name "rh" in the DomainParticipantFactory.

Returns

Pointer to RHSM history plugin factory

MT Safety:

This operation is thread safe.

5.26 WHSM Writer History API

Functions

- struct RT_ComponentFactoryI * WHSM_HistoryFactory_get_interface (void)

  Gets the singleton instance of the WHSM writer history factory.

5.26.1 Detailed Description

The WHSM plugin is an implementation of the RTI Connext Micro WriterHistory plugin interface and implements the DataWriter writer cache as supported by RTI Connext Micro. RTI Connext Micro requires that a WriterHistory plugin is registered with the name "wh" in the DomainParticipantFactory before a DataWriter can be created. Each instance of a DataWriter creates its own instance of the WriterHistory plugin. Please refer to User Manual for an example on how to register the writer history factory.

5.26.2 Function Documentation

5.26.2.1 struct RT_ComponentFactoryI* WHSM_HistoryFactory_get_interface (void) [read]

Gets the singleton instance of the WHSM writer history factory.

This function gets the singleton instance of the WHSM history plugin factory that is used by the middleware to create a WriterHistory plugin. The singleton instance of the factory must be registered with the name "wh" in the DomainParticipantFactory.

Returns

Pointer to WHSM history plugin factory

MT Safety:

This operation is thread safe.

5.27 NETIO API

<<eXtension>> *<<cert>>* The NETIO API.

Modules

- NETIO Packet API

  NETIO_Packet implementation.
- NETIO Interface API
- NETIO Address API

5.27.1 Detailed Description

<<eXtension>> *<<cert>>* The NETIO API.

5.28 RTPS Transport API

Data Structures

- union RTPS_Checksum

  <<eXtension>> <<cert>> RTPS checksum type
- struct RTPS_ChecksumClass

  <<eXtension>> <<cert>> Definition of a checksum function
- struct RTPS_ChecksumProperty

  <<eXtension>> <<cert>> Checksum properties for the RTPS plugin
- struct RTPS_InterfaceFactoryProperty

  <<eXtension>> <<cert>> Properties for the RTPS Transport.

Typedefs

- typedef union RTPS_Checksum RTPS_Checksum_T

  <<eXtension>> <<cert>> RTPS checksum type
- typedef RTI_BOOL(∗ RTPS_ChecksumCalculate_T)(void ∗context, const struct REDA_Buffer ∗buf, RTI_UINT32 buf_length, RTPS_Checksum_T ∗checksum)

  <<eXtension>> <<cert>> The checksum calculation function
- typedef struct RTPS_ChecksumClass RTPS_ChecksumClass_T

  <<eXtension>> <<cert>> Definition of a checksum function
- typedef struct RTPS_InterfaceFactoryProperty RTPS_InterfaceFactoryProperty_T

  <<eXtension>> <<cert>> Properties for the RTPS Transport.

Enumerations

- enum RTPS_ChecksumTxMode_T { RTPS_CHECKSUM_TXMODE_RTICRC32, RTPS_CHECKSUM_TXMODE_OMG }

  <<eXtension>> <<cert>> Transmit mode for checksums

Functions

- struct RT_ComponentFactoryI ∗ RTPS_InterfaceFactory_get_interface (void)

  Gets the singleton instance of the RTPS interface factory.

Variables

- RTI_UINT32 RTPS_Checksum::checksum32

  A 32-bit unsigned checksum in host endianess order.
- RTI_UINT64 RTPS_Checksum::checksum64

  A 64-bit unsigned integer checksum in host endianess order.
- RTI_UINT8 RTPS_Checksum::checksum128 [16]

  A 128-bit checksum as a 16-byte array.
- RTPS_ChecksumClassId_T RTPS_ChecksumClass::class_id

  The checksum class ID.
- void ∗ RTPS_ChecksumClass::context

  User defined context.
- RTPS_ChecksumCalculate_T RTPS_ChecksumClass::checksum_calculate

  User defined function to calculate a checksum.

- RTPS_ChecksumClass_T RTPS_ChecksumProperty::builtin_checksum32_class

  The builtin 32-bit checksum class.
- RTPS_ChecksumClass_T RTPS_ChecksumProperty::builtin_checksum64_class

  The builtin 64-bit checksum class.
- RTPS_ChecksumClass_T RTPS_ChecksumProperty::builtin_checksum128_class

  The builtin 128-bit checksum class.
- RTPS_ChecksumTxMode_T RTPS_ChecksumProperty::checksum_tx_mode

  The transmit mode for checksums.
- RTI_BOOL RTPS_ChecksumProperty::allow_builtin_override

  If TRUE, allow overriding the builtin checksum functions.
- struct RTPS_ChecksumProperty RTPS_InterfaceFactoryProperty::checksum

  The checksum properties for RTPS.
- struct RTPS_InterfaceFactoryProperty RTPS_INTERFACE_FACTORY_DEFAULT

  <<eXtension>> <<cert>> Default properties for the RTPS Transport.

5.28.1 Detailed Description

<<eXtension>> <<cert>> The RTPS Plugin implements the RTI Connext Micro RTPS protocol as an RTI Connext Micro NETIO interface.

5.28.2 Typedef Documentation

5.28.2.1 typedef union RTPS_Checksum RTPS_Checksum_T

<<eXtension>> <<cert>> RTPS checksum type

5.28.2.2 typedef RTI_BOOL(∗ RTPS_ChecksumCalculate_T)(void ∗context, const struct REDA_Buffer ∗buf, RTI_UINT32 buf_length, RTPS_Checksum_T ∗checksum)

<<eXtension>> <<cert>> The checksum calculation function

Parameters

in	*context*	Context passed as property
in	*buf*	Vector of buffers to calculate the checksum over
in	*buf_length*	Number of elements in buf
out	*checksum*	Calculated checksum

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.28.2.3 typedef struct RTPS_ChecksumClass RTPS_ChecksumClass_T

<<eXtension>> <<cert>> Definition of a checksum function

5.28.2.4 typedef struct RTPS_InterfaceFactoryProperty RTPS_InterfaceFactoryProperty_T

<<eXtension>> <<cert>> Properties for the RTPS Transport.

5.28.3 Enumeration Type Documentation

5.28.3.1 enum RTPS_ChecksumTxMode_T

<<eXtension>> <<cert>> Transmit mode for checksums

Enumerator:

RTPS_CHECKSUM_TXMODE_RTICRC32 Always use CRC32, fail if not possible.

RTPS_CHECKSUM_TXMODE_OMG Always use OMG's header extension [Default].

5.28.4 Function Documentation

5.28.4.1 struct RT_ComponentFactoryI* RTPS_InterfaceFactory_get_interface (void) [read]

Gets the singleton instance of the RTPS interface factory.

This function gets the singleton instance of the RTPS factory that is used by the middleware to create an RTPS transport instance.

Returns

Pointer to RTPS factory instance

MT Safety:

This operation is thread safe.

5.28.5 Variable Documentation

5.28.5.1 RTI_UINT32 RTPS_Checksum::checksum32

A 32-bit unsigned checksum in host endianess order.

5.28.5.2 RTI_UINT64 RTPS_Checksum::checksum64

A 64-bit unsigned integer checksum in host endianess order.

5.28.5.3 RTI_UINT8 RTPS_Checksum::checksum128[16]

A 128-bit checksum as a 16-byte array.

5.28.5.4 RTPS_ChecksumClassId_T RTPS_ChecksumClass::class_id

The checksum class ID.

5.28.5.5 void* RTPS_ChecksumClass::context

User defined context.

5.28.5.6 RTPS_ChecksumCalculate_T RTPS_ChecksumClass::checksum_calculate

User defined function to calculate a checksum.

5.28.5.7 RTPS_ChecksumClass_T RTPS_ChecksumProperty::builtin_checksum32_class

The builtin 32-bit checksum class.

5.28.5.8 RTPS_ChecksumClass_T RTPS_ChecksumProperty::builtin_checksum64_class

The builtin 64-bit checksum class.

5.28.5.9 RTPS_ChecksumClass_T RTPS_ChecksumProperty::builtin_checksum128_class

The builtin 128-bit checksum class.

5.28.5.10 RTPS_ChecksumTxMode_T RTPS_ChecksumProperty::checksum_tx_mode

The transmit mode for checksums.

5.28.5.11 RTI_BOOL RTPS_ChecksumProperty::allow_builtin_override

If TRUE, allow overriding the builtin checksum functions.

5.28.5.12 struct RTPS_ChecksumProperty RTPS_InterfaceFactoryProperty::checksum

The checksum properties for RTPS.

5.28.5.13 struct RTPS_InterfaceFactoryProperty RTPS_INTERFACE_FACTORY_DEFAULT

<<eXtension>> *<<cert>>* Default properties for the RTPS Transport.

5.29 NETIO DGRAM API

NETIO DGRAM Interface.

Data Structures

- struct NETIO_DGRAM_InterfaceMultiCastGroup
  <<eXtension>> <<cert>>.
- struct NETIO_DGRAM_InterfaceTableEntry
  <<eXtension>> <<cert>>.
- struct NETIO_DGRAM_InterfaceRouteEntry
  <<eXtension>> <<cert>>.
- struct NETIO_DGRAM_InterfaceTableEntrySeq
  A sequence of NETIO_DGRAMInterfaceTableEntry elements.
- struct NETIO_DGRAM_InterfaceI
  <<eXtension>> <<cert>>.

Macros

- #define NETIO_DGRAM_INTERFACE_SHARED_PORT_FLAG (0x1U)
  Flag that indicates that an interface uses shared ports.
- #define NETIO_DGRAM_INTERFACE_MAX_INTERFACES (8)
  The maximum number of network interfaces a NETIO_DGRAM Transport instance can manage.
- #define NETIO_DGRAM_INTERFACE_MAX_ADDRESSES (8)
  The maximum number of addresses which can be returned in an address reservation request.
- #define NETIO_DGRAM_InterfaceMultiCastGroup_INITIALIZER
  Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as empty.
- #define NETIO_DGRAM_InterfaceMultiCastGroup_Invalid
  Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as invalid.
- #define NETIO_DGRAM_InterfaceMultiCastGroup_UDPv4
  Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as a UDPv4 multicast group.
- #define NETIO_DGRAM_InterfaceMultiCastGroup_UDPv6
  Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as a UDPv6 multicast group.
- #define NETIO_DGRAM_InterfaceTableEntry_INITIALIZER
  Constant to initialize an NETIO_DGRAM_InterfaceTableEntry attribute as an empty entry.
- #define NETIO_DGRAMINTERFACETABLEENTRY_UDPV4(name_, addr_)
  Constant to initialize a UDPv4 interface.
- #define NETIO_DGRAM_InterfaceTableEntrySeq_INITIALIZER REDA_DEFINE_SEQUENCE_INITIALIZER(struct NETIO_DGRAM_InterfaceTableEntry)
  Constant to initialize a NETIO_DGRAM_InterfaceTableEntry as empty.

Typedefs

- typedef NETIO_Interface_T *(* NETIO_DGRAM_Interface_createFunc)(NETIO_Interface_T *upstream, void *property)
  The NETIO_DGRAM create interface function required by NETIO_DGRAM_InterfaceI.
- typedef RTI_BOOL(* NETIO_DGRAMUserInterface_get_interface_listFunc)(NETIO_Interface_T *user_intf, struct NETIO_DGRAM_InterfaceTableEntrySeq *if_table)
  The NETIO_DGRAM get interface list function required by NETIO_DGRAM_InterfaceI.
- typedef RTI_BOOL(* NETIO_DGRAMUserInterface_resolve_addressFunc)(NETIO_Interface_T *netio_intf, const struct NETIO_DGRAM_InterfaceTableEntry *if_entry, const char *address_string, struct NETIO_Address *address_value, RTI_BOOL *is_invalid)

The NETIO_DGRAM address resolve function.

- typedef RTI_BOOL(* NETIO_DGRAMUserInterface_bind_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source)

  The NETIO_DGRAM bind address function required by NETIO_DGRAM_InterfaceI.

- typedef struct NETIO_DGRAM_InterfaceI NETIO_DGRAM_InterfaceI

  <<*eXtension*>> <<*cert*>>.

Functions

- RTI_BOOL NETIO_DGRAM_InterfaceFactory_register (RT_Registry_T *registry, const char *name, NETIO_DGRAM_InterfaceI *user_intf, void *user_property)

  Register the DGRAM interface.

- RTI_BOOL NETIO_DGRAM_InterfaceFactory_unregister (RT_Registry_T *registry, const char *name)

  Unregister the DGRAM interface.

- RTI_BOOL NETIO_DGRAM_Interface_upstream_receive (NETIO_Interface_T *netio_intf, struct NETIO_Address *source, NETIO_Packet_T *packet)

  Forward data to the upstream interface.

- RTI_BOOL NETIO_DGRAM_Interface_add_route_to_seq (struct NETIO_AddressSeq *address_seq, struct NETIO_NetmaskSeq *netmask_seq, const struct NETIO_Address *address, const struct NETIO_Netmask *netmask)

  Adds the address and netmask to the respective sequences.

- RTI_BOOL NETIO_DGRAM_Interface_add_route_entry_to_seq (struct NETIO_AddressSeq *address_seq, struct NETIO_NetmaskSeq *netmask_seq, const struct NETIO_DGRAM_InterfaceRouteEntry *entry)

  Add a route entry to address and netmask sequence.

- RTI_BOOL NETIO_DGRAM_Interface_resolve_ipv4_address (NETIO_Interface_T *netio_intf, const struct NETIO_DGRAM_InterfaceTableEntry *if_entry, const char *address_string, struct NETIO_Address *address_value, RTI_BOOL *is_invalid)

  Resolve IPv4 address in dotted notation.

Variables

- struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv4_ANY_UNICAST_ROUTE

  Constant to initialize a UDPv4 unicast route.

- struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv4_ANY_MULTICAST_ROUTE

  Constant to initialize UDPv4 multicast route.

- struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv6_ANY_UNICAST_ROUTE

  Constant to initialize a UDPv6 unicast route.

- struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv6_ANY_MULTICAST_ROUTE

  Constant to initialize UDPv6 multicast route.

5.29.1 Detailed Description

NETIO DGRAM Interface. <<*eXtension*>> <<*cert*>> The NETIO DGRAM implements the RTI Connext Micro NETIO interface and exposes a subset of the interface enabling users to write an integration with concrete network interface.

5.29.2 Macro Definition Documentation

5.29.2.1 #define NETIO_DGRAM_INTERFACE_SHARED_PORT_FLAG (0x1U)

Flag that indicates that an interface uses shared ports.

RTI Connext Micro uses locators to specify transport addresses to send and receive data. A locator consists of a kind, a port, and a transport address. The kind indicates the type of transport (such as UDPv4), the port is used to reach a DDS DomainParticipant, and the address is used to reach the destination transport. RTI Connext Micro can work with two different types of transports, ones that use shared ports and one that does not.

When a transport uses shared ports, it does not matter which transport address a message was received on; only the port matters. For example, if a computer has two network interfaces A and B and is listening for messages on port P, it does not matter if the message is received on A or B. As long as the message is received on any network interface capable of receiving on port P, the message is accepted.

When a transport does not use shared ports, it does matter which transport address a message was received on. For example, if a computer has two network interfaces A and B and is listening for messages on port P, but has only specified that the A should receive on port P, then messages received on interface B and port P are ignored.

RTI Connext Micro supports this flag on a per-transport basis.

It is important to note that when a message is accepted, it is routed to all relevant DDS DataReaders and DataWriters. Thus, this feature cannot be used to control that some DDS topics should only be accepted when received on a specific transport interface. However, this feature could be useful to allow different DDS DomainParticipants to use the same port, but with different network interfaces.

5.29.2.2 #define NETIO_DGRAM_INTERFACE_MAX_INTERFACES (8)

The maximum number of network interfaces a NETIO_DGRAM Transport instance can manage.

Each NETIO_DGRAM Transport instance is associated with a network interface as returned by the NETIO_DGRAM_InterfaceTableEntrySeq. The maximum number of interfaces that can be managed by NETIO_DGRAM is NETIO_DGRAMINTERFACE_MAX_INTERFACES.

5.29.2.3 #define NETIO_DGRAM_INTERFACE_MAX_ADDRESSES (8)

The maximum number of addresses which can be returned in an address reservation request.

Each NETIO_DGRAM Transport instance receives address reservation requests to reserve addresses for receiving discovery and user data. The maximum number of addresses that can be reserved in a single request is NETIO_DGRAM_INTERFACE_MAX_ADDRESSES.

5.29.2.4 #define NETIO_DGRAM_InterfaceMultiCastGroup_INITIALIZER

Value:

```
{\
    NETIO_Address_INITIALIZER,\
    NETIO_Address_INITIALIZER,\
    NETIO_Netmask_INITIALIZER\
}
```

Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as empty.

5.29.2.5 #define NETIO_DGRAM_InterfaceMultiCastGroup_Invalid

Value:

```
{\
    {-1,0,{{0,0,0,0}}},\
    {-1,0,{{0,0,0,0}}},\
    {0,{0,0,0,0}} \
}
```

Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as invalid.

5.29.2.6 #define NETIO_DGRAM_InterfaceMultiCastGroup_UDPv4

Value:

```
{\
    {NETIO_ADDRESS_KIND_UDPv4,0,{{NETIO_htonl(0
      xe0000000),0,0,0}}},\
    {NETIO_ADDRESS_KIND_UDPv4,0,{{NETIO_htonl(0
      xefffffff),0,0,0}}},\
    {4,{0,0,0,0}} \
}
```

Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as a UDPv4 multicast group.

5.29.2.7 #define NETIO_DGRAM_InterfaceMultiCastGroup_UDPv6

Value:

```
{\
    {NETIO_ADDRESS_KIND_UDPv6,0,{{NETIO_htonl(0
      xff000000),0,0,0}}},\
    {NETIO_ADDRESS_KIND_UDPv6,0,{{0xffffffff,0xffffffff
      ,0xffffffff,0xffffffff}}},\
    {8,{0,0,0,0}} \
}
```

Constant to initialize a NETIO_DGRAM_InterfaceMultiCastGroup attribute as a UDPv6 multicast group.

5.29.2.8 #define NETIO_DGRAM_InterfaceTableEntry_INITIALIZER

Value:

```
{\
    0U,\
    0U,\
    0U,\
    NETIO_Address_INITIALIZER,\
    NETIO_DGRAM_InterfaceMultiCastGroup_INITIALIZER,\
    NULL\
}
```

Constant to initialize an NETIO_DGRAM_InterfaceTableEntry attribute as an empty entry.

5.29.2.9 #define NETIO_DGRAMINTERFACETABLEENTRY_UDPV4(*name_*, *addr_*)

Value:

```
{\
    1,\
    NETIO_DGRAM_INTERFACE_SHARED_PORT_FLAG,\
    65507U,\
    {1,0,{{NETIO_htonl(addr_),0,0,0}}},\
    NETIO_DGRAM_InterfaceMultiCastGroup_UDPv4,\
    name_\
}
```

Constant to initialize a UDPv4 interface.

A UDPv4 interface called 'name_' with the default UDPv4 address 'addr_'.

Parameters

*name_*	<<*in*>> The name associated with the entry.
*addr_*	<<*in*>> The UDPV4 address.

5.29.2.10 #define NETIO_DGRAM_InterfaceTableEntrySeq_INITIALIZER REDA_DEFINE_SEQUENCE_INITIALIZER(struct NETIO_DGRAM_InterfaceTableEntry)

Constant to initialize a NETIO_DGRAM_InterfaceTableEntry as empty.

5.29.3 Typedef Documentation

5.29.3.1 typedef NETIO_Interface_T*(* NETIO_DGRAM_Interface_createFunc)(NETIO_Interface_T *upstream, void *property)

The NETIO_DGRAM create interface function required by NETIO_DGRAM_InterfaceI.

5.29.3.2 typedef RTI_BOOL(* NETIO_DGRAMUserInterface_get_interface_listFunc)(NETIO_Interface_T *user_intf, struct NETIO_DGRAM_InterfaceTableEntrySeq *if_table)

The NETIO_DGRAM get interface list function required by NETIO_DGRAM_InterfaceI.

5.29.3.3 typedef RTI_BOOL(* NETIO_DGRAMUserInterface_resolve_addressFunc)(NETIO_Interface_T *netio_intf, const struct NETIO_DGRAM_InterfaceTableEntry *if_entry, const char *address_string, struct NETIO_Address *address_value, RTI_BOOL *is_invalid)

The NETIO_DGRAM address resolve function.

5.29.3.4 typedef RTI_BOOL(* NETIO_DGRAMUserInterface_bind_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source)

The NETIO_DGRAM bind address function required by NETIO_DGRAM_InterfaceI.

5.29.3.5 typedef struct NETIO_DGRAM_InterfaceI NETIO_DGRAM_InterfaceI

<<eXtension>> <<cert>>.

5.29.4 Function Documentation

5.29.4.1 RTI_BOOL NETIO_DGRAM_InterfaceFactory_register (RT_Registry_T * *registry,* const char * *name,* NETIO_DGRAM_InterfaceI * *user_intf,* void * *user_property*)

Register the DGRAM interface.

Registers the DGRAM interface component factory, sets the interface name, sets the instance's properties, and sets the user interface operations.

The interface operations are the supporting network layer functions that the DGRAM uses for it's routing, binding, and communication with peer NETIO.

Parameters

registry	*<<in>>* The registry instance.
name	*<<in>>* The interface name to register.
user_intf	*<<in>>* The user interface.
user_property	*<<in>>* The user defined property.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

5.29.4.2 RTI_BOOL NETIO_DGRAM_InterfaceFactory_unregister (RT_Registry_T * *registry,* const char * *name*)

Unregister the DGRAM interface.

Unregister the DGRAM interface.

Parameters

registry	<<*in*>> The registry instance.
name	<<*in*>> The interface name to unregister.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

5.29.4.3 RTI_BOOL NETIO_DGRAM_Interface_upstream_receive (NETIO_Interface_T * *netio_intf,* struct NETIO_Address * *source,* NETIO_Packet_T * *packet*)

Forward data to the upstream interface.

A NETIO_DGRAM interface receives data from a downstream interface. The data can be in different protocol formats. Depending on the downstream protocol, different actions may be taken, but semantically it does not matter where data is coming from. The receive interface is synchronous; all processing takes place in the context of the downstream receive context, such as a thread. The packet is a downstream packet that's being forwarded and must not be modified.

Parameters

netio_intf	<<*in*>> The interface receiving data.
source	<<*in*>> The source address of the data.
packet	<<*in*>> A NETIO_Packet with the payload

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.29.4.4 RTI_BOOL NETIO_DGRAM_Interface_add_route_to_seq (struct NETIO_AddressSeq * *address_seq,* struct NETIO_NetmaskSeq * *netmask_seq,* const struct NETIO_Address * *address,* const struct NETIO_Netmask * *netmask*)

Adds the address and netmask to the respective sequences.

Sets the address and netmask to the repsective sequence.

Parameters

address_seq	<<*in*>> <<*out*>> The address sequence to add to.
netmask_seq	<<*in*>> <<*out*>> The netmask sequence to add to.
address	<<*in*>> The address to add.
netmask	<<*in*>> The netmask to add.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

5.29.4.5 RTI_BOOL NETIO_DGRAM_Interface_add_route_entry_to_seq (struct NETIO_AddressSeq * *address_seq*, struct NETIO_NetmaskSeq * *netmask_seq*, const struct NETIO_DGRAM_InterfaceRouteEntry * *entry*)

Add a route entry to address and netmask sequence.

Adds the address_value to the address and route sequence.

Parameters

address_seq	<<*in*>> <<*out*>> The address sequence to add to.
netmask_seq	<<*in*>> <<*out*>> The netmask sequence to add to.
entry	<<*in*>> Entry to add.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

5.29.4.6 RTI_BOOL NETIO_DGRAM_Interface_resolve_ipv4_address (NETIO_Interface_T * *netio_intf*, const struct NETIO_DGRAM_InterfaceTableEntry * *if_entry*, const char * *address_string*, struct NETIO_Address * *address_value*, RTI_BOOL * *is_invalid*)

Resolve IPv4 address in dotted notation.

Resolve an address string with the user defined interface. The address_value kind and port values are ignored.

Parameters

netio_intf	<<*in*>> The interface.
if_entry	<<*in*>> The interface entry requested to resolve the address.
address_string	<<*in*>> Address to convert.
address_value	<<*out*>> Converted address.
is_invalid	<<*out*>> If the address is valid or not.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.29.5 Variable Documentation

5.29.5.1 struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv4_ANY_UNICAST_ROUTE

Constant to initialize a UDPv4 unicast route.

RTI Connext Micro uses an internal route table to determine which RTI Connext Micro transport to use to send outgoing messages (refer to NETIO_TransportI.get_route_table for details). The NETIO_ADDRESS_UDPv4_ANY-_UNICAST_ROUTE can be used to configure a transport as being able to send messages to any UDPv4 address.

5.29.5.2 struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv4_ANY_MULTICAST_ROUTE

Constant to initialize UDPv4 multicast route.

RTI Connext Micro uses an internal route table to determine which RTI Connext Micro transport to use to send outgoing messages (refer to NETIO_TransportI.get_route_table for details). The NETIO_ADDRESS_UDPv4_ANY_MULTICAST_ROUTE can be used to configure a transport as being able to send messages to any UDPv4 multicast address.

5.29.5.3 struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv6_ANY_UNICAST_ROUTE

Constant to initialize a UDPv6 unicast route.

RTI Connext Micro uses an internal route table to determine which RTI Connext Micro transport to use to send outgoing messages on (refer to NETIO_TransportI.get_route_table for details). The NETIO_ADDRESS_UDPv6_ANY_UNICAST_ROUTE can be used to configure a transport as being able to send messages to any UDPv6 address.

5.29.5.4 struct NETIO_DGRAM_InterfaceRouteEntry NETIO_ADDRESS_UDPv6_ANY_MULTICAST_ROUTE

Constant to initialize UDPv6 multicast route.

RTI Connext Micro uses an internal route table to determine which RTI Connext Micro transport to use to send outgoing messages on (refer to NETIO_TransportI.get_route_table for details). The NETIO_ADDRESS_UDPv6_ANY_MULTICAST_ROUTE can be used to configure a transport as being able to send messages to any UDPv6 multicast address.

5.30 Zero Copy v2 API

NETIO Zero Copy v2 interface.

Modules

- Notification Interface

  ZCOPY Notification Interface.
- User notification mechanism interface

  Notification mechanism definitions.

Functions

- RTI_BOOL NDDS_Transport_ZeroCopy_initialize (RT_Registry_T *registry, const char *transport_name, struct ZCOPY_NotifInterfaceFactoryProperty *property)

  Initialize the NDDS transport with ZeroCopy support.

Variables

- const char *const NETIO_DEFAULT_NOTIF_NAME

  Default name for notification interface.

5.30.1 Detailed Description

NETIO Zero Copy v2 interface. *<<eXtension>> <<cert>>* The NETIO Zero Copy v2 module implements the RTI Connext Micro Zero Copy v2 interface.

5.30.2 Function Documentation

5.30.2.1 RTI_BOOL NDDS_Transport_ZeroCopy_initialize (RT_Registry_T * *registry*, const char * *transport_name*, struct ZCOPY_NotifInterfaceFactoryProperty * *property*)

Initialize the NDDS transport with ZeroCopy support.

This function initializes the NDDS transport with ZeroCopy support by registering the transport and associated history factories in the provided registry. It also allows you to specify the transport name and a set of properties for the ZeroCopy notification interface.

Parameters

in	*registry*	The registry where the transport and history factories will be registered.
in	*transport_name*	The name of the transport. If NULL, the default transport name will be used.
in	*property*	A pointer to the properties for the ZeroCopy notification interface. If NULL, the transport will not be registered.

Returns

RTI_TRUE if the initialization was successful, RTI_FALSE otherwise.

Note

It is acceptable for `transport_name` and `property` to be NULL. If `transport_name` is NULL, the default transport name is used. If `property` is NULL, the transport is not registered.

5.30.3 Variable Documentation

5.30.3.1 const char* const NETIO_DEFAULT_NOTIF_NAME

Default name for notification interface.

5.31 OS API

Modules

- OSAPI Heap
- OSAPI Mutex

  Mutex API.
- OSAPI Process

  Abstract Process API.
- OSAPI Semaphore

  Abstract Semaphore API.
- OSAPI_SharedMemoryMonitor

  A shared memory monitor to provide mutual exclusion between processes and inter-process signaling.
- OSAPI_SharedMemorySegment

  Allow access to a shared memory segment (memory block) so that multiple processes can read-and write a common memory area.
- OSAPI Memory API
- OSAPI String API
- OSAPI System API

  Abstract System API.
- OSAPI Thread API

  The OSAPI Thread API.
- OSAPI Types

  OSAPI Types.

5.31.1 Detailed Description

5.32 Log API

The logging API provides access to log information.

Data Structures

- struct OSAPI_LogProperty

  Configuration of logging functionality.

Typedefs

- typedef void(∗ OSAPI_Log_write_buffer_T)(const char ∗buffer, RTI_SIZE_T length)

  Optional user-defined function to output a log/trace buffer.
- typedef void(∗ OSAPI_LogHandler_T)(void ∗param, OSAPI_LogEntry_T ∗entry)

  Optional user-defined function for processing new log messages.

Enumerations

- enum OSAPI_LogVerbosity_T { OSAPI_LOG_VERBOSITY_SILENT = 0, OSAPI_LOG_VERBOSITY_ERROR = 1, OSAPI_LOG_VERBOSITY_WARNING = 2, OSAPI_LOG_VERBOSITY_DEBUG = 3 }

  Logging verbosity.

Functions

- RTI_BOOL OSAPI_Log_set_log_handler (OSAPI_LogHandler_T handler, void ∗param)

  Install a log handler.
- RTI_BOOL OSAPI_Log_get_log_handler (OSAPI_LogHandler_T ∗handler, void ∗∗param)

  Return the current log handler.
- RTI_BOOL OSAPI_Log_set_trace_handler (OSAPI_TraceHandler_T handler, void ∗param)

  Install a trace handler.
- RTI_BOOL OSAPI_Log_get_trace_handler (OSAPI_TraceHandler_T ∗handler, void ∗∗param)

  Return the trace handler.
- RTI_BOOL OSAPI_Log_set_display_handler (OSAPI_LogDisplay_T handler, void ∗param)

  Install a display handler.
- RTI_BOOL OSAPI_Log_get_display_handler (OSAPI_LogDisplay_T ∗handler, void ∗∗param)

  Return the current display function.
- void OSAPI_Log_set_verbosity (OSAPI_LogVerbosity_T verbosity)

  Set the log verbosity.
- void OSAPI_Trace_set_trace_mask (RTI_UINT32 mask)

  Set the trace mask.
- RTI_BOOL OSAPI_Log_clear (void)

  Clear the log buffer.
- RTI_INT32 OSAPI_Log_get_last_error_code (void)

  Returns the error code for a function that failed.
- void OSAPI_Log_get_property (struct OSAPI_LogProperty ∗property)

  Get the current log properties.
- RTI_BOOL OSAPI_Log_set_property (struct OSAPI_LogProperty ∗property)

  Set the log properties.

5.32.1 Detailed Description

The logging API provides access to log information. Please note that the logging API is only available when RTI Connext Micro Cert is compiled in Debug mode.

5.32.2 Typedef Documentation

5.32.2.1 typedef void(* OSAPI_Log_write_buffer_T)(const char *buffer, RTI_SIZE_T length)

Optional user-defined function to output a log/trace buffer.

The handler is set by OSAPI_Log_set_property().

Parameters

in	*buffer*	Pointer to buffer with log to write
in	*length*	Length of the buffer to write

5.32.2.2 typedef void(* OSAPI_LogHandler_T)(void *param, OSAPI_LogEntry_T *entry)

Optional user-defined function for processing new log messages.

Definition of a function that can be installed with the logger and that will be called for each new log event.

The handler is set by OSAPI_Log_set_log_handler().

When called, provides parameters containing the raw log entry and additional optional information.

Parameters

in	*param*	User-defined payload
in	*msg*	Log entry
in	*kind*	Kind of log message
in	*ec*	Unique error code of log message
in	*module*	Module of log message
in	*file*	Name of source file. Optional.
in	*func*	Name of function. Optional.
in	*line*	Line number. Optional.
in	*argc*	Variable length argument count
in	*ap*	Variable length argument list

5.32.3 Enumeration Type Documentation

5.32.3.1 enum OSAPI_LogVerbosity_T

Logging verbosity.

Enumerator:

OSAPI_LOG_VERBOSITY_SILENT Logs are not written.
OSAPI_LOG_VERBOSITY_ERROR Only error logs are written.
OSAPI_LOG_VERBOSITY_WARNING Error and warning logs are written.
OSAPI_LOG_VERBOSITY_DEBUG All logs are written.

5.32.4 Function Documentation

5.32.4.1 RTI_BOOL OSAPI_Log_set_log_handler (OSAPI_LogHandler_T *handler,* void * *param*)

Install a log handler.

The log functionality allows the user to specify a log handler. The log handler is a function which is called for every logged event. It is up to the user to decide what to do with the log message. The handler is a global function pointer.

Parameters

in	*handler*	Pointer to log handler function
in	*param*	Parameter passed to the log handler function. This parameter is transparent to the log functionality.

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.32.4.2 RTI_BOOL OSAPI_Log_get_log_handler (OSAPI_LogHandler_T * *handler,* void ** *param*)

Return the current log handler.

Return the current log handler.

Parameters

in	*handler*	Pointer to store log handler function
in	*param*	Pointer to store the current log handler parameter.

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.32.4.3 RTI_BOOL OSAPI_Log_set_trace_handler (OSAPI_TraceHandler_T *handler,* void * *param*)

Install a trace handler.

Install a custom trace handler. Traces are not stored in the log-buffer and is generally used to analyze behavior interactively. the default trace handler outputs the trace data to a console.

Parameters

in	*handler*	Pointer to trace handler function
in	*param*	Parameter passed to the trace handler function. This parameter is transparent to the trace functionality.

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.32.4.4 RTI_BOOL OSAPI_Log_get_trace_handler (OSAPI_TraceHandler_T * *handler,* void ** *param*)

Return the trace handler.

Return the current trace handler and trace parameter. This information can be used to daisy-chain calls to multiple trace-handlers.

Parameters

in	*handler*	Pointer to store trace handler function
in	*param*	Pointer to store the trace handler parameter.

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.32.4.5 RTI_BOOL OSAPI_Log_set_display_handler (OSAPI_LogDisplay_T *handler,* void * *param*)

Install a display handler.

The display handler is responsible for outputting log messages to a console.

Parameters

in	*handler*	Pointer to display function
in	*param*	Parameter passed to the display handler function. This parameter is transparent to the log functionality.

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.32.4.6 RTI_BOOL OSAPI_Log_get_display_handler (OSAPI_LogDisplay_T * *handler,* void ** *param*)

Return the current display function.

Return the current log handler.

Parameters

in	*handler*	Pointer to store display handler function
in	*param*	Pointer to store the current display handler parameters.

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.32.4.7 void OSAPI_Log_set_verbosity (OSAPI_LogVerbosity_T *verbosity*)

Set the log verbosity.

Change the log verbosity. The new setting takes immediate effect.

Parameters

in	*verbosity*	New log verbosity

5.32.4.8 void OSAPI_Trace_set_trace_mask (RTI_UINT32 *mask*)

Set the trace mask.

Parameters

in	*mask*	New trace mask

5.32.4.9 RTI_BOOL OSAPI_Log_clear (void)

Clear the log buffer.

Clear the log buffer, all the current entries are lost

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.32.4.10 RTI_INT32 OSAPI_Log_get_last_error_code (void)

Returns the error code for a function that failed.

Many functions returns RTI_FALSE or NULL on failure. In order to provide additional information about reason for the failure fucntions may set an additional error code. This function returns the last error-code recorded for the calling thread.

Returns

Last error-code recorded for this thread

5.32.4.11 void OSAPI_Log_get_property (struct OSAPI_LogProperty * *property*)

Get the current log properties.

Return the current log properties

Parameters

out	*property*	- Current log properties

See Also OSAPI_Log_set_property

5.32.4.12 RTI_BOOL OSAPI_Log_set_property (struct OSAPI_LogProperty * *property*)

Set the log properties.

Set the current log properties. It is not possible to set new properties after OSAPI_Log_initialize() has been called/

Parameters

in	*property*	New log properties

Returns

RTI_TRUE if new properties can be set, RTI_FALSE otherwise See Also OSAPI_Log_get_property

5.33 REDA

Real-time Efficient Data Structures and Algorithms. ModuleID = 1.

Macros

- #define REDA_LOG_BUFFERPOOL_OUT_OF_RESOURCES_EC (REDA_LOG_BASE + 1)

  Not sufficient memory to allocate buffer-pool.
- #define REDA_LOG_BUFFERPOOL_BUFFER_INITIALIZATION_FAILED_EC (REDA_LOG_BASE + 2)

  The buffer initialization method failed.
- #define REDA_LOG_BUFFERPOOL_NOT_EMPTY_EC (REDA_LOG_BASE + 3)

  Cannot delete buffer-pool due to buffer(s) not returned to pool.
- #define REDA_LOG_BUFFERPOOL_DOUBLE_FREE_EC (REDA_LOG_BASE + 4)

  A buffer was freed more than once.
- #define REDA_LOG_SEQUENCE_COPY_FAILED_EC (REDA_LOG_BASE + 100)

  An error occurred while copying a sequence.
- #define REDA_LOG_SEQUENCE_ALLOC_FAILED_EC (REDA_LOG_BASE + 101)

  An error occured while allocating space for a sequence.
- #define REDA_LOG_SEQUENCE_SET_MAX_FAILED_EC (REDA_LOG_BASE + 102)

  An error occured while setting the maximum sequence size.
- #define REDA_LOG_SEQUENCE_REALLOCATION_EC (REDA_LOG_BASE + 103)

  An attempt was made to resize an already allocated sequence.
- #define REDA_LOG_SEQUENCE_INVALID_OPERATION_EC (REDA_LOG_BASE + 104)

  An error occured while operating on two sequences (copy/compare)
- #define REDA_LOG_SEQUENCE_INVALID_LENGTH_EC (REDA_LOG_BASE + 105)

  An attempt was made to set an illegal length (length $<$ 0 or length $>$ max_length)
- #define REDA_LOG_SEQUENCE_INDEX_OUT_OF_BOUNDS_EC (REDA_LOG_BASE + 106)

  An illegal sequence index was specified (index $<$ 0 or index $>$ length)
- #define REDA_LOG_STRING_ALLOC_FAILED_EC (REDA_LOG_BASE + 200)

  An error occured while allocating memory for a string.
- #define REDA_LOG_INDEX_FULL_EC (REDA_LOG_BASE + 300)

  An attempt was made to add an element to a full index.
- #define REDA_LOG_INDEX_ENTRY_EXISTS_EC (REDA_LOG_BASE + 301)

  An attempt was made to add an existing element to an index.
- #define REDA_LOG_INVALID_LIST_NODE_EC (REDA_LOG_BASE + 302)

  An attempt was made to add an existing element to an index.

5.33.1 Detailed Description

Real-time Efficient Data Structures and Algorithms. ModuleID = 1.

5.33.2 Macro Definition Documentation

5.33.2.1 #define REDA_LOG_BUFFERPOOL_OUT_OF_RESOURCES_EC (REDA_LOG_BASE + 1)

Not sufficient memory to allocate buffer-pool.

5.33.2.2 #define REDA_LOG_BUFFERPOOL_BUFFER_INITIALIZATION_FAILED_EC (REDA_LOG_BASE + 2)

The buffer initialization method failed.

5.33.2.3 #define REDA_LOG_BUFFERPOOL_NOT_EMPTY_EC (REDA_LOG_BASE + 3)

Cannot delete buffer-pool due to buffer(s) not returned to pool.

5.33.2.4 #define REDA_LOG_BUFFERPOOL_DOUBLE_FREE_EC (REDA_LOG_BASE + 4)

A buffer was freed more than once.

5.33.2.5 #define REDA_LOG_SEQUENCE_COPY_FAILED_EC (REDA_LOG_BASE + 100)

An error occurred while copying a sequence.

5.33.2.6 #define REDA_LOG_SEQUENCE_ALLOC_FAILED_EC (REDA_LOG_BASE + 101)

An error occured while allocating space for a sequence.

5.33.2.7 #define REDA_LOG_SEQUENCE_SET_MAX_FAILED_EC (REDA_LOG_BASE + 102)

An error occured while setting the maximum sequence size.

5.33.2.8 #define REDA_LOG_SEQUENCE_REALLOCATION_EC (REDA_LOG_BASE + 103)

An attempt was made to resize an already allocated sequence.

5.33.2.9 #define REDA_LOG_SEQUENCE_INVALID_OPERATION_EC (REDA_LOG_BASE + 104)

An error occured while operating on two sequences (copy/compare)

5.33.2.10 #define REDA_LOG_SEQUENCE_INVALID_LENGTH_EC (REDA_LOG_BASE + 105)

An attempt was made to set an illegal length (length $<$ 0 or length $>$ max_length)

5.33.2.11 #define REDA_LOG_SEQUENCE_INDEX_OUT_OF_BOUNDS_EC (REDA_LOG_BASE + 106)

An illegal sequence index was specified (index $<$ 0 or index $>$ length)

5.33.2.12 #define REDA_LOG_STRING_ALLOC_FAILED_EC (REDA_LOG_BASE + 200)

An error occured while allocating memory for a string.

5.33.2.13 #define REDA_LOG_INDEX_FULL_EC (REDA_LOG_BASE + 300)

An attempt was made to add an element to a full index.

5.33.2.14 #define REDA_LOG_INDEX_ENTRY_EXISTS_EC (REDA_LOG_BASE + 301)

An attempt was made to add an existing element to an index.

5.33.2.15 #define REDA_LOG_INVALID_LIST_NODE_EC (REDA_LOG_BASE + 302)

An attempt was made to add an existing element to an index.

5.34 OSAPI Heap

Macros

- #define OSAPI_ALIGNMENT_DEFAULT (-1)

  The default alignment.

Typedefs

- typedef RTI_INT32 OSAPI_Alignment_T

  The alignment type.

Functions

- void OSAPI_Heap_allocate_buffer (char **buffer, RTI_SIZE_T size, OSAPI_Alignment_T alignment)

  Allocate a block of memory of the provided size from the heap.

5.34.1 Detailed Description

5.34.2 Macro Definition Documentation

5.34.2.1 #define OSAPI_ALIGNMENT_DEFAULT (-1)

The default alignment.

5.34.3 Typedef Documentation

5.34.3.1 typedef RTI_INT32 OSAPI_Alignment_T

The alignment type.

Certain methods allow a default alignment. This should be an alignment that follows the "malloc" alignment of the architecture (aligned sufficiently to store any C-structure efficiently).

RTI Connext Micro Allocates all memory with OSAPI_ALIGNMENT_DEFAULT.

5.34.4 Function Documentation

5.34.4.1 void OSAPI_Heap_allocate_buffer (char ** *buffer*, RTI_SIZE_T *size*, OSAPI_Alignment_T *alignment*)

Allocate a block of memory of the provided size from the heap.

Parameters

buffer	<<*out*>> Cannot be NULL.
size	<<*in*>> Must be > 0.
alignment	<<*in*>> Power of 2 and >0 or OSAPI_ALIGNMENT_DEFAULT.

If the alignment is OSAPI_ALIGNMENT_DEFAULT, the returned pointer must have "default alignment" (meaning that any C-structure can start at the beginning of the buffer).

5.35 OSAPI Mutex

Mutex API.

Typedefs

- typedef struct OSAPI_Mutex OSAPI_Mutex_T
  The abstract mutex type.

Functions

- OSAPI_Mutex_T * OSAPI_Mutex_new (void)
  Create a mutex.
- RTI_BOOL OSAPI_Mutex_take (OSAPI_Mutex_T *self)
  Take a mutex.
- RTI_BOOL OSAPI_Mutex_give (OSAPI_Mutex_T *self)
  Give a mutex.

5.35.1 Detailed Description

Mutex API.

5.35.2 Typedef Documentation

5.35.2.1 typedef struct OSAPI_Mutex OSAPI_Mutex_T

The abstract mutex type.

5.35.3 Function Documentation

5.35.3.1 OSAPI_Mutex_T* OSAPI_Mutex_new (void)

Create a mutex.

Returns

Pointer to a mutex in a not taken condition. NULL on failure.

5.35.3.2 RTI_BOOL OSAPI_Mutex_take (OSAPI_Mutex_T * *self*)

Take a mutex.

A mutex can only be taken if it is not currently already taken by another thread; however, it can be taken if is it already taken by the same thread. In order to release a mutex, it must be given as many times as it has been taken. Take will block indefinitely.

Note: The mutex is not required to support priority inversion; there is no protection against deadlocks or starvation.

Parameters

self	<<*in*>> Take a mutex previously created with OSAPI_Mutex_new.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.35.3.3 RTI_BOOL OSAPI_Mutex_give (OSAPI_Mutex_T * *self*)

Give a mutex.

A mutex can only be given if it is owned by the calling thread. The mutex must be given as many times as it has been taken.

Parameters

self	<<*in*>> Take a mutex previously created with OSAPI_Mutex_new.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.36 OSAPI Process

Abstract Process API.

Typedefs

- typedef RTI_UINT64 OSAPI_ProcessId
  OSAPI_ProcessId type.

Functions

- OSAPI_ProcessId OSAPI_Process_getpid (void)
  Return the process ID, which is unique for the application.

5.36.1 Detailed Description

Abstract Process API.

5.36.2 Typedef Documentation

5.36.2.1 typedef RTI_UINT64 OSAPI_ProcessId

OSAPI_ProcessId type.

5.36.3 Function Documentation

5.36.3.1 OSAPI_ProcessId OSAPI_Process_getpid (void)

Return the process ID, which is unique for the application.

Returns

Process id

5.37 OSAPI Semaphore

Abstract Semaphore API.

Typedefs

- typedef struct OSAPI_Semaphore OSAPI_Semaphore_T

  The abstract semaphore type.

Functions

- OSAPI_Semaphore_T * OSAPI_Semaphore_new (void)

  Create a Semaphore.
- RTI_BOOL OSAPI_Semaphore_take (OSAPI_Semaphore_T *self, RTI_INT32 timeout, RTI_INT32 *fail_reason)

  Take a semaphore.
- RTI_BOOL OSAPI_Semaphore_give (OSAPI_Semaphore_T *self)

  Give a semaphore.

5.37.1 Detailed Description

Abstract Semaphore API.

5.37.2 Typedef Documentation

5.37.2.1 typedef struct OSAPI_Semaphore OSAPI_Semaphore_T

The abstract semaphore type.

5.37.3 Function Documentation

5.37.3.1 OSAPI_Semaphore_T* OSAPI_Semaphore_new (void)

Create a Semaphore.

The created semaphore is a binary semaphore with an initial semaphore value set to 0 and a maximum value of 1. Thus, a call to OSAPI_Semaphore_take will block until the semaphore is signaled with a call to OSAPI_Semaphore_give.

Returns

A pointer to a semaphore with a value of 1 on success, NULL on failure.

5.37.3.2 RTI_BOOL OSAPI_Semaphore_take (OSAPI_Semaphore_T * *self*, RTI_INT32 *timeout*, RTI_INT32 * *fail_reason*)

Take a semaphore.

Parameters

self	<<*in*>> A semaphore previously created with OSAPI_Semaphore_new.
timeout	<<*in*>> The timeout in ms. The take call will wait at most timeout ms before returning. If OSAPI_SEMAPHORE_TIMEOUT_INFINITE is specified, the call will not return until the semaphore has a value of 1 or higher.

fail_reason	<<*out*>> Additional information when the call returns. If RTI_TRUE is returned, but timed out, fail_reason is set to OSAPI_SEMAPHORE_RESULT_TIMEOUT, otherwise fail_reason is set to OSAPI_SEMAPHORE_RESULT_OK. If RTI_FALSE is returned, fail_reason is set toOSAPI_SEMAPHORE_RESULT_ERROR.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.37.3.3 RTI_BOOL OSAPI_Semaphore_give (OSAPI_Semaphore_T * *self*)

Give a semaphore.

A mutex can only be given if it is owned by the calling thread. The mutex must be given as many times as it has been taken.

Parameters

self	<<*in*>> The semaphore previously created with OSAPI_Semaphore_new.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.38 OSAPI_SharedMemoryMonitor

A shared memory monitor to provide mutual exclusion between processes and inter-process signaling.

Functions

- RTI_SIZE_T OSAPI_SharedMemoryMonitor_get_size (void)

  Get the amount of memory required to store a OSAPI_SharedMemoryMonitor on the current platform.
- RTI_BOOL OSAPI_SharedMemoryMonitor_initialize (RTI_SIZE_T mem_len, void *mem, struct OSAPI_SharedMemoryMonitor **self_out)

  Initialize a shared memory monitor in a provided memory segment.
- RTI_BOOL OSAPI_SharedMemoryMonitor_acquire (struct OSAPI_SharedMemoryMonitor *self, OSAPI_SHMEM_STATUS *status_out)

  Acquire the lock protecting the monitor.
- RTI_BOOL OSAPI_SharedMemoryMonitor_release (struct OSAPI_SharedMemoryMonitor *self)

  Release the lock protecting the monitor.
- RTI_BOOL OSAPI_SharedMemoryMonitor_wait (struct OSAPI_SharedMemoryMonitor *self, OSAPI_SHMEM_STATUS *status_out)

  Wait for the monitor to be signaled.
- RTI_BOOL OSAPI_SharedMemoryMonitor_signal (struct OSAPI_SharedMemoryMonitor *self)

  Signal a thread waiting on a monitor to wake up.
- RTI_BOOL OSAPI_SharedMemoryMonitor_mark_consistent (struct OSAPI_SharedMemoryMonitor *self)

  Mark the shared state protected by the monitor as consistent after a process died while holding the lock on the monitor.

5.38.1 Detailed Description

A shared memory monitor to provide mutual exclusion between processes and inter-process signaling.

5.38.2 Function Documentation

5.38.2.1 RTI_SIZE_T OSAPI_SharedMemoryMonitor_get_size (void)

Get the amount of memory required to store a OSAPI_SharedMemoryMonitor on the current platform.

Returns

The number of bytes required to store a shared memory monitor.

5.38.2.2 RTI_BOOL OSAPI_SharedMemoryMonitor_initialize (RTI_SIZE_T *mem_len*, void * *mem*, struct OSAPI_SharedMemoryMonitor ** *self_out*)

Initialize a shared memory monitor in a provided memory segment.

Parameters

in	*mem_len*	Size of provided memory.
in	*mem*	Memory segment.
out	*self_out*	Pointer to initialized monitor.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

5.38.2.3 RTI_BOOL OSAPI_SharedMemoryMonitor_acquire (struct OSAPI_SharedMemoryMonitor * *self*, OSAPI_SHMEM_STATUS * *status_out*)

Acquire the lock protecting the monitor.

Parameters

in	*self*	Monitor to acquire.
out	*status_out*	If successful, the status of the protected state.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

If this function returns RTI_TRUE, the calling thread is guaranteed mutual exclusion by the monitor. No other thread can acquire the lock until the lock is returned with OSAPI_SharedMemoryMonitor_release. This function shall not be called by a thread which currently holds the lock.

If this function returns status_out=OSAPI_SHMEM_STATUS_OWNER_DEAD, then any shared data protected by the monitor may be in an inconsistent state. If the shared data can be restored to a consistent state, then it can be marked as consistent with OSAPI_SharedMemoryMonitor_mark_consistent. Otherwise, any subsequent attempts to acquire or wait on the monitor will fail.

5.38.2.4 RTI_BOOL OSAPI_SharedMemoryMonitor_release (struct OSAPI_SharedMemoryMonitor * *self*)

Release the lock protecting the monitor.

Parameters

in	*self*	Monitor to release.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

This function can only be called on a monitor which is currently locked by the calling thread.

5.38.2.5 RTI_BOOL OSAPI_SharedMemoryMonitor_wait (struct OSAPI_SharedMemoryMonitor * *self*, OSAPI_SHMEM_STATUS * *status_out*)

Wait for the monitor to be signaled.

Parameters

in	*self*	Monitor to wait on.
out	*status_out*	If successful, the status of the protected state.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

Only one thread can wait on a monitor at any one time. This function can only be called on a monitor which is currently locked by the calling thread. The function will return successfully only when another thread has called OSAPI_SharedMemoryMonitor_signal and when the calling thread has acquired the lock.

If this function returns status_out=OSAPI_SHMEM_STATUS_OWNER_DEAD, then any shared data protected by the monitor may be in an inconsistent state. If the shared data can be restored to a consistent state, then it can be marked as consistent with OSAPI_SharedMemoryMonitor_mark_consistent. Otherwise, any subsequent attempts to acquire or wait on the monitor will fail.

5.38.2.6 RTI_BOOL OSAPI_SharedMemoryMonitor_signal (struct OSAPI_SharedMemoryMonitor * *self*)

Signal a thread waiting on a monitor to wake up.

Parameters

in	*self*	Monitor to signal.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

This function can only be called on a monitor which is currently locked by the calling thread.

5.38.2.7 RTI_BOOL OSAPI_SharedMemoryMonitor_mark_consistent (struct OSAPI_SharedMemoryMonitor * *self*)

Mark the shared state protected by the monitor as consistent after a process died while holding the lock on the monitor.

Parameters

in	*self*	Monitor to mark consistent.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

This function can only be called on a monitor which is currently locked by the calling thread and currently inconsistent. The function shall return the monitor to a consistent state and allow other functions to be called on it. This function may not be supported on all platforms.

5.39 OSAPI_SharedMemorySegment

Allow access to a shared memory segment (memory block) so that multiple processes can read-and write a common memory area.

Data Structures

- struct OSAPI_SharedMemorySegmentHeader
- struct OSAPI_SharedMemorySegmentHandle

Enumerations

- enum OSAPI_SHMEM_MODE { OSAPI_SHMEM_MODE_UNKNOWN, OSAPI_SHMEM_MODE_READ, OSAPI_SHMEM_MODE_WRITE, OSAPI_SHMEM_MODE_LOCKABLE, OSAPI_SHMEM_MODE_ROBUST }
- enum OSAPI_SHMEM_STATUS { OSAPI_SHMEM_STATUS_OK, OSAPI_SHMEM_STATUS_OWNER_DEAD }
- enum OSAPI_SHMEM_ATTACH_STATUS { OSAPI_SHMEM_ATTACH_STATUS_OK, OSAPI_SHMEM_ATTACH_STATUS_SEGMENT_NOT_FOUND }

Functions

- RTI_UINT64 OSAPI_SharedMemorySegment_get_max_size (void)

  Get maximum size of a shared memory segment on the current platform.
- RTI_BOOL OSAPI_SharedMemorySegment_is_owner_alive (struct OSAPI_SharedMemorySegmentHandle *handle, RTI_BOOL *alive)

  Check if the process which created the shared memory segment is still alive.
- RTI_SIZE_T OSAPI_SharedMemorySegmentHandle_get_size (OSAPI_SHMEM_MODE mode)

  Get the amount of memory required to store a OSAPI_SharedMemorySegmentHandle on the current platform for a given mode.
- RTI_SIZE_T OSAPI_SharedMemorySegmentHeader_get_size (OSAPI_SHMEM_MODE mode)

  Get the amount of memory, aligned to the maximum alignment, required to store a OSAPI_SharedMemorySegmentHeader on the current platform for a given mode.
- RTI_BOOL OSAPI_SharedMemorySegment_create_impl (struct OSAPI_SharedMemorySegmentHandle *handle, const char *name, RTI_UINT64 size, OSAPI_SHMEM_MODE mode, RTI_BOOL *exists)

  Creates a new shared memory segment with the provided key and initialize a handle to it.
- RTI_BOOL OSAPI_SharedMemorySegment_delete_impl (struct OSAPI_SharedMemorySegmentHandle *handle)

  Delete a shared memory segment.
- RTI_BOOL OSAPI_SharedMemorySegment_attach_impl (struct OSAPI_SharedMemorySegmentHandle *handle, const char *name, OSAPI_SHMEM_MODE mode, OSAPI_SHMEM_ATTACH_STATUS *status_out)

  Attach to the shared memory and initialize a handle to it.
- RTI_BOOL OSAPI_SharedMemorySegment_detach_impl (struct OSAPI_SharedMemorySegmentHandle *handle)

  detach the shared memory segment.
- RTI_BOOL OSAPI_SharedMemorySegment_lock_impl (struct OSAPI_SharedMemorySegmentHandle *handle, OSAPI_SHMEM_STATUS *status_out)

  Lock a shared memory segment.
- RTI_BOOL OSAPI_SharedMemorySegment_unlock_impl (struct OSAPI_SharedMemorySegmentHandle *handle)

  Unlock a shared memory segment previously locked with OSAPI_SharedMemorySegment_lock_impl.

- RTI_BOOL OSAPI_SharedMemorySegment_mark_consistent_impl (struct OSAPI_SharedMemorySegmentHandle *handle)

  Mark a shared memory segment as consistent. This function can only be called if a previous call to OSAPI_SharedMemorySegment_lock_impl returned OSAPI_SHMEM_FAIL_REASON_OWNER_DEAD.

5.39.1 Detailed Description

Allow access to a shared memory segment (memory block) so that multiple processes can read-and write a common memory area.

5.39.2 Enumeration Type Documentation

5.39.2.1 enum OSAPI_SHMEM_MODE

Mode used to create or attach to a shared memory segment. Each mode defines specific capabilities of the segment.

Enumerator:

OSAPI_SHMEM_MODE_UNKNOWN Invalid mode

OSAPI_SHMEM_MODE_READ Read permission

OSAPI_SHMEM_MODE_WRITE Write permission, implies OSAPI_SHMEM_MODE_READ

OSAPI_SHMEM_MODE_LOCKABLE Locking capability, implies OSAPI_SHMEM_MODE_WRITE

OSAPI_SHMEM_MODE_ROBUST Robust locking capability, implies OSAPI_SHMEM_MODE_LOCKABLE

5.39.2.2 enum OSAPI_SHMEM_STATUS

Status used to convey the state of a shared memory object.

Enumerator:

OSAPI_SHMEM_STATUS_OK Success

OSAPI_SHMEM_STATUS_OWNER_DEAD Previous owner of lock died while holding it

5.39.2.3 enum OSAPI_SHMEM_ATTACH_STATUS

Status used to convey the state of a shared memory object's attachment to a shared memory segment.

Enumerator:

OSAPI_SHMEM_ATTACH_STATUS_OK Success

OSAPI_SHMEM_ATTACH_STATUS_SEGMENT_NOT_FOUND Segment was not found when trying to attach

5.39.3 Function Documentation

5.39.3.1 RTI_UINT64 OSAPI_SharedMemorySegment_get_max_size (void)

Get maximum size of a shared memory segment on the current platform.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

5.39.3.2 RTI_BOOL OSAPI_SharedMemorySegment_is_owner_alive (struct OSAPI_SharedMemorySegmentHandle * *handle*, RTI_BOOL * *alive*)

Check if the process which created the shared memory segment is still alive.

Parameters

in	*handle*	The shared memory segment handle identifying the segment.
out	*alive*	If the process is alive.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

The behavior or this function is undefined if the segment is not currently attached.

5.39.3.3 RTI_SIZE_T OSAPI_SharedMemorySegmentHandle_get_size (OSAPI_SHMEM_MODE *mode*)

Get the amount of memory required to store a OSAPI_SharedMemorySegmentHandle on the current platform for a given mode.

Parameters

in	*mode*	The creation or attachment mode of the handle.

Returns

Number of bytes required to store a handle.

5.39.3.4 RTI_SIZE_T OSAPI_SharedMemorySegmentHeader_get_size (OSAPI_SHMEM_MODE *mode*)

Get the amount of memory, aligned to the maximum alignment, required to store a OSAPI_SharedMemorySegmentHeader on the current platform for a given mode.

Parameters

in	*mode*	The creation mode of the segment containing the header.

Returns

Number of bytes required to store a header.

The number of bytes returned by this function shall be aligned to the maximum alignment of the platform.

5.39.3.5 RTI_BOOL OSAPI_SharedMemorySegment_create_impl (struct OSAPI_SharedMemorySegmentHandle * *handle*, const char * *name*, RTI_UINT64 *size*, OSAPI_SHMEM_MODE *mode*, RTI_BOOL * *exists*)

Creates a new shared memory segment with the provided key and initialize a handle to it.

Parameters

in,out	*handle*	Handle to be initialized.
in	*name*	Name to identify the shared memory segment.
in	*size*	Size of memory to be allocated.
in	*mode*	Mode used to create the shared memory segment. The mode determines the capabilities of the segment. The mode must be >= OSAPI_SHMEM_MODE_-WRITE.
out	*exists*	If the segment already exists

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

The new shared memory segment shall be initialized with null bytes ('\0').

5.39.3.6 RTI_BOOL OSAPI_SharedMemorySegment_delete_impl (struct OSAPI_SharedMemorySegmentHandle * *handle*)

Delete a shared memory segment.

Parameters

in	*handle*	The shared memory segment handle identifying the segment to delete.

Returns

RTI_TRUE on success. RTI_FALSE if either the precondition was not met or the shared memory segment could not be deleted.

Once this function is called (even if it returns RTI_FALSE) you can no longer use this shared memory segment.

5.39.3.7 RTI_BOOL OSAPI_SharedMemorySegment_attach_impl (struct OSAPI_SharedMemorySegmentHandle * *handle,* const char * *name,* OSAPI_SHMEM_MODE *mode,* OSAPI_SHMEM_ATTACH_STATUS * *status_out*)

Attach to the shared memory and initialize a handle to it.

Parameters

in,out	*handle*	Handle to be initialized.
in	*name*	Name to identify the shared memory segment.
in	*mode*	Mode used to attach the shared memory segment.
out	*status_out*	If successful, the attachment status of the shared memory segment.

Returns

RTI_TRUE on success. Note that the attachment status may still be different from OK, so checking the value of status_out is mandatory. RTI_FALSE if an error occurred.

5.39.3.8 RTI_BOOL OSAPI_SharedMemorySegment_detach_impl (struct OSAPI_SharedMemorySegmentHandle * *handle*)

detach the shared memory segment.

Parameters

in	*handle*	The shared memory segment handle identifying the segment to detach.

Returns

RTI_TRUE on success. RTI_FALSE if either the precondition was not met or the shared memory segment could not be detached.

Once this function is called (even if it returns RTI_FALSE) you can no longer use this shared memory segment handle.

5.39.3.9 RTI_BOOL OSAPI_SharedMemorySegment_lock_impl (struct OSAPI_SharedMemorySegmentHandle * *handle*, OSAPI_SHMEM_STATUS * *status_out*)

Lock a shared memory segment.

Parameters

in	*handle*	The shared memory segment handle identifying the segment to lock.
out	*status_out*	Success or failure reason of acquiring the lock.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

5.39.3.10 RTI_BOOL OSAPI_SharedMemorySegment_unlock_impl (struct OSAPI_SharedMemorySegmentHandle * *handle*)

Unlock a shared memory segment previously locked with OSAPI_SharedMemorySegment_lock_impl.

Parameters

in	*handle*	The shared memory segment handle identifying the segment to unlock.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

The behavior of this method is undefined if the calling process has not previously acquired the lock with OSAPI_SharedMemorySegment_lock_impl.

5.39.3.11 RTI_BOOL OSAPI_SharedMemorySegment_mark_consistent_impl (struct OSAPI_SharedMemorySegmentHandle * *handle*)

Mark a shared memory segment as consistent. This function can only be called if a previous call to OSAPI_SharedMemorySegment_lock_impl returned OSAPI_SHMEM_FAIL_REASON_OWNER_DEAD.

Parameters

in	*handle*	The shared memory segment handle identifying the segment to mark consistent.

Returns

RTI_TRUE on success. RTI_FALSE if an error occurred.

5.40 OSAPI Memory API

Functions

- void OSAPI_Memory_copy (void *dest, const void *src, RTI_SIZE_T size)

  Copies size bytes from memory address source (src) to destination (dest).
- void OSAPI_Memory_zero (void *mem, RTI_SIZE_T size)

  Sets memory region (mem) to 0.
- RTI_INT32 OSAPI_Memory_compare (const void *left, const void *right, RTI_SIZE_T size)

  Byte-wise comparison of two memory regions.
- void OSAPI_Memory_move (void *dest, const void *source, RTI_SIZE_T size)

  Moves size bytes from non-overlapping source to destination (dest).
- void * OSAPI_Memory_fndchr (const void *s, RTI_INT32 c, RTI_SIZE_T n)

  Locate byte in byte string.

5.40.1 Detailed Description

5.40.2 Function Documentation

5.40.2.1 void OSAPI_Memory_copy (void * *dest*, const void * *src*, RTI_SIZE_T *size*)

Copies size bytes from memory address source (src) to destination (dest).

Valid destination and source memory regions must be >= size, where size >= 0. Additionally, the source and destination memory regions must not overlap, or the results will be undetermined. On success, the value of the first size bytes in the destination and source regions will be identical.

Parameters

dest	<<*out*>> Pointer to the destination memory region.
src	<<*in*>> Pointer to the source memory region.
size	<<*in*>> Number of bytes to copy, starting at the destination.

Exceptions

None.	

MT Safety:

The source region cannot be modified concurrently and the destination region cannot be modified or read concurrently.

5.40.2.2 void OSAPI_Memory_zero (void * *mem*, RTI_SIZE_T *size*)

Sets memory region (mem) to 0.

Clear a memory region. Valid memory >= size, where size >= 0. On success, the memory region is filled with 0.

Parameters

mem	<<*out*>> Pointer to the memory region to be modified.
size	<<*in*>> Number of bytes to fill with zeroes, starting at mem.

Exceptions

None.	

MT Safety:

The memory region cannot be modified or read concurrently.

5.40.2.3 RTI_INT32 OSAPI_Memory_compare (const void * *left*, const void * *right*, RTI_SIZE_T *size*)

Byte-wise comparison of two memory regions.

This function performs a byte-wise comparison of two memory regions (of size bytes). The left and right parameters must be >= 0 and size >= 0. This function does not alter any memory content.

Parameters

left	<<*in*>> Pointer to logical "left" side of the inequality test.
right	<<*in*>> Pointer to logical "right" side of the inequality test.
size	<<*in*>> Number of bytes to compare, starting from left to right.

Returns

Less than, equal to, or greater than 0, according to whether left is lexicographically less than, equal to, or greater than right when taken as unsigned characters.

5.40.2.4 void OSAPI_Memory_move (void * *dest*, const void * *source*, RTI_SIZE_T *size*)

Moves size bytes from non-overlapping source to destination (dest).

Copy size bytes from the source to the destination. Valid destination and source regions must be >= size, where size >= 0. The source and destination memory regions may overlap. On return, size number of bytes from source has been copied to the destination region and is identical to the source region.

Parameters

dest	<<*out*>> Pointer to the destination memory region.
source	<<*in*>> Pointer to the source memory region.
size	<<*in*>> Number of bytes to copy from the source to the destination.

Exceptions

None.	

MT Safety:

The source cannot be modified concurrently; dest cannot be modified or read concurrently.

5.40.2.5 void* OSAPI_Memory_fndchr (const void * *s*, RTI_INT32 *c*, RTI_SIZE_T *n*)

Locate byte in byte string.

Locate byte in byte string.

Parameters

s	<<*in*>> Pointer to beginning of byte string.
c	<<*in*>> Byte to search for.
n	<<*in*>> Maximum search length.

Returns

Pointer to first occurrence of c if found, NULL otherwise.

5.41 OSAPI String API

Functions

- RTI_SIZE_T OSAPI_String_length (const char *s)

  Return length of ASCIIZ string.
- RTI_INT32 OSAPI_String_cmp (const char *l, const char *r)

  Compare two ASCIIZ strings.
- RTI_INT32 OSAPI_String_ncmp (const char *l, const char *r, RTI_SIZE_T num)

  Return the process ID.

5.41.1 Detailed Description

5.41.2 Function Documentation

5.41.2.1 RTI_SIZE_T OSAPI_String_length (const char * *s*)

Return length of ASCIIZ string.

Return length of ASCIIZ string not including \0.

Parameters

s	<<*in*>> Pointer to beginning of byte string.

Returns

Length of string not including \0.

5.41.2.2 RTI_INT32 OSAPI_String_cmp (const char * *l*, const char * *r*)

Compare two ASCIIZ strings.

Lexicographically compare the null-terminated strings l and r. The comparison is using unsigned arithmetic.

Parameters

l	<<*in*>> Left side string.
r	<<*in*>> Right side string.

Returns

0 if the strings are identical, a positive integer if left is greater then right, and a negative integer if left less then right.

5.41.2.3 RTI_INT32 OSAPI_String_ncmp (const char * *l*, const char * *r*, RTI_SIZE_T *num*)

Return the process ID.

Lexicographically compare null-terminated strings l and r, including up to num characters (if the null terminator is not found before the specified number of characters has been compared). The comparison uses unsigned arithmetic.

Parameters

l	<<*in*>> Left side string.
r	<<*in*>> Right side string.
num	<<*in*>> Maximum number of characters to compare.

Returns

0 if the strings are identical, a positive integer if left is greated than right, and a negative integer if left is less than right.

5.42 OSAPI System API

Abstract System API.

Data Structures

- struct OSAPI_SystemI
  <<cert>> The System API.
- struct OSAPI_SystemProperty
  <<cert>> The system properties.

Macros

- #define OSAPI_SystemI_INITIALIZER
  <<cert>> The initializer for OSAPI_SystemI.

Typedefs

- typedef RTI_BOOL(* OSAPI_System_start_timer_T)(OSAPI_Timer_T self, OSAPI_TimerTickHandlerFunction tick_handler)
  Start an OSAPI timer.
- typedef RTI_INT32(* OSAPI_System_get_timer_resolution_T)(void)
  Get the resolution of the clock driving the timer in nanoseconds.
- typedef RTI_BOOL(* OSAPI_System_get_time_T)(OSAPI_SystemTime *now)
  Get the current system time.
- typedef RTI_BOOL(* OSAPI_System_initialize_T)(void)
  Initialize the system.
- typedef RTI_BOOL(* OSAPI_System_get_hostname_T)(char *const hostname)
  Get the hostname.
- typedef RTI_BOOL(* OSAPI_System_generate_uuid_T)(struct OSAPI_SystemUUID *uuid_out)
  Generate a unique universal identifier (UUID).
- typedef RTI_BOOL(* OSAPI_System_get_ticktime_T)(RTI_INT32 *sec, RTI_UINT32 *nanosec)
  Get the current tick time.
- typedef struct OSAPI_SystemI OSAPI_SystemI
  <<cert>> The System API.

Functions

- RTI_BOOL OSAPI_System_set_interface (struct OSAPI_SystemI *intf)
  Set the system interface.
- RTI_BOOL OSAPI_System_get_property (struct OSAPI_SystemProperty *property)
  Get the system properties.
- RTI_BOOL OSAPI_System_set_property (struct OSAPI_SystemProperty *property)
  Set the system properties.
- void OSAPI_System_get_native_interface (struct OSAPI_SystemI *intf)
  Get the native system interface.
- RTI_BOOL OSAPI_System_clock_tick (RTI_UINT32 ticks)
  Advance the internal RTI Connext Micro timers with the specified number of clock ticks.
- RTI_BOOL OSAPI_System_clock_tick_from_time (void)
  Advance the internal RTI Connext Micro timers using the OSAPI_SystemI::get_time API.

5.42.1 Detailed Description

Abstract System API. The system is defined as the physical or virtual execution environment and implements functions that are not necessarily provided by the operating system. For example, a custom embedded board may include a special real-time clock which is synchronized via GPS, but is not available via a regular OS system call.

The following functions are not thread-safe:

- OSAPI_System_set_interface
- OSAPI_System_get_property
- OSAPI_System_set_property
- OSAPI_System_get_native_interface
- OSAPI_System_clock_tick
- OSAPI_System_clock_tick_from_time

5.42.2 Macro Definition Documentation

5.42.2.1 #define OSAPI_SystemI_INITIALIZER

Value:

```
{\
    NULL,\
    NULL,\
    NULL,\
    NULL,\
    NULL,\
    NULL,\
    NULL\
}
```

<<*cert*>> The initializer for OSAPI_SystemI.

5.42.3 Typedef Documentation

5.42.3.1 typedef RTI_BOOL(* OSAPI_System_start_timer_T)(OSAPI_Timer_T self, OSAPI_TimerTickHandlerFunction tick_handler)

Start an OSAPI timer.

Parameters

self	<<*in*>> The timer object.
tick_handler	<<*in*>> The timer handle.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.42.3.2 typedef RTI_INT32(* OSAPI_System_get_timer_resolution_T)(void)

Get the resolution of the clock driving the timer in nanoseconds.

This function must return the frequency of the system timer used to implement OSAPI_SystemI::start_timer and OSAPI_SystemI::stop_timer API.

Returns

The timer resolution in nanoseconds.

5.42.3.3 typedef RTI_BOOL(* OSAPI_System_get_time_T)(OSAPI_SystemTime *now)

Get the current system time.

In general, the system time is used by components to correlate both internal and external events, such as data reception and ordering. Thus, it is recommended that this function returns the real time. However, it is not strictly required.

Notes:

- It is assumed that the time returned by get_time is monotonically increasing. It is up to the implementation of this function to ensure this holds true.

- It is ok to return the same time as the last call.

- The clock used to report real-time can be different than the clock used to support start_timer and stop_timer.

Parameters

now	<<*out*>> Time.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.42.3.4 typedef RTI_BOOL(* OSAPI_System_initialize_T)(void)

Initialize the system.

This function initializes the system and calls the port-specific initialization method first. The port-specific method must return RTI_TRUE on success and RTI_FALSE on failure. A system can only be initialized once.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.42.3.5 typedef RTI_BOOL(* OSAPI_System_get_hostname_T)(char *const hostname)

Get the hostname.

Parameters

hostname	<<*out*>> The buffer to store the hostname. Must be at least OSAPI_SYSTEM_MAX_HOSTNAME bytes. If the actual hostname is longer than OSAPI_SYSTEM_MAX_HOSTNAME bytes (including the NUL termination), the hostname is truncated.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.42.3.6 typedef RTI_BOOL(* OSAPI_System_generate_uuid_T)(struct OSAPI_SystemUUID *uuid_out)

Generate a unique universal identifier (UUID).

Parameters

uuid_out	<<*out*>> The generated UUID value.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.42.3.7 typedef RTI_BOOL(* OSAPI_System_get_ticktime_T)(RTI_INT32 *sec, RTI_UINT32 *nanosec)

Get the current tick time.

The ticktime is a time measurement used by RTI Connext Micro to determine how much time has elapsed in a period. It does not have to be an absolute time, but it must be monotonically increasing. For this reason it is important to choose the time source with care. For example, the system time may be adjusted backward or forward, and this could affect the measured time lapse. The resolution of the tick time is expected to be no less than the system timer, although this is not a requirement.

Parameters

sec	<<*out*>> The current ticktime in seconds.
nanosec	<<*out*>> Additional nanoseconds in the current ticktime.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.42.3.8 typedef struct OSAPI_SystemI OSAPI_SystemI

<<*cert*>> The System API.

The methods in this interface are called by the corresponding OSAPI_System function. For example, OSAPI_System_get_time() calls system -> get_time().

5.42.4 Function Documentation

5.42.4.1 RTI_BOOL OSAPI_System_set_interface (struct OSAPI_SystemI * *intf*)

Set the system interface.

This function sets the system interface, overriding the current system interface.

Parameters

intf	<<*in*>> If intf is NULL, all interfaces are set to NULL.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

API Restriction:

This function may be called before DDS_DomainParticipantFactory_get_instance.

5.42.4.2 RTI_BOOL OSAPI_System_get_property (struct OSAPI_SystemProperty * *property*)

Get the system properties.

'Copy the current system properties to the argument.

Parameters

property	<<*out*>> The current system properties on success.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

API Restriction:

This function may be called before DDS_DomainParticipantFactory_get_instance.

5.42.4.3 RTI_BOOL OSAPI_System_set_property (struct OSAPI_SystemProperty * *property*)

Set the system properties.

The system properties can be set until the system has been initialized by calling OSAPI_System_initialize(). Note that OSAPI_System_initialize() is called by DDS_DomainParticipantFactory_get_instance().

Parameters

property	<<*in*>> The system properties to set.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

API Restriction:

This function may be called before DDS_DomainParticipantFactory_get_instance.

5.42.4.4 void OSAPI_System_get_native_interface (struct OSAPI_SystemI * *intf*)

Get the native system interface.

This function returns the native system interface.

Parameters

intf	<<*out*>> Cannot be NULL.

MT Safety:

This operation is not thread safe.

API Restriction:

This function may be called before DDS_DomainParticipantFactory_get_instance.

5.42.4.5 RTI_BOOL OSAPI_System_clock_tick (RTI_UINT32 *ticks*)

Advance the internal RTI Connext Micro timers with the specified number of clock ticks.

RTI Connext Micro uses internal timers to maintain and detect DDS events, such as sending participant announcements and detecting missed deadlines. These timers are updated at fixed intervals called ticks where each tick is

equal to the time specified by the timer resolution from OSAPI_System_get_timer_resolution. However, because RTI Connext Micro does not maintain its own timer reference, it is up to the platform integration to inform RTI Connext Micro when ticks have elapsed by calling the clock_tick function. The clock_tick function advances the internal timers with ticks. It is up to the caller to determine how many ticks have elapsed when calling the clock_tick function.

Parameters

ticks	<<*out*>> The number of ticks that have passed.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.42.4.6 RTI_BOOL OSAPI_System_clock_tick_from_time (void)

Advance the internal RTI Connext Micro timers using the OSAPI_SystemI::get_time API.

RTI Connext Micro uses internal timers to maintain and detect DDS events, such as sending participant announcements and detecting missed deadlines. These timers are updated at fixed intervals called ticks where each tick is equal to the time specified by the timer resolution from OSAPI_SystemI.get_timer_resolution. However, because RTI Connext Micro does not maintain its own timer reference it is up to the platform integration to inform RTI Connext Micro when ticks have elapsed by calling the clock_tick or 'clock_tick_from_time' function.

The clock_tick_from_time function advances the internal timers with ticks calculated from the amount of time that has elapsed between calls to OSAPI_SystemI.get_time. Each time clock_tick_from_time is called, OSAPI_SystemI.get_time is called to determine how much time has elapsed since the previous call to clock_tick_from_time. The difference is converted to ticks using the resolution from OSAPI_SystemI.get_timer_resolution and all internal timers are advanced using the calculated number of ticks. Any residual time left (that is, any time less than the OSAPI_SystemI.get_timer_resolution) is carried over and used on the next call to clock_tick_from_time.

Note: The first call to clock_tick_from_time establishes the baseline for calculating time differences.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.43 OSAPI Thread API

The OSAPI Thread API.

Data Structures

- struct OSAPI_ThreadProperty
  The properties for thread creation.

Macros

- #define OSAPI_THREAD_PRIORITY_LOW -1
  Low priority.
- #define OSAPI_THREAD_PRIORITY_BELOW_NORMAL -2
  Below normal priority.
- #define OSAPI_THREAD_PRIORITY_NORMAL -3
  Normal priority.
- #define OSAPI_THREAD_PRIORITY_ABOVE_NORMAL -4
  Above normal priority.
- #define OSAPI_THREAD_PRIORITY_HIGH -5
  High priority.
- #define OSAPI_THREAD_PRIORITY_INHERIT -6
  Use the same priority as the spawning thread. Only applies to POSIX.
- #define OSAPI_THREAD_USE_OSDEFAULT_STACKSIZE 0
  Use the default stack size when creating a thread.
- #define OSAPI_THREAD_DEFAULT_OPTIONS 0x00
  Use only the default options from the OS.
- #define OSAPI_THREAD_FLOATING_POINT 0x01
  Hint to the OS to create a thread that supports floating point.
- #define OSAPI_THREAD_STDIO 0x02
  Hint to the OS to create a thread that supports standard I/O.
- #define OSAPI_THREAD_REALTIME_PRIORITY 0x08
  Hint to the OS to create a thread that runs in real-time priority mode.
- #define OSAPI_THREAD_PROPERTY_DEFAULT
  The default thread properties.
- #define OSAPI_ThreadProperty_INITIALIZER OSAPI_THREAD_PROPERTY_DEFAULT
  The initializer for OSAPI_ThreadProperty.

Typedefs

- typedef RTI_UINT32 OSAPI_ThreadOptions
  Thread options.

5.43.1 Detailed Description

The OSAPI Thread API.

5.43.2 Macro Definition Documentation

5.43.2.1 #define OSAPI_THREAD_PRIORITY_LOW -1

Low priority.

5.43.2.2 #define OSAPI_THREAD_PRIORITY_BELOW_NORMAL -2

Below normal priority.

5.43.2.3 #define OSAPI_THREAD_PRIORITY_NORMAL -3

Normal priority.

5.43.2.4 #define OSAPI_THREAD_PRIORITY_ABOVE_NORMAL -4

Above normal priority.

5.43.2.5 #define OSAPI_THREAD_PRIORITY_HIGH -5

High priority.

5.43.2.6 #define OSAPI_THREAD_PRIORITY_INHERIT -6

Use the same priority as the spawning thread. Only applies to POSIX.

5.43.2.7 #define OSAPI_THREAD_USE_OSDEFAULT_STACKSIZE 0

Use the default stack size when creating a thread.

5.43.2.8 #define OSAPI_THREAD_DEFAULT_OPTIONS 0x00

Use only the default options from the OS.

5.43.2.9 #define OSAPI_THREAD_FLOATING_POINT 0x01

Hint to the OS to create a thread that supports floating point.

5.43.2.10 #define OSAPI_THREAD_STDIO 0x02

Hint to the OS to create a thread that supports standard I/O.

5.43.2.11 #define OSAPI_THREAD_REALTIME_PRIORITY 0x08

Hint to the OS to create a thread that runs in real-time priority mode.

5.43.2.12 #define OSAPI_THREAD_PROPERTY_DEFAULT

Value:

```
{ \
    OSAPI_THREAD_USE_OSDEFAULT_STACKSIZE, \
    OSAPI_THREAD_PRIORITY_DEFAULT, \
    OSAPI_THREAD_DEFAULT_OPTIONS \
}
```

The default thread properties.

5.43.2.13 #define OSAPI_ThreadProperty_INITIALIZER OSAPI_THREAD_PROPERTY_DEFAULT

The initializer for OSAPI_ThreadProperty.

5.43.3 Typedef Documentation

5.43.3.1 typedef RTI_UINT32 OSAPI_ThreadOptions

Thread options.

5.44 OSAPI Types

OSAPI Types.

Macros

- #define RTI_INT8
  A 2's complement 8-bit signed integer.
- #define RTI_UINT8
  An 8-bit unsigned integer.
- #define RTI_INT16
  A 2's complement 16-bit signed integer.
- #define RTI_UINT16
  A 2's complement 16-bit unsigned integer.
- #define RTI_INT32
  A 2's complement 32-bit signed integer.
- #define RTI_UINT32
  A 32-bit unsigned integer.
- #define RTI_INT64
  A 64-bit signed integer.
- #define RTI_UINT64
  A 64-bit unsigned integer.
- #define RTI_FALSE ((RTI_BOOL) 0)
- #define RTI_TRUE ((RTI_BOOL) 1)

Typedefs

- typedef RTI_INT32 RTI_BOOL
- typedef RTI_UINT32 RTI_SIZE_T

5.44.1 Detailed Description

OSAPI Types.

5.44.2 Macro Definition Documentation

5.44.2.1 #define RTI_INT8

A 2's complement 8-bit signed integer.

5.44.2.2 #define RTI_UINT8

An 8-bit unsigned integer.

5.44.2.3 #define RTI_INT16

A 2's complement 16-bit signed integer.

5.44.2.4 #define RTI_UINT16

A 2's complement 16-bit unsigned integer.

5.44.2.5 #define RTI_INT32

A 2's complement 32-bit signed integer.

5.44.2.6 #define RTI_UINT32

A 32-bit unsigned integer.

5.44.2.7 #define RTI_INT64

A 64-bit signed integer.

5.44.2.8 #define RTI_UINT64

A 64-bit unsigned integer.

5.44.2.9 #define RTI_FALSE ((RTI_BOOL) 0)

Functions that return RTI_BOOL return RTI_FALSE as logical false.

5.44.2.10 #define RTI_TRUE ((RTI_BOOL) 1)

Functions that return RTI_BOOL return RTI_TRUE as logical true.

5.44.3 Typedef Documentation

5.44.3.1 typedef RTI_INT32 RTI_BOOL

The type RTI_BOOL uses RTI_TRUE as logical true and RTI_FALSE as logical false. In general, a zero value is considered false, and a non-zero value is considered true.

5.44.3.2 typedef RTI_UINT32 RTI_SIZE_T

5.45 DB

Database. ModuleID = 2.

Macros

- #define DB_LOG_SORTED_ALLOC_EC (DB_LOG_BASE + 1)

  Not sufficient memory to allocate sorted list for an index.
- #define DB_LOG_NAME_TOO_LONG_EC (DB_LOG_BASE + 2)

  The specified database name is too long.
- #define DB_LOG_ILLEGAL_TABLE_SIZE_EC (DB_LOG_BASE + 3)

  Illegal table size specified.
- #define DB_LOG_ILLEGAL_LOCK_MODE_EC (DB_LOG_BASE + 4)

  Illegal combination of lock mode and mutex given.
- #define DB_LOG_MUTEX_ALLOC_EC (DB_LOG_BASE + 5)

  Failed to allocate database mutex.
- #define DB_LOG_ALLOC_TABLE_POOL_EC (DB_LOG_BASE + 6)

  Failed to allocate buffer pool.
- #define DB_LOG_TABLES_INUSE_EC (DB_LOG_BASE + 7)

  Table is in use.
- #define DB_LOG_TABLE_NAME_TOO_LONG_EC (DB_LOG_BASE + 8)

  Specified table name too long.
- #define DB_LOG_ILLEGAL_RECORD_COUNT_EC (DB_LOG_BASE + 9)

  Illegal record count specified.
- #define DB_LOG_TABLE_EXISTS_EC (DB_LOG_BASE + 10)

  Specified table already exists.
- #define DB_LOG_OUT_OF_TABLES_EC (DB_LOG_BASE + 11)

  Table resources exceeded.
- #define DB_LOG_OUT_OF_RECORDS_EC (DB_LOG_BASE + 12)

  Table record resources exceeded.
- #define DB_LOG_OUT_OF_INDICES_EC (DB_LOG_BASE + 13)

  Table index resources exceeded.
- #define DB_LOG_OUT_OF_CURSORS_EC (DB_LOG_BASE + 14)

  Table cursor resources exceeded.
- #define DB_LOG_RECORDS_INUSE_EC (DB_LOG_BASE + 15)

  Cannot delete table, records are still in table.
- #define DB_LOG_CURSORS_INUSE_EC (DB_LOG_BASE + 16)

  Cannot delete table, cursors are still in use.
- #define DB_LOG_INDEX_INUSE_EC (DB_LOG_BASE + 17)

  Cannot delete table, indices are still in use.
- #define DB_LOG_ALLOC_CURSOR_POOL_EC (DB_LOG_BASE + 18)

  Failed to allocate cursor buffer pool.
- #define DB_LOG_ALLOC_INDEX_POOL_EC (DB_LOG_BASE + 19)

  Failed to allocate index buffer pool.
- #define DB_LOG_ALLOC_RECORD_POOL_EC (DB_LOG_BASE + 20)

  Failed to allocate record buffer pool.
- #define DB_LOG_ALLOC_DATABASE_EC (DB_LOG_BASE + 21)

  Failed to allocate database.
- #define DB_LOG_TABLE_NOT_INUSE_EC (DB_LOG_BASE + 22)

  An attempt was made to delete a table not in use.
- #define DB_LOG_RECORD_ALREADY_EXISTS_EC (DB_LOG_BASE + 23)

An attempt was made to insert an already existing record.

- #define DB_LOG_RECORD_DOES_NOT_EXIST_EC (DB_LOG_BASE + 24)

  An attempt was made to delete a record that does not exist.
- #define DB_LOG_CURSOR_INVALIDATED_EC (DB_LOG_BASE + 25)

  An attempt was made to use a cursor that has been invalidated.
- #define DB_LOG_LOCK_FAILURE_EC (DB_LOG_BASE + 26)

  Failed to lock the database.
- #define DB_LOG_UNLOCK_FAILURE_EC (DB_LOG_BASE + 27)

  Failed to unlock the database.

5.45.1 Detailed Description

Database. ModuleID = 2.

5.45.2 Macro Definition Documentation

5.45.2.1 #define DB_LOG_SORTED_ALLOC_EC (DB_LOG_BASE + 1)

Not sufficient memory to allocate sorted list for an index.

5.45.2.2 #define DB_LOG_NAME_TOO_LONG_EC (DB_LOG_BASE + 2)

The specified database name is too long.

5.45.2.3 #define DB_LOG_ILLEGAL_TABLE_SIZE_EC (DB_LOG_BASE + 3)

Illegal table size specified.

5.45.2.4 #define DB_LOG_ILLEGAL_LOCK_MODE_EC (DB_LOG_BASE + 4)

Illegal combination of lock mode and mutex given.

5.45.2.5 #define DB_LOG_MUTEX_ALLOC_EC (DB_LOG_BASE + 5)

Failed to allocate database mutex.

5.45.2.6 #define DB_LOG_ALLOC_TABLE_POOL_EC (DB_LOG_BASE + 6)

Failed to allocate buffer pool.

5.45.2.7 #define DB_LOG_TABLES_INUSE_EC (DB_LOG_BASE + 7)

Table is in use.

5.45.2.8 #define DB_LOG_TABLE_NAME_TOO_LONG_EC (DB_LOG_BASE + 8)

Specified table name too long.

5.45.2.9 #define DB_LOG_ILLEGAL_RECORD_COUNT_EC (DB_LOG_BASE + 9)

Illegal record count specified.

5.45.2.10 #define DB_LOG_TABLE_EXISTS_EC (DB_LOG_BASE + 10)

Specified table already exists.

5.45.2.11 #define DB_LOG_OUT_OF_TABLES_EC (DB_LOG_BASE + 11)

Table resources exceeded.

5.45.2.12 #define DB_LOG_OUT_OF_RECORDS_EC (DB_LOG_BASE + 12)

Table record resources exceeded.

5.45.2.13 #define DB_LOG_OUT_OF_INDICES_EC (DB_LOG_BASE + 13)

Table index resources exceeded.

5.45.2.14 #define DB_LOG_OUT_OF_CURSORS_EC (DB_LOG_BASE + 14)

Table cursor resources exceeded.

5.45.2.15 #define DB_LOG_RECORDS_INUSE_EC (DB_LOG_BASE + 15)

Cannot delete table, records are still in table.

5.45.2.16 #define DB_LOG_CURSORS_INUSE_EC (DB_LOG_BASE + 16)

Cannot delete table, cursors are still in use.

5.45.2.17 #define DB_LOG_INDEX_INUSE_EC (DB_LOG_BASE + 17)

Cannot delete table, indices are still in use.

5.45.2.18 #define DB_LOG_ALLOC_CURSOR_POOL_EC (DB_LOG_BASE + 18)

Failed to allocate cursor buffer pool.

5.45.2.19 #define DB_LOG_ALLOC_INDEX_POOL_EC (DB_LOG_BASE + 19)

Failed to allocate index buffer pool.

5.45.2.20 #define DB_LOG_ALLOC_RECORD_POOL_EC (DB_LOG_BASE + 20)

Failed to allocate record buffer pool.

5.45.2.21 #define DB_LOG_ALLOC_DATABASE_EC (DB_LOG_BASE + 21)

Failed to allocate database.

5.45.2.22 #define DB_LOG_TABLE_NOT_INUSE_EC (DB_LOG_BASE + 22)

An attempt was made to delete a table not in use.

5.45.2.23 #define DB_LOG_RECORD_ALREADY_EXISTS_EC (DB_LOG_BASE + 23)

An attempt was made to insert an already existing record.

5.45.2.24 #define DB_LOG_RECORD_DOES_NOT_EXIST_EC (DB_LOG_BASE + 24)

An attempt was made to delete a record that does not exist.

5.45.2.25 #define DB_LOG_CURSOR_INVALIDATED_EC (DB_LOG_BASE + 25)

An attempt was made to use a cursor that has been invalidated.

5.45.2.26 #define DB_LOG_LOCK_FAILURE_EC (DB_LOG_BASE + 26)

Failed to lock the database.

5.45.2.27 #define DB_LOG_UNLOCK_FAILURE_EC (DB_LOG_BASE + 27)

Failed to unlock the database.

5.46 CDR

CDR. ModuleID = 5.

Macros

- #define CDR_LOG_STREAM_ALLOC_EC (CDR_LOG_BASE + 1)

  Failed to allocate stream or stream buffer.
- #define CDR_LOG_SET_OFFSET_EC (CDR_LOG_BASE + 2)

  Failed to set stream's current offset.
- #define CDR_LOG_INCR_OFFSET_EC (CDR_LOG_BASE + 3)

  Failed to increament stream's current offset.
- #define CDR_LOG_UNSUPPORTED_OPTIONS_EC (CDR_LOG_BASE + 5)

  Unsupported CDR encapsulations were received. The sample is dropped.

5.46.1 Detailed Description

CDR. ModuleID = 5.

5.46.2 Macro Definition Documentation

5.46.2.1 #define CDR_LOG_STREAM_ALLOC_EC (CDR_LOG_BASE + 1)

Failed to allocate stream or stream buffer.

5.46.2.2 #define CDR_LOG_SET_OFFSET_EC (CDR_LOG_BASE + 2)

Failed to set stream's current offset.

5.46.2.3 #define CDR_LOG_INCR_OFFSET_EC (CDR_LOG_BASE + 3)

Failed to increament stream's current offset.

5.46.2.4 #define CDR_LOG_UNSUPPORTED_OPTIONS_EC (CDR_LOG_BASE + 5)

Unsupported CDR encapsulations were received. The sample is dropped.

5.47 RT

RT. ModuleID = 3.

Macros

- #define RT_LOG_SET_IMMUTABLE_PROPERTY_EC (RT_LOG_BASE + 1)

  Failed to set registry property due to registry already being enabled.
- #define RT_LOG_REGISTRY_INIT_FAILURE_EC (RT_LOG_BASE + 2)

  Failed to initialize registry due to failed table creation.
- #define RT_LOG_REGISTRY_FINALIZE_EC (RT_LOG_BASE + 3)

  Failed to finalize registry due to failed table deletion.
- #define RT_LOG_REGISTRY_EXISTS_EC (RT_LOG_BASE + 4)

  An attempt was made to register a factory that already existed.
- #define RT_LOG_REGISTRY_REGISTER_EC (RT_LOG_BASE + 5)

  Error registering a component factory. May have exceeded RT_RegistryProperty.max_factories.
- #define RT_LOG_REGISTRY_INIT_FACTORY_EC (RT_LOG_BASE + 7)

  A registered factory failed to initialize.
- #define RT_LOG_REGISTRY_NAME_TOO_LONG_EC (RT_LOG_BASE + 8)

  Factory name longer than maximum length of 8.
- #define RT_LOG_REGISTRY_INCONSISTENT_CID_EC (RT_LOG_BASE + 9)

  Factory name exists, but the class ID is not of the requested type.
- #define RT_LOG_REGISTRY_NOT_INITIALIZED_EC (RT_LOG_BASE + 10)

  An attempt was made to operate on an uninitialized registry.

5.47.1 Detailed Description

RT. ModuleID = 3.

5.47.2 Macro Definition Documentation

5.47.2.1 #define RT_LOG_SET_IMMUTABLE_PROPERTY_EC (RT_LOG_BASE + 1)

Failed to set registry property due to registry already being enabled.

5.47.2.2 #define RT_LOG_REGISTRY_INIT_FAILURE_EC (RT_LOG_BASE + 2)

Failed to initialize registry due to failed table creation.

5.47.2.3 #define RT_LOG_REGISTRY_FINALIZE_EC (RT_LOG_BASE + 3)

Failed to finalize registry due to failed table deletion.

5.47.2.4 #define RT_LOG_REGISTRY_EXISTS_EC (RT_LOG_BASE + 4)

An attempt was made to register a factory that already existed.

5.47.2.5 #define RT_LOG_REGISTRY_REGISTER_EC (RT_LOG_BASE + 5)

Error registering a component factory. May have exceeded RT_RegistryProperty.max_factories.

5.47.2.6 #define RT_LOG_REGISTRY_INIT_FACTORY_EC (RT_LOG_BASE + 7)

A registered factory failed to initialize.

5.47.2.7 #define RT_LOG_REGISTRY_NAME_TOO_LONG_EC (RT_LOG_BASE + 8)

Factory name longer than maximum length of 8.

5.47.2.8 #define RT_LOG_REGISTRY_INCONSISTENT_CID_EC (RT_LOG_BASE + 9)

Factory name exists, but the class ID is not of the requested type.

5.47.2.9 #define RT_LOG_REGISTRY_NOT_INITIALIZED_EC (RT_LOG_BASE + 10)

An attempt was made to operate on an uninitialized registry.

5.48 NETIO Packet API

NETIO_Packet implementation.

Data Structures

- struct NETIO_Packet

  Implementation of the abstract NETIO_Packet type.

Typedefs

- typedef struct NETIO_Packet NETIO_Packet_T

  A NETIO Packet.

Functions

- void * NETIO_Packet_get_head (const NETIO_Packet_T *const packet)

  Get a pointer to the current head.
- RTI_BOOL NETIO_Packet_set_payload (NETIO_Packet_T *const packet, void *buffer, RTI_INT32 buffer_length)

  Initialize a packet with a payload.
- RTI_SIZE_T NETIO_Packet_get_payload_length (const NETIO_Packet_T *const packet)

  Get the length the packet payload.

5.48.1 Detailed Description

NETIO_Packet implementation.

5.48.2 Typedef Documentation

5.48.2.1 typedef struct NETIO_Packet NETIO_Packet_T

A NETIO Packet.

The NETIO layers passed NETIO_Packet structures upstream/downstream.

5.48.3 Function Documentation

5.48.3.1 void* NETIO_Packet_get_head (const NETIO_Packet_T *const *packet*)

Get a pointer to the current head.

Get a pointer to the current head. Note that it can be assumed that the payload buffer is contiguous.

Parameters

packet	<<*in*>> A packet to get the head pointer for.

Returns

A pointer to the packet head.

MT Safety:

This operation is not thread safe.

5.48.3.2 RTI_BOOL NETIO_Packet_set_payload (NETIO_Packet_T *const *packet*, void * *buffer*, RTI_INT32 *buffer_length*)

Initialize a packet with a payload.

Parameters

packet	<<*in*>> The packet to initialize.
buffer	<<*in*>> The payload to initialize a packet with.
buffer_length	<<*in*>> The length of the payload in bytes.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

5.48.3.3 RTI_SIZE_T NETIO_Packet_get_payload_length (const NETIO_Packet_T *const *packet*)

Get the length the packet payload.

Parameters

packet	<<*in*>> The packet to return the payload length for.

Returns

Length of payload, 0xffffffff if the length is invalid.

MT Safety:

This operation is not thread safe.

5.49 NETIO Interface API

Data Structures

- struct NETIO_Interface

  Base-class definition for all NETIO interfaces.

Macros

- #define NETIO_Interface_receive(self_, src_, dst_, p_)

  <<eXtension>> <<cert>> *Call receive on a NETIO_Interface.*

Typedefs

- typedef struct NETIO_Interface NETIO_Interface_T
- typedef RTI_BOOL(* NETIO_Interface_sendFunc)(NETIO_Interface_T *self, struct NETIO_Interface *source, struct NETIO_Address *destination, NETIO_Packet_T *packet)

  Definition of the NETIO_InterfaceI send method.
- typedef RTI_BOOL(* NETIO_Interface_receiveFunc)(NETIO_Interface_T *netio_intf, struct NETIO_Address *src_addr, struct NETIO_Address *dst_addr, NETIO_Packet_T *packet)

  Definition of the NETIO_InterfaceI receive method.
- typedef RTI_BOOL(* NETIO_Interface_resolve_addressFunc)(NETIO_Interface_T *netio_intf, const char *address_string, struct NETIO_Address *address_value, RTI_BOOL *invalid)

  <<eXtension>> <<cert>> *Resolve an IPv4 address string to NETIO_Address if possible.*
- typedef RTI_BOOL(* NETIO_Interface_get_route_tableFunc)(NETIO_Interface_T *netio_intf, struct NETIO_AddressSeq *address, struct NETIO_NetmaskSeq *netmask)

  <<eXtension>> <<cert>> *The definition of the::NETIO_InterfaceI get_route_table method.*

Variables

- NETIO_DGRAMUserInterface_resolve_addressFunc NETIO_DGRAM_InterfaceI::resolve_address

  <<eXtension>> <<cert>> *Resolve an IPv4 address string to NETIO_Address if possible.*
- NETIO_Interface_sendFunc NETIO_DGRAM_InterfaceI::send
- NETIO_Interface_get_route_tableFunc NETIO_DGRAM_InterfaceI::get_route_table

  <<eXtension>> <<cert>> *The definition of the::NETIO_InterfaceI get_route_table method.*

5.49.1 Detailed Description

5.49.2 Macro Definition Documentation

5.49.2.1 #define NETIO_Interface_receive(*self_*, *src_*, *dst_*, *p_*)

Value:

```
((struct NETIO_InterfaceI*)(\
            (self_)->_parent._intf))->receive(self_,src_,dst_,p_)
```

<<eXtension>> <<cert>> Call receive on a NETIO_Interface.

A NETIO interface receives data from a downstream interface. The data can be in different protocol formats. Depending on the downstream protocol different actions may be taken, but semantically it does not matter where data is coming from. The receive interface is synchronous; all processing takes place in the context of the downstream receive context, such as a thread.

Parameters

*self_*	<<*in*>> The interface receiving data
*src_*	<<*in*>> The source address of the data
*dst_*	<<*in*>> The destination address of the data
*p_*	<<*in*>> A NETIO_Packet with the payload

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.49.3 Typedef Documentation

5.49.3.1 typedef struct NETIO_Interface NETIO_Interface_T

<<*eXtension*>> <<*cert*>> The abstract NETIO Interface type.

5.49.3.2 typedef RTI_BOOL(∗ NETIO_Interface_sendFunc)(NETIO_Interface_T ∗self, struct NETIO_Interface ∗source, struct NETIO_Address ∗destination, NETIO_Packet_T ∗packet)

Definition of the NETIO_InterfaceI send method.

<<*eXtension*>> <<*cert*>> The NETIO_Interface resolve_address function.

Parameters

in	*self*	NETIO interface to send from
in	*source*	The source interface for the packet
in	*destination*	The destination address for the packet
in	*packet*	The packet to send

Returns

RTI_TRUE on success, RTI_FALSE on failure

5.49.3.3 typedef RTI_BOOL(∗ NETIO_Interface_receiveFunc)(NETIO_Interface_T ∗netio_intf, struct NETIO_Address ∗src_addr, struct NETIO_Address ∗dst_addr, NETIO_Packet_T ∗packet)

Definition of the NETIO_InterfaceI receive method.

<<*eXtension*>> <<*cert*>> The NETIO_Interface send function.

Parameters

in	*netio_intf*	NETIO interface to receive in
in	*src_addr*	The source address of the packet
in	*dst_addr*	The destination address for the packet
in	*packet*	The forwarded packet from downstream

Returns

RTI_TRUE on success, RTI_FALSE on failure

See Also NETIO_Interface_receive, NETIO_InterfaceI

5.49.3.4 typedef RTI_BOOL(∗ NETIO_Interface_resolve_addressFunc)(NETIO_Interface_T ∗netio_intf, const char ∗address_string, struct NETIO_Address ∗address_value, RTI_BOOL ∗invalid)

<<*eXtension*>> <<*cert*>> Resolve an IPv4 address string to NETIO_Address if possible.

The implementation of the NETIO resolve_address function. This function is called by resolvers to determine if this interface understands the string address.

Parameters

netio_intf	<<*in*>> The interface.
address_string	<<*in*>> The address to convert.
address_value	<<*out*>> The converted address on success.
is_invalid	<<*out*>> Whether the address is valid or not.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.49.3.5 typedef RTI_BOOL(∗ NETIO_Interface_get_route_tableFunc)(NETIO_Interface_T ∗netio_intf, struct NETIO_AddressSeq ∗address, struct NETIO_NetmaskSeq ∗netmask)

<<*eXtension*>> <<*cert*>> The definition of the::NETIO_InterfaceI get_route_table method.

Parameters

netio_intf	<<*in*>> The NETIO interface.
address	<<*in*>> <<*out*>> The sequence of NETIO addresses this interface understands.
netmask	<<*in*>> <<*out*>> The sequence of the corresponding netmasks.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.49.4 Variable Documentation

5.49.4.1 NETIO_DGRAMUserInterface_resolve_addressFunc NETIO_DGRAM_InterfaceI::resolve_address

<<*eXtension*>> <<*cert*>> Resolve an IPv4 address string to NETIO_Address if possible.

The implementation of the NETIO resolve_address function. This function is called by resolvers to determine if this interface understands the string address.

Parameters

netio_intf	<<*in*>> The interface.
address_string	<<*in*>> The address to convert.
address_value	<<*out*>> The converted address on success.
is_invalid	<<*out*>> Whether the address is valid or not.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.49.4.2 NETIO_Interface_sendFunc NETIO_DGRAM_InterfaceI::send

<<*eXtension*>> <<*cert*>> The NETIO_Interface resolve_address function.

5.49.4.3 NETIO_Interface_get_route_tableFunc NETIO_DGRAM_InterfaceI::get_route_table

<<*eXtension*>> <<*cert*>> The definition of the::NETIO_InterfaceI get_route_table method.

Parameters

netio_intf	<<*in*>> The NETIO interface.
address	<<*in*>> <<*out*>> The sequence of NETIO addresses this interface understands.
netmask	<<*in*>> <<*out*>> The sequence of the corresponding netmasks.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

5.50 NETIO

Network I/O. ModuleID = 4.

Macros

- #define NETIO_LOG_GETHOST_BYNAME_EC (NETIO_LOG_BASE + 1)

  Failed to get host address from a string representation.
- #define NETIO_LOG_BIND_NEW_EC (NETIO_LOG_BASE + 2)

  Failed to allocate memory for a bind-resolver.
- #define NETIO_LOG_BIND_NEW_RTABLE_EC (NETIO_LOG_BASE + 3)

  Failed to create a database table for routes of a bind-resolver.
- #define NETIO_LOG_BIND_CREATE_RTABLE_EC (NETIO_LOG_BASE + 4)

  Failed to create route record.
- #define NETIO_LOG_BIND_DELETE_RTABLE_EC (NETIO_LOG_BASE + 5)

  Failed to delete a database table for routes of a bind_resolver.
- #define NETIO_LOG_BIND_SET_LENGTH_EC (NETIO_LOG_BASE + 6)

  Failed to set the length for a sequence of resolved addresses.
- #define NETIO_LOG_BIND_SET_MAX_EC (NETIO_LOG_BASE + 7)

  Failed to set the maximum length of an internal sequence of addresses.
- #define NETIO_LOG_BIND_EXTERNAL_FAILED_EC (NETIO_LOG_BASE + 8)

  An interface failed on bind_external.
- #define NETIO_LOG_UNBIND_EXTERNAL_FAILED_EC (NETIO_LOG_BASE + 10)

  An interface failed on unbind_external.
- #define NETIO_LOG_ROUTE_RTABLE_ALLOC_EC (NETIO_LOG_BASE + 11)

  Failed to allocate memory for a route resolver.
- #define NETIO_LOG_ROUTE_RTABLE_CREATE_EC (NETIO_LOG_BASE + 12)

  Failed to create a database table for routes of a route-resolver.
- #define NETIO_LOG_ROUTE_RTABLE_DELETE_EC (NETIO_LOG_BASE + 13)

  Failed to delete a database table for routes of a route-resolver.
- #define NETIO_LOG_ROUTE_RTABLE_ADD_EC (NETIO_LOG_BASE + 14)

  Failed to create route resolver record.
- #define NETIO_LOG_ROUTE_RTABLE_UPDATE_EC (NETIO_LOG_BASE + 15)

  Failed to find routes to update for a route-resolver.
- #define NETIO_LOG_AR_ALLOC_FAILED_EC (NETIO_LOG_BASE + 16)

  Failed to allocate memory for address resolver.
- #define NETIO_LOG_AR_ADD_INVALID_INTERFACE_EC (NETIO_LOG_BASE + 17)

  An attempt was made to add an existing interface with a different port resolver to the address resolver.
- #define NETIO_LOG_AR_ADDRESS_RESOLVE_FAILED_EC (NETIO_LOG_BASE + 18)

  An interface failed to resolve an address.
- #define NETIO_LOG_AR_PORT_RESOLVE_FAILED_EC (NETIO_LOG_BASE + 19)

  An interface failed to resolve a port.
- #define NETIO_LOG_AR_CONTEXT_UNINITIALIZED_EC (NETIO_LOG_BASE + 21)

  NETIO_AddressResolver_resolve/get_next was called with an uninitialized context.
- #define NETIO_LOG_AR_INVALID_TOKEN_EC (NETIO_LOG_BASE + 22)

  An invalid token was encountered in the address string.
- #define NETIO_LOG_AR_ADDRESS_STRING_EXCEEDED_EC (NETIO_LOG_BASE + 23)

  The length of the address exceeded the address buffer passed in.
- #define NETIO_LOG_INVALID_ADDRESS_INDEX_EC (NETIO_LOG_BASE + 24)

  An address index was not parsed correctly.
- #define NETIO_LOG_ILLEGAL_ROUTE_OPERATION_EC (NETIO_LOG_BASE + 25)

Illegal route operation attempted.

- #define NETIO_LOG_SET_NAME_EC (UDP_LOG_BASE + 26)

  Failed to set the name of a runtime component interface.
- #define NETIO_LOG_PACKET_SET_HEAD_EC (UDP_LOG_BASE + 27)

  Failed to set packet's head.
- #define NETIO_LOG_PACKET_SET_TAIL_EC (UDP_LOG_BASE + 28)

  Failed to set packet's tail.
- #define NETIO_LOG_PACKET_INITIALIZE_EC (UDP_LOG_BASE + 29)

  Failed to initialize packet.
- #define NETIO_LOG_LOOP_DELETE_TABLE_EC (NETIO_LOG_BASE + 100)

  Failed to delete a database table for a NETIO loopback interface.
- #define NETIO_LOG_LOOP_CREATE_TABLE_EC (NETIO_LOG_BASE + 101)

  Failed to create a database table for a NETIO loopback interface.
- #define NETIO_LOG_LOOP_SELECT_TABLE_EC (NETIO_LOG_BASE + 102)

  Failed to select a valid record when sending with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_FWD_EC (NETIO_LOG_BASE + 103)

  Failed to send forward with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_BINDX_DB_EC (NETIO_LOG_BASE + 104)

  Failed to bind an external interface with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_UNBINDX_DB_EC (NETIO_LOG_BASE + 105)

  Failed to unbind an external interface with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_SET_LENGTH_EC (NETIO_LOG_BASE + 106)

  Failed to set the length of an address sequence.
- #define NETIO_LOG_LOOP_INVALID_PROPERTY_EC (NETIO_LOG_BASE + 107)

  Invalid property passed in when creating a loopback interface.
- #define NETIO_LOG_LOOP_INITIALIZE_FAILED_EC (NETIO_LOG_BASE + 108)

  Failed to initialize the loopback interface.
- #define NETIO_LOG_LOOP_CURSOR_ERROR_EC (NETIO_LOG_BASE + 109)

  An error occurred with a cursor while iterating over a table.
- #define NETIO_LOG_LOOP_INVALID_FACTORY_EC (NETIO_LOG_BASE + 110)

  An invalid factory was passed in to create a loopback interface.
- #define NETIO_LOG_LOOP_INVALID_COMPONENT_EC (NETIO_LOG_BASE + 111)

  An invalid component was passed in to delete a loopback interface.
- #define UDP_LOG_SOCKET_INIT_EC (UDP_LOG_BASE + 200)

  Failed to initialize use of sockets.
- #define UDP_LOG_GETHOSTNAME_EC (UDP_LOG_BASE + 201)

  Failed to get the host address given the host name.
- #define UDP_LOG_SOCKET_SET_MCAST_EC (UDP_LOG_BASE + 202)

  Failed to set a socket for multicast loopback.
- #define UDP_LOG_SOCKET_SET_MCASTIF_EC (UDP_LOG_BASE + 203)

  Failed to set a socket for multicast bound to a specific interface.
- #define UDP_LOG_SOCKET_CREATE_EC (UDP_LOG_BASE + 204)

  Failed to create a new socket, the sysrc code is the OS return code and reason for failure.
- #define UDP_LOG_SOCKET_SETFD_EC (UDP_LOG_BASE + 205)

  Failed to set the close-on-exec flag for a socket.
- #define UDP_LOG_SOCKET_SNDBUF_EC (UDP_LOG_BASE + 206)

  Failed to set the socket send buffer size.
- #define UDP_LOG_SOCKET_TTL_EC (UDP_LOG_BASE + 207)

  Failed to set multicast TTL for a socket.
- #define UDP_LOG_PACKET_INIT_EC (UDP_LOG_BASE + 208)

  Failed to initialize a receive packet.

- #define UDP_LOG_PACKET_HEAD_EC (UDP_LOG_BASE + 209)

  Failed to set the head of a receive packet.
- #define UDP_LOG_PACKET_FWD_EC (UDP_LOG_BASE + 210)

  Failed reception upstream for a received packet.
- #define UDP_LOG_NAT_DELETE_EC (UDP_LOG_BASE + 211)

  Failed to delete a database record or index from an internal NAT.
- #define UDP_LOG_FINALIZE_EC (UDP_LOG_BASE + 212)

  Failed to finalize the parent NETIO interface.
- #define UDP_LOG_SOCKET_REUSE_PORT_EC (UDP_LOG_BASE + 213)

  Failed to set socket to reuse a port or address.
- #define UDP_LOG_SOCKET_BIND_EC (UDP_LOG_BASE + 214)

  Failed a socket bind.
- #define UDP_LOG_SOCKET_ADDGRP_EC (UDP_LOG_BASE + 215)

  Failed to add membership to a multicast group for a socket.
- #define UDP_LOG_PORT_ENTRY_EC (UDP_LOG_BASE + 216)

  Failed to create a bind entry for a receive port.
- #define UDP_LOG_PORT_ALLOC_EC (UDP_LOG_BASE + 217)

  Failed to allocate memory for a receive buffer of a specific port.
- #define UDP_LOG_SOCKET_RXBUF_EC (UDP_LOG_BASE + 218)

  Failed to set the receive buffer size of a socket.
- #define UDP_LOG_PORT_ADD_EC (UDP_LOG_BASE + 219)

  Failed to add a port entry record to a database table.
- #define UDP_LOG_PORT_THREAD_EC (UDP_LOG_BASE + 220)

  Failed to create a receive thread for a receive port.
- #define UDP_LOG_SOCKET_IFLIST_EC (UDP_LOG_BASE + 221)

  Failed to get a list of interface addresses.
- #define UDP_LOG_SOCKET_IFFLAGS_EC (UDP_LOG_BASE + 222)

  Failed to get the network interface flags or mask of a socket.
- #define UDP_LOG_SOCKET_CLOSE_EC (UDP_LOG_BASE + 223)

  Failed to close a socket.
- #define UDP_LOG_LOADLIB_EC (UDP_LOG_BASE + 224)

  Failed to load a dynamic library.
- #define UDP_LOG_GET_BUF_SIZE_EC (UDP_LOG_BASE + 225)

  Failed to get interface buffer size.
- #define UDP_LOG_GET_BUF_ALLOC_EC (UDP_LOG_BASE + 226)

  Out of memory to allocate interface buffer.
- #define UDP_LOG_GET_IFLIST_EC (UDP_LOG_BASE + 227)

  Failed to get an interface list.
- #define UDP_LOG_GET_NAMEINFO_EC (UDP_LOG_BASE + 228)

  Failed getnameinfo()
- #define UDP_LOG_MULTICAST_ENABLE_EC (UDP_LOG_BASE + 229)

  Failed to enable multicast loopback.
- #define UDP_LOG_UDP_CREATE_TABLE_EC (UDP_LOG_BASE + 230)

  Failed to create a database table for a UDP interface.
- #define UDP_LOG_UDP_CREATE_INDEX_EC (UDP_LOG_BASE + 231)

  Failed to create an index for a UDP interface.
- #define UDP_LOG_RECORD_EC (UDP_LOG_BASE + 232)

  Failed to create or insert a database record for a UDP interface.
- #define UDP_LOG_PROPERTY_FINALIZE_EC (UDP_LOG_BASE + 233)

  Failed to finalize UDP interface factory property.
- #define UDP_LOG_ALLOC_EC (UDP_LOG_BASE + 234)

Failed to allocate memory for a UDP interface.

- #define UDP_LOG_SEND_PING_EC (UDP_LOG_BASE + 235)

  Failed sending a ping message, upon adding a route or waking up a receive thread.
- #define UDP_LOG_DROPGRP_EC (UDP_LOG_BASE + 236)

  Failed to drop membership from a multicast group.
- #define UDP_LOG_GET_LENGTH_EC (UDP_LOG_BASE + 237)

  Length of address sequence exceeds its maximum.
- #define UDP_LOG_SET_LENGTH_EC (UDP_LOG_BASE + 240)

  Failed to set the length of an address sequence.
- #define UDP_LOG_WSA_STARTUP_EC (UDP_LOG_BASE + 241)

  Failed WSAStartup()
- #define UDP_LOG_WINSOCK_INCOMPATIBLE_EC (UDP_LOG_BASE + 242)

  Incompatible version of Winsock, must not be older than 2.0.
- #define UDP_LOG_INCONSISTENT_MAX_MESSSAGE_SIZE_EC (UDP_LOG_BASE + 243)

  Inconsistent max_message_size.
- #define UDP_LOG_INITIALIZE_FAILED_EC (UDP_LOG_BASE + 244)

  Failed to initialized the loopback interface.
- #define UDP_LOG_PORT_NOT_FOUND_EC (UDP_LOG_BASE + 245)

  Did not find the specified thread port.
- #define UDP_LOG_CURSOR_ERROR_EC (UDP_LOG_BASE + 246)

  An error occurred while iterating over a cursor.
- #define UDP_LOG_SEND_ERROR_EC (UDP_LOG_BASE + 247)

  Failed to send message.
- #define UDP_LOG_MULTICAST_NOT_ENABLED_EC (UDP_LOG_BASE + 248)

  Multicast address ignored because multicast supported is not compiled in.
- #define UDP_LOG_TRANSPORT_NO_INTERFACES_EC (UDP_LOG_BASE + 249)

  Transport does not have any interfaces.
- #define UDP_LOG_RECV_ERROR_EC (UDP_LOG_BASE + 250)

  Failed to receive message.

5.50.1 Detailed Description

Network I/O. ModuleID = 4.

5.50.2 Macro Definition Documentation

5.50.2.1 #define NETIO_LOG_GETHOST_BYNAME_EC (NETIO_LOG_BASE + 1)

Failed to get host address from a string representation.

5.50.2.2 #define NETIO_LOG_BIND_NEW_EC (NETIO_LOG_BASE + 2)

Failed to allocate memory for a bind-resolver.

5.50.2.3 #define NETIO_LOG_BIND_NEW_RTABLE_EC (NETIO_LOG_BASE + 3)

Failed to create a database table for routes of a bind-resolver.

5.50.2.4 #define NETIO_LOG_BIND_CREATE_RTABLE_EC (NETIO_LOG_BASE + 4)

Failed to create route record.

May have exceeded DomainParticipantQos.resource_limits.max_receive_ports or DomainParticipantQos.resource_limits.local_writer_allocation

5.50.2.5 #define NETIO_LOG_BIND_DELETE_RTABLE_EC (NETIO_LOG_BASE + 5)

Failed to delete a database table for routes of a bind_resolver.

5.50.2.6 #define NETIO_LOG_BIND_SET_LENGTH_EC (NETIO_LOG_BASE + 6)

Failed to set the length for a sequence of resolved addresses.

5.50.2.7 #define NETIO_LOG_BIND_SET_MAX_EC (NETIO_LOG_BASE + 7)

Failed to set the maximum length of an internal sequence of addresses.

5.50.2.8 #define NETIO_LOG_BIND_EXTERNAL_FAILED_EC (NETIO_LOG_BASE + 8)

An interface failed on bind_external.

5.50.2.9 #define NETIO_LOG_UNBIND_EXTERNAL_FAILED_EC (NETIO_LOG_BASE + 10)

An interface failed on unbind_external.

5.50.2.10 #define NETIO_LOG_ROUTE_RTABLE_ALLOC_EC (NETIO_LOG_BASE + 11)

Failed to allocate memory for a route resolver.

5.50.2.11 #define NETIO_LOG_ROUTE_RTABLE_CREATE_EC (NETIO_LOG_BASE + 12)

Failed to create a database table for routes of a route-resolver.

5.50.2.12 #define NETIO_LOG_ROUTE_RTABLE_DELETE_EC (NETIO_LOG_BASE + 13)

Failed to delete a database table for routes of a route-resolver.

5.50.2.13 #define NETIO_LOG_ROUTE_RTABLE_ADD_EC (NETIO_LOG_BASE + 14)

Failed to create route resolver record.
May have exceeded DomainParticipantQos.resource_limits.max_destination_ports

5.50.2.14 #define NETIO_LOG_ROUTE_RTABLE_UPDATE_EC (NETIO_LOG_BASE + 15)

Failed to find routes to update for a route-resolver.

5.50.2.15 #define NETIO_LOG_AR_ALLOC_FAILED_EC (NETIO_LOG_BASE + 16)

Failed to allocate memory for address resolver.

5.50.2.16 #define NETIO_LOG_AR_ADD_INVALID_INTERFACE_EC (NETIO_LOG_BASE + 17)

An attempt was made to add an existing interface with a different port resolver to the address resolver.

5.50.2.17 #define NETIO_LOG_AR_ADDRESS_RESOLVE_FAILED_EC (NETIO_LOG_BASE + 18)

An interface failed to resolve an address.

5.50.2.18 #define NETIO_LOG_AR_PORT_RESOLVE_FAILED_EC (NETIO_LOG_BASE + 19)

An interface failed to resolve a port.

5.50.2.19 #define NETIO_LOG_AR_CONTEXT_UNINITIALIZED_EC (NETIO_LOG_BASE + 21)

NETIO_AddressResolver_resolve/get_next was called with an uninitialized context.

5.50.2.20 #define NETIO_LOG_AR_INVALID_TOKEN_EC (NETIO_LOG_BASE + 22)

An invalid token was encountered in the address string.

5.50.2.21 #define NETIO_LOG_AR_ADDRESS_STRING_EXCEEDED_EC (NETIO_LOG_BASE + 23)

The length of the address exceeded the address buffer passed in.

5.50.2.22 #define NETIO_LOG_INVALID_ADDRESS_INDEX_EC (NETIO_LOG_BASE + 24)

An address index was not parsed correctly.

When parsing an address string the index was not specified correctly. Legal formats are: N@ [,N]@ [N]@ [M,N]@

Where N and M are integers >= 0

5.50.2.23 #define NETIO_LOG_ILLEGAL_ROUTE_OPERATION_EC (NETIO_LOG_BASE + 25)

Illegal route operation attempted.

5.50.2.24 #define NETIO_LOG_SET_NAME_EC (UDP_LOG_BASE + 26)

Failed to set the name of a runtime component interface.

5.50.2.25 #define NETIO_LOG_PACKET_SET_HEAD_EC (UDP_LOG_BASE + 27)

Failed to set packet's head.

NETIO_Packet_set_head() failed, due to an invalid delta that would have moved the head position out-of-bounds of the packet.

5.50.2.26 #define NETIO_LOG_PACKET_SET_TAIL_EC (UDP_LOG_BASE + 28)

Failed to set packet's tail.

NETIO_Packet_set_tail() failed, due to an invalid delta that would have moved the tail position out-of-bounds of the packet.

5.50.2.27 #define NETIO_LOG_PACKET_INITIALIZE_EC (UDP_LOG_BASE + 29)

Failed to initialize packet.

NETIO_Packet_initialize() failed due to invalid arguments

5.50.2.28 #define NETIO_LOG_LOOP_DELETE_TABLE_EC (NETIO_LOG_BASE + 100)

Failed to delete a database table for a NETIO loopback interface.

5.50.2.29 #define NETIO_LOG_LOOP_CREATE_TABLE_EC (NETIO_LOG_BASE + 101)

Failed to create a database table for a NETIO loopback interface.

5.50.2.30 #define NETIO_LOG_LOOP_SELECT_TABLE_EC (NETIO_LOG_BASE + 102)

Failed to select a valid record when sending with the NETIO loopback interface.

5.50.2.31 #define NETIO_LOG_LOOP_FWD_EC (NETIO_LOG_BASE + 103)

Failed to send forward with the NETIO loopback interface.

5.50.2.32 #define NETIO_LOG_LOOP_BINDX_DB_EC (NETIO_LOG_BASE + 104)

Failed to bind an external interface with the NETIO loopback interface.

5.50.2.33 #define NETIO_LOG_LOOP_UNBINDX_DB_EC (NETIO_LOG_BASE + 105)

Failed to unbind an external interface with the NETIO loopback interface.

5.50.2.34 #define NETIO_LOG_LOOP_SET_LENGTH_EC (NETIO_LOG_BASE + 106)

Failed to set the length of an address sequence.

5.50.2.35 #define NETIO_LOG_LOOP_INVALID_PROPERTY_EC (NETIO_LOG_BASE + 107)

Invalid property passed in when creating a loopback interface.

5.50.2.36 #define NETIO_LOG_LOOP_INITIALIZE_FAILED_EC (NETIO_LOG_BASE + 108)

Failed to initialize the loopback interface.

5.50.2.37 #define NETIO_LOG_LOOP_CURSOR_ERROR_EC (NETIO_LOG_BASE + 109)

An error occurred with a cursor while iterating over a table.

5.50.2.38 #define NETIO_LOG_LOOP_INVALID_FACTORY_EC (NETIO_LOG_BASE + 110)

An invalid factory was passed in to create a loopback interface.

5.50.2.39 #define NETIO_LOG_LOOP_INVALID_COMPONENT_EC (NETIO_LOG_BASE + 111)

An invalid component was passed in to delete a loopback interface.

5.50.2.40 #define UDP_LOG_SOCKET_INIT_EC (UDP_LOG_BASE + 200)

Failed to initialize use of sockets.

5.50.2.41 #define UDP_LOG_GETHOSTNAME_EC (UDP_LOG_BASE + 201)

Failed to get the host address given the host name.

5.50.2.42 #define UDP_LOG_SOCKET_SET_MCAST_EC (UDP_LOG_BASE + 202)

Failed to set a socket for multicast loopback.

5.50.2.43 #define UDP_LOG_SOCKET_SET_MCASTIF_EC (UDP_LOG_BASE + 203)

Failed to set a socket for multicast bound to a specific interface.

5.50.2.44 #define UDP_LOG_SOCKET_CREATE_EC (UDP_LOG_BASE + 204)

Failed to create a new socket, the sysrc code is the OS return code and reason for failure.

5.50.2.45 #define UDP_LOG_SOCKET_SETFD_EC (UDP_LOG_BASE + 205)

Failed to set the close-on-exec flag for a socket.

5.50.2.46 #define UDP_LOG_SOCKET_SNDBUF_EC (UDP_LOG_BASE + 206)

Failed to set the socket send buffer size.

5.50.2.47 #define UDP_LOG_SOCKET_TTL_EC (UDP_LOG_BASE + 207)

Failed to set multicast TTL for a socket.

5.50.2.48 #define UDP_LOG_PACKET_INIT_EC (UDP_LOG_BASE + 208)

Failed to initialize a receive packet.

5.50.2.49 #define UDP_LOG_PACKET_HEAD_EC (UDP_LOG_BASE + 209)

Failed to set the head of a receive packet.

5.50.2.50 #define UDP_LOG_PACKET_FWD_EC (UDP_LOG_BASE + 210)

Failed reception upstream for a received packet.

5.50.2.51 #define UDP_LOG_NAT_DELETE_EC (UDP_LOG_BASE + 211)

Failed to delete a database record or index from an internal NAT.

5.50.2.52 #define UDP_LOG_FINALIZE_EC (UDP_LOG_BASE + 212)

Failed to finalize the parent NETIO interface.

5.50.2.53 #define UDP_LOG_SOCKET_REUSE_PORT_EC (UDP_LOG_BASE + 213)

Failed to set socket to reuse a port or address.

5.50.2.54 #define UDP_LOG_SOCKET_BIND_EC (UDP_LOG_BASE + 214)

Failed a socket bind.

5.50.2.55 #define UDP_LOG_SOCKET_ADDGRP_EC (UDP_LOG_BASE + 215)

Failed to add membership to a multicast group for a socket.

5.50.2.56 #define UDP_LOG_PORT_ENTRY_EC (UDP_LOG_BASE + 216)

Failed to create a bind entry for a receive port.

5.50.2.57 #define UDP_LOG_PORT_ALLOC_EC (UDP_LOG_BASE + 217)

Failed to allocate memory for a receive buffer of a specific port.

5.50.2.58 #define UDP_LOG_SOCKET_RXBUF_EC (UDP_LOG_BASE + 218)

Failed to set the receive buffer size of a socket.

5.50.2.59 #define UDP_LOG_PORT_ADD_EC (UDP_LOG_BASE + 219)

Failed to add a port entry record to a database table.

5.50.2.60 #define UDP_LOG_PORT_THREAD_EC (UDP_LOG_BASE + 220)

Failed to create a receive thread for a receive port.

5.50.2.61 #define UDP_LOG_SOCKET_IFLIST_EC (UDP_LOG_BASE + 221)

Failed to get a list of interface addresses.

5.50.2.62 #define UDP_LOG_SOCKET_IFFLAGS_EC (UDP_LOG_BASE + 222)

Failed to get the network interface flags or mask of a socket.

5.50.2.63 #define UDP_LOG_SOCKET_CLOSE_EC (UDP_LOG_BASE + 223)

Failed to close a socket.

5.50.2.64 #define UDP_LOG_LOADLIB_EC (UDP_LOG_BASE + 224)

Failed to load a dynamic library.

5.50.2.65 #define UDP_LOG_GET_BUF_SIZE_EC (UDP_LOG_BASE + 225)

Failed to get interface buffer size.

5.50.2.66 #define UDP_LOG_GET_BUF_ALLOC_EC (UDP_LOG_BASE + 226)

Out of memory to allocate interface buffer.

5.50.2.67 #define UDP_LOG_GET_IFLIST_EC (UDP_LOG_BASE + 227)

Failed to get an interface list.

5.50.2.68 #define UDP_LOG_GET_NAMEINFO_EC (UDP_LOG_BASE + 228)

Failed getnameinfo()

5.50.2.69 #define UDP_LOG_MULTICAST_ENABLE_EC (UDP_LOG_BASE + 229)

Failed to enable multicast loopback.

5.50.2.70 #define UDP_LOG_UDP_CREATE_TABLE_EC (UDP_LOG_BASE + 230)

Failed to create a database table for a UDP interface.

5.50.2.71 #define UDP_LOG_UDP_CREATE_INDEX_EC (UDP_LOG_BASE + 231)

Failed to create an index for a UDP interface.

5.50.2.72 #define UDP_LOG_RECORD_EC (UDP_LOG_BASE + 232)

Failed to create or insert a database record for a UDP interface.

5.50.2.73 #define UDP_LOG_PROPERTY_FINALIZE_EC (UDP_LOG_BASE + 233)

Failed to finalize UDP interface factory property.

5.50.2.74 #define UDP_LOG_ALLOC_EC (UDP_LOG_BASE + 234)

Failed to allocate memory for a UDP interface.

5.50.2.75 #define UDP_LOG_SEND_PING_EC (UDP_LOG_BASE + 235)

Failed sending a ping message, upon adding a route or waking up a receive thread.

5.50.2.76 #define UDP_LOG_DROPGRP_EC (UDP_LOG_BASE + 236)

Failed to drop membership from a multicast group.

5.50.2.77 #define UDP_LOG_GET_LENGTH_EC (UDP_LOG_BASE + 237)

Length of address sequence exceeds its maximum.

5.50.2.78 #define UDP_LOG_SET_LENGTH_EC (UDP_LOG_BASE + 240)

Failed to set the length of an address sequence.

5.50.2.79 #define UDP_LOG_WSA_STARTUP_EC (UDP_LOG_BASE + 241)

Failed WSAStartup()

5.50.2.80 #define UDP_LOG_WINSOCK_INCOMPATIBLE_EC (UDP_LOG_BASE + 242)

Incompatible version of Winsock, must not be older than 2.0.

5.50.2.81 #define UDP_LOG_INCONSISTENT_MAX_MESSSAGE_SIZE_EC (UDP_LOG_BASE + 243)

Inconsistent max_message_size.

UDP_InterfaceProperty.max_message_size is set to an inconsistent value, larger than UDP_InterfaceProperty.max_send_buffer_size and/or larger than UDP_InterfaceProperty.max_receive_buffer_size .

5.50.2.82 #define UDP_LOG_INITIALIZE_FAILED_EC (UDP_LOG_BASE + 244)

Failed to initialized the loopback interface.

5.50.2.83 #define UDP_LOG_PORT_NOT_FOUND_EC (UDP_LOG_BASE + 245)

Did not find the specified thread port.

5.50.2.84 #define UDP_LOG_CURSOR_ERROR_EC (UDP_LOG_BASE + 246)

An error occurred while iterating over a cursor.

5.50.2.85 #define UDP_LOG_SEND_ERROR_EC (UDP_LOG_BASE + 247)

Failed to send message.

A message could not be sent due to system error

5.50.2.86 #define UDP_LOG_MULTICAST_NOT_ENABLED_EC (UDP_LOG_BASE + 248)

Multicast address ignored because multicast supported is not compiled in.

A multicast address was specified as an address, but multicast support has not been enabled with NETIO_CONFIG_ENABLE_MULTICAST in netio_config.h

5.50.2.87 #define UDP_LOG_TRANSPORT_NO_INTERFACES_EC (UDP_LOG_BASE + 249)

Transport does not have any interfaces.

UDP configuration might be wrong.

5.50.2.88 #define UDP_LOG_RECV_ERROR_EC (UDP_LOG_BASE + 250)

Failed to receive message.

A message could not be received due to system error

5.51 RTPS

Real-Time Publish-Subscribe. ModuleID = 6.

Macros

- #define RTPS_LOG_INITIALIZE_INTERFACE_EC (RTPS_LOG_BASE + 1)

  Failed to initialize an RTPS interface.
- #define RTPS_LOG_ALLOCATE_EC (RTPS_LOG_BASE + 2)

  Failed to allocate heap memory for internal resources.
- #define RTPS_LOG_CREATE_ROUTE_TABLE_EC (RTPS_LOG_BASE + 3)

  Failed to create a database table for storing RTPS route info.
- #define RTPS_LOG_DB_CREATE_ROUTE_PEER_INDEX_EC (RTPS_LOG_BASE + 4)

  Failed to create database index for route-peer info.
- #define RTPS_LOG_DB_CREATE_ROUTE_XPORT_INDEX_EC (RTPS_LOG_BASE + 5)

  Failed to create database index for route-transport info.
- #define RTPS_LOG_DB_CREATE_BIND_PEER_INDEX_EC (RTPS_LOG_BASE + 6)

  Failed to create database index for bind-peer info.
- #define RTPS_LOG_INDEX_ADD_ENTRY_EC (RTPS_LOG_BASE + 8)

  Failed to add an entry to a database index.
- #define RTPS_LOG_DB_SELECT_ALL_EC (RTPS_LOG_BASE + 9)

  Failed to select all entries of a database table.
- #define RTPS_LOG_DB_SELECT_MATCH_EC (RTPS_LOG_BASE + 10)

  Matching entry not found in a database table.
- #define RTPS_LOG_DB_SELECT_RANGE_EC (RTPS_LOG_BASE + 11)

  No matching entries found in a database table.
- #define RTPS_LOG_DB_CREATE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 12)

  Failed to create database entry for route.
- #define RTPS_LOG_DB_INSERT_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 13)

  Failed to insert entry into route table.
- #define RTPS_LOG_DB_DELETE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 15)

  Failed to delete entry from route table.
- #define RTPS_LOG_DB_CREATE_BIND_ENTRY_EC (RTPS_LOG_BASE + 16)

  Failed to create database entry for bind.
- #define RTPS_LOG_DB_INSERT_BIND_ENTRY_EC (RTPS_LOG_BASE + 17)

  Failed to insert an entry into the bind table.
- #define RTPS_LOG_DB_REMOVE_BIND_ENTRY_EC (RTPS_LOG_BASE + 18)

  Failed to remove an entry from the bind table.
- #define RTPS_LOG_DB_REMOVE_EXTERNAL_BIND_ENTRY_EC (RTPS_LOG_BASE + 19)

  Failed to remove an entry from the external bind table.
- #define RTPS_LOG_DB_DELETE_EXTERNAL_BIND_ENTRY_EC (RTPS_LOG_BASE + 20)

  Failed to delete entry from external bind table.
- #define RTPS_LOG_SEND_EC (RTPS_LOG_BASE + 21)

  Failed to send a packet due to a lower module failing to send the packet.
- #define RTPS_LOG_RECEIVE_EC (RTPS_LOG_BASE + 22)

  Failed to receive a packet due to a higher module failing to receive the packet.
- #define RTPS_LOG_ROUTE_PACKET_EC (RTPS_LOG_BASE + 23)

  Failed to send a packet, when routing to a lower module or peer.
- #define RTPS_LOG_FORWARD_UPSTREAM_EC (RTPS_LOG_BASE + 24)

  Failed to receive a packet, when forwarding to a higher upstream module.
- #define RTPS_LOG_BAD_PARAMETER_EC (RTPS_LOG_BASE + 25)

Bad parameter to an RTPS interface function.

- #define RTPS_LOG_NOT_ENABLED_EC (RTPS_LOG_BASE + 26)

  Failed because interface is not enabled.
- #define RTPS_LOG_READER_UNSUPPORTED_EC (RTPS_LOG_BASE + 27)

  RTPS reader does not support send.
- #define RTPS_LOG_INTERFACE_MISMATCH_EC (RTPS_LOG_BASE + 28)

  Upstream interface does not match source or destination of packet being sent or received.
- #define RTPS_LOG_FULL_SEND_WINDOW_EC (RTPS_LOG_BASE + 29)

  Failed to send due to reaching maximum send window size.
- #define RTPS_LOG_NETIO_PACKET_SET_TAIL_EC (RTPS_LOG_BASE + 31)

  Failed to set tail of packet to send.
- #define RTPS_LOG_FUNC_UNSUPPORTED_EC (RTPS_LOG_BASE + 32)

  RTPS interface does not support this function.
- #define RTPS_LOG_EXCEEDED_LIMIT_TRANSPORTS_EC (RTPS_LOG_BASE + 33)

  Out of transport entries.
- #define RTPS_LOG_EXCEEDED_LIMIT_PEERS_EC (RTPS_LOG_BASE + 34)

  Out of peer entries.
- #define RTPS_LOG_ASSERT_PEER_EC (RTPS_LOG_BASE + 35)

  Failed to assert a remote writer or reader.
- #define RTPS_LOG_ASSERT_TRANSPORT_EC (RTPS_LOG_BASE + 36)

  Failed to assert a downstream transport.
- #define RTPS_LOG_FIND_EXTERNAL_INTERFACE_EC (RTPS_LOG_BASE + 37)

  Failed to lookup an external interface.
- #define RTPS_LOG_NONEXISTENT_ROUTE_EC (RTPS_LOG_BASE + 38)

  Failed to delete a nonexistent route.
- #define RTPS_LOG_NONEXISTENT_BIND_EC (RTPS_LOG_BASE + 39)

  Failed to remove a nonexistent bind entry.
- #define RTPS_LOG_NONEXISTENT_EXTERNAL_BIND_EC (RTPS_LOG_BASE + 40)

  Failed to remove a nonexistent external bind entry.
- #define RTPS_LOG_STALE_ACK_EPOCH__EC (RTPS_LOG_BASE + 41)

  Dropped an ACKNACK submessage with an old epoch.
- #define RTPS_LOG_STALE_HB_EPOCH_EC (RTPS_LOG_BASE + 42)

  Dropped a HEARTBEAT submessage with an old epoch.
- #define RTPS_LOG_ACK_EC (RTPS_LOG_BASE + 43)

  Failed an acknack() upstream.
- #define RTPS_LOG_REQUEST_EC (RTPS_LOG_BASE + 45)

  Failed a request() upstream.
- #define RTPS_LOG_RETURN_LOAN_EC (RTPS_LOG_BASE + 46)

  Failed a return_loan() upstream.
- #define RTPS_LOG_NETIO_PACKET_SET_HEAD_EC (RTPS_LOG_BASE + 47)

  Failed to set the head of a packet.
- #define RTPS_LOG_SEND_HEARTBEAT_EC (RTPS_LOG_BASE + 48)

  Failed to send a HEARTBEAT submessage.
- #define RTPS_LOG_SHIFT_BITMAP_EC (RTPS_LOG_BASE + 49)

  Failed to shift a bitmap.
- #define RTPS_LOG_FULLY_ACKED_READER_EC (RTPS_LOG_BASE + 50)

  A Reader is fully acknowledged and does not need to respond to a final HEARTBEAT.
- #define RTPS_LOG_DATA_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 51)

  Dropping a DATA submessage whose sequence number is outside of a Reader's receive window.
- #define RTPS_LOG_DATA_ALREADY_RECEIVED_EC (RTPS_LOG_BASE + 52)

  Dropping a DATA submessage whose sequence number was previously received.

- #define RTPS_LOG_INTERFACE_NOT_ENABLED_EC (RTPS_LOG_BASE + 53)

  Failed to receive a message because interface is not enabled.
- #define RTPS_LOG_INVALID_PACKET_EC (RTPS_LOG_BASE + 54)

  Dropped a message with an invalid packet header.
- #define RTPS_LOG_UNSUPPORTED_INTERFACE_EC (RTPS_LOG_BASE + 55)

  Received a message on an unsupported interface.
- #define RTPS_LOG_UNKNOWN_SUBMESSAGE_EC (RTPS_LOG_BASE + 56)

  Received a submessage with an unknown ID.
- #define RTPS_LOG_PROCESS_ACKNACK_EC (RTPS_LOG_BASE + 57)

  Failed to process received ACKNACK submessage.
- #define RTPS_LOG_PROCESS_DATA_EC (RTPS_LOG_BASE + 58)

  Failed to process received DATA submessage.
- #define RTPS_LOG_PROCESS_GAP_EC (RTPS_LOG_BASE + 59)

  Failed to process received GAP submessage.
- #define RTPS_LOG_PROCESS_HEARTBEAT_EC (RTPS_LOG_BASE + 60)

  Failed to process received HEARTBEAT submessage.
- #define RTPS_LOG_CREATE_HB_EVENT_EC (RTPS_LOG_BASE + 61)

  Failed to create periodic HEARTBEAT event.
- #define RTPS_LOG_CREATE_EVENT_TIMER_EC (RTPS_LOG_BASE + 62)

  Failed to create periodic HEARTBEAT event timer.
- #define RTPS_LOG_SEQ_NUM_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 63)

  Failed to set bitmap bit due to out-of-range sequence number.
- #define RTPS_LOG_FIRST_SN_GREATER_LAST_SN_EC (RTPS_LOG_BASE + 65)

  Invalid range of sequence numbers to fill in bitmap.
- #define RTPS_LOG_BITCOUNT_OUT_OF_BOUNDS_EC (RTPS_LOG_BASE + 66)

  Deserialized an invalid out-of-bounds bitcount for a bitmap.
- #define RTPS_LOG_SHIFT_SN_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 67)

  Failed to shift bitmap due to invalid starting sequence number.
- #define RTPS_LOG_SERIALIZE_HOST_ID_EC (RTPS_LOG_BASE + 68)

  Failed to serialize host ID of GUID.
- #define RTPS_LOG_SERIALIZE_APP_ID_EC (RTPS_LOG_BASE + 69)

  Failed to serialize app ID of GUID.
- #define RTPS_LOG_SERIALIZE_INSTANCE_ID_EC (RTPS_LOG_BASE + 70)

  Failed to serialized instance ID of GUID.
- #define RTPS_LOG_SERIALIZE_OBJECT_ID_EC (RTPS_LOG_BASE + 71)

  Failed to serialize object ID of GUID.
- #define RTPS_LOG_DESERIALIZE_HOST_ID_EC (RTPS_LOG_BASE + 72)

  Failed to deserialize host ID of GUID.
- #define RTPS_LOG_DESERIALIZE_APP_ID_EC (RTPS_LOG_BASE + 73)

  Faild to deserialize app ID of GUID.
- #define RTPS_LOG_DESERIALIZE_INSTANCE_ID_EC (RTPS_LOG_BASE + 74)

  Failed to deserialize instance ID of GUID.
- #define RTPS_LOG_DESERIALIZE_OBJECT_ID_EC (RTPS_LOG_BASE + 75)

  Failed to deserialize object ID of GUID.
- #define RTPS_LOG_DB_CREATE_EXT_BIND_ENTRY_EC (RTPS_LOG_BASE + 76)

  Failed to create database entry for external bind table.
- #define RTPS_LOG_BITMAP_SET_BIT_EC (RTPS_LOG_BASE + 77)

  Failed to set bit in bitmap.
- #define RTPS_LOG_UNSUPPORTED_INFO_REPLY_EC (RTPS_LOG_BASE + 78)

  Received and dropped currently unsupported INFO_REPLY submessage.
- #define RTPS_LOG_UNSUPPORTED_INFO_REPLY_IP4_EC (RTPS_LOG_BASE + 79)

Received and dropped currently unsupported INFO_REPLY_IP4 submessage.

- #define RTPS_LOG_UNSUPPORTED_INFO_SRC_EC (RTPS_LOG_BASE + 80)

  Received and dropped currently unsupported INFO_SRC submessage.
- #define RTPS_LOG_DELETE_INDEX_EC (RTPS_LOG_BASE + 81)

  Failed to delete index of a database table.
- #define RTPS_LOG_DELETE_TABLE_EC (RTPS_LOG_BASE + 82)

  Failed to delete a database table.
- #define RTPS_LOG_DB_LOCK_EC (RTPS_LOG_BASE + 83)

  Failed to take database lock.
- #define RTPS_LOG_DB_UNLOCK_EC (RTPS_LOG_BASE + 84)

  Failed to give database lock.
- #define RTPS_LOG_CREATE_BIND_TABLE_EC (RTPS_LOG_BASE + 85)

  Failed to create a database table for storing RTPS bind info.
- #define RTPS_LOG_STATUS_CHANGE_EC (RTPS_LOG_BASE + 86)

  Failed to update status for reliable reader activity changed.
- #define RTPS_LOG_SET_GAP_EC (RTPS_LOG_BASE + 87)

  Failed to update status for reliable reader activity changed.
- #define RTPS_LOG_WINDOW_POOL_ALLOC_EC (RTPS_LOG_BASE + 88)

  Out of memory to allocate send window pool.
- #define RTPS_LOG_GET_WINDOW_BUFFER_EC (RTPS_LOG_BASE + 89)

  Failed to get buffer from send window pool.
- #define RTPS_LOG_INSERT_WINDOW_FULL_EC (RTPS_LOG_BASE + 90)

  Cannot insert sample into full window.
- #define RTPS_LOG_WINDOW_INSERT_EC (RTPS_LOG_BASE + 91)

  Failed to insert sample into send window.
- #define RTPS_LOG_TIMER_DELETE_TIMEOUT_EC (RTPS_LOG_BASE + 92)

  Failed to delete a timeout event.
- #define RTPS_LOG_BUFFERPOOL_DELETE_EC (RTPS_LOG_BASE + 93)

  Failed to delete a buffer pool.
- #define RTPS_LOG_DIRECT_SEND_EC (RTPS_LOG_BASE + 94)

  Failed to send a message to a specific peer.
- #define RTPS_LOG_PACKET_INITIALIZE_EC (RTPS_LOG_BASE + 95)

  RTPS writer failed to initialize a packet.
- #define RTPS_LOG_WRITER_REQUEST_SENT_HISTORY_EC (RTPS_LOG_BASE + 96)

  RTPS reliable writer failed to request and resend history.
- #define RTPS_LOG_BITMAP_SHIFT_EC (RTPS_LOG_BASE + 97)

  Failed to shift bitmap to new lead sequence number.
- #define RTPS_LOG_WINDOW_ADVANCE_EC (RTPS_LOG_BASE + 98)

  Send window of a remote reader failed to advance.
- #define RTPS_LOG_WINDOW_ADVANCE_UNACKED_EC (RTPS_LOG_BASE + 99)

  Failed when advancing window ahead of unacknowledged sequence number.
- #define RTPS_LOG_LOOKUP_PEER_EC (RTPS_LOG_BASE + 100)

  Failed to find peer that should exist.
- #define RTPS_LOG_READER_SEND_ACKNACK_EC (RTPS_LOG_BASE + 101)

  RTPS reader failed to send an ACKNACK message.
- #define RTPS_LOG_FINALIZE_INTERFACE_EC (RTPS_LOG_BASE + 102)

  Failed to finalize an RTPS interface.
- #define RTPS_LOG_INDEXER_REMOVE_ENTRY_EC (RTPS_LOG_BASE + 103)

  Failed to remove index entry for external interface upon deleting RTPS interface.
- #define RTPS_LOG_LOOKUP_EXT_BIND_ENTRY_EC (RTPS_LOG_BASE + 104)

  Failed when looking up external bind entry from database.

- #define RTPS_LOG_SEQ_INIT_EC (RTPS_LOG_BASE + 105)

  Failed to initialize a local sequence.
- #define RTPS_LOG_SEQ_MAX_EC (RTPS_LOG_BASE + 106)

  Failed to set maximum of a local sequence.
- #define RTPS_LOG_SEQ_LEN_EC (RTPS_LOG_BASE + 107)

  Failed to set length of a local sequence.
- #define RTPS_LOG_INTF_MODE_UNDEF_EC (RTPS_LOG_BASE + 108)

  Initializing an RTPS interface with undefined mode.
- #define RTPS_LOG_DELETE_BUFFER_POOL_EC (RTPS_LOG_BASE + 109)

  Failed to delete buffer pool.
- #define RTPS_LOG_CREATE_BUFFER_POOL_EC (RTPS_LOG_BASE + 110)

  Failed to create buffer pool.
- #define RTPS_LOG_CREATE_PEERS_INDEX_EC (RTPS_LOG_BASE + 111)

  Failed to create index of peers.
- #define RTPS_LOG_DELETE_INDEXER_EC (RTPS_LOG_BASE + 112)

  Failed to delete indexer.
- #define RTPS_LOG_GET_PEER_BUFFER_EC (RTPS_LOG_BASE + 113)

  Failed to get buffer from peer buffer pool.
- #define RTPS_LOG_DB_CREATE_DIRECT_INDEX_EC (RTPS_LOG_BASE + 114)

  Failed to create database index for direct sends.
- #define RTPS_LOG_DB_CREATE_GROUP_INDEX_EC (RTPS_LOG_BASE + 115)

  Failed to create database index for group sends.
- #define RTPS_LOG_NETIO_PACKET_INSUFFICIENT_BUFFER_EC (RTPS_LOG_BASE + 116)

  Insufficient buffer in packet.
- #define RTPS_LOG_PROCESS_DATA_BATCH_EC (RTPS_LOG_BASE + 117)

  Failed to process received DATA BATCH submessage.
- #define RTPS_LOG_PROCESS_HEARTBEAT_BATCH_EC (RTPS_LOG_BASE + 118)

  Failed to process received HEARTBEAT_BATCH submessage.
- #define RTPS_LOG_NEGATIVE_RESERVATION_COUNT_EC (RTPS_LOG_BASE + 119)

  The reservation count it zero.
- #define RTPS_LOG_DB_CREATE_SELECTED_GROUP_INDEX_EC (RTPS_LOG_BASE + 148)

  Failed to create database index for the selected route group.
- #define RTPS_LOG_CHECKSUM_MISSING_CHECKSUM_LENGTH_EC (RTPS_LOG_BASE + 149)

  A header extension was received, but the checksum length is not present.
- #define RTPS_LOG_CHECKSUM_UNSUPPORTED_EC (RTPS_LOG_BASE + 150)

  Unsupported checksum bits received, message dropped.
- #define RTPS_LOG_CHECKSUM_LENGTH_DESERIALIZE_ERROR_EC (RTPS_LOG_BASE + 151)

  Checksum length could not be deserialized.
- #define RTPS_LOG_CHECKSUM_INCONSISTENT_LENGTH_EC (RTPS_LOG_BASE + 152)

  The payload length is less than the deserialized checksum length.
- #define RTPS_LOG_CHECKSUM_CHECKSUM_DESERIALIZE_ERROR_EC (RTPS_LOG_BASE + 153)

  Checksum length could not be deserialized.
- #define RTPS_LOG_CHECKSUM_CHECKSUM_ERROR_EC (RTPS_LOG_BASE + 154)

  The checksum is invalid.
- #define RTPS_LOG_CHECKSUM_INVALID_CRC32_FLAGS_EC (RTPS_LOG_BASE + 155)

  The CRC32 submsg flags are invalid.
- #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM128_EC (RTPS_LOG_BASE + 200)

  The property tries to override the BUILTIN128 checksum without setting allow_builtin_override to TRUE.
- #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM32_EC (RTPS_LOG_BASE + 201)

  The property tries to override the BUILTIN32 checksum without setting allow_builtin_override to TRUE.

- #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM64_EC (RTPS_LOG_BASE + 202)

  The property tries to override the BUILTIN64 checksum without setting allow_builtin_override to TRUE.
- #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN128_EC (RTPS_LOG_BASE + 203)

  The property override of the BUILTIN128 checksum is invalid. It is not legal to reuse any of the built-in functions.
- #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN32_EC (RTPS_LOG_BASE + 204)

  The property override of the BUILTIN32 checksum is invalid. It is not legal to reuse any of the built-in functions functions.
- #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN64_EC (RTPS_LOG_BASE + 205)

  The property override of the BUILTIN64 checksum is invalid. It is not legal to reuse any of the built-in functions functions.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM256_EC (RTPS_LOG_BASE + 206)

  The property specifies an invalid custom256 checksum function.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM32_EC (RTPS_LOG_BASE + 207)

  The property specifies an invalid custom32 checksum function.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM64_EC (RTPS_LOG_BASE + 208)

  The property specifies an invalid custom64 checksum function.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM128_EC (RTPS_LOG_BASE + 209)

  The property specifies an invalid custom128 checksum function.
- #define RTPS_LOG_CHECKSUM_CHECKSUM_CALCULATION_FAILED_EC (RTPS_LOG_BASE + 210)

  RTPS_Interface_calculate_checksum() failed to calculate the checksum.
- #define RTPS_LOG_UNSUPPORTED_MANDTORY_HDR_EXT_PID_EC (RTPS_LOG_BASE + 211)

  The HeaderExtension contained a mandatory PID that was not understood.
- #define RTPS_LOG_INVALID_HDR_EXT_PID_LIST_EC (RTPS_LOG_BASE + 212)

  An invalid HeaderExtension PID was detected.
- #define RTPS_LOG_INVALID_HDR_EXT_PID_ERROR_EC (RTPS_LOG_BASE + 213)

  Failed to decode the HeaderExtension PID list.
- #define RTPS_LOG_INCONSISTENT_HDR_EXT_LENGTH_ERROR_EC (RTPS_LOG_BASE + 214)

  The HeaderExtension length is isconsistent with the decoded encoded HeaderExtension length.
- #define RTPS_LOG_MESSAGE_EXCEED_LENGTH_ERROR_EC (DDSC_LOG_BASE + 215)

  A RTPS message which was longer than the indicated message length in the RTPS header was received. This causes the rest of the RTPS message to be invalidated and the remaining submessages to be ignored.

5.51.1 Detailed Description

Real-Time Publish-Subscribe. ModuleID = 6.

5.51.2 Macro Definition Documentation

5.51.2.1 #define RTPS_LOG_INITIALIZE_INTERFACE_EC (RTPS_LOG_BASE + 1)

Failed to initialize an RTPS interface.

5.51.2.2 #define RTPS_LOG_ALLOCATE_EC (RTPS_LOG_BASE + 2)

Failed to allocate heap memory for internal resources.

5.51.2.3 #define RTPS_LOG_CREATE_ROUTE_TABLE_EC (RTPS_LOG_BASE + 3)

Failed to create a database table for storing RTPS route info.

5.51.2.4 #define RTPS_LOG_DB_CREATE_ROUTE_PEER_INDEX_EC (RTPS_LOG_BASE + 4)

Failed to create database index for route-peer info.

5.51.2.5 #define RTPS_LOG_DB_CREATE_ROUTE_XPORT_INDEX_EC (RTPS_LOG_BASE + 5)

Failed to create database index for route-transport info.

5.51.2.6 #define RTPS_LOG_DB_CREATE_BIND_PEER_INDEX_EC (RTPS_LOG_BASE + 6)

Failed to create database index for bind-peer info.

5.51.2.7 #define RTPS_LOG_INDEX_ADD_ENTRY_EC (RTPS_LOG_BASE + 8)

Failed to add an entry to a database index.

5.51.2.8 #define RTPS_LOG_DB_SELECT_ALL_EC (RTPS_LOG_BASE + 9)

Failed to select all entries of a database table.

Failure reason given by database return code (dbrc)

5.51.2.9 #define RTPS_LOG_DB_SELECT_MATCH_EC (RTPS_LOG_BASE + 10)

Matching entry not found in a database table.

Failure reason given by database return code (dbrc)

5.51.2.10 #define RTPS_LOG_DB_SELECT_RANGE_EC (RTPS_LOG_BASE + 11)

No matching entries found in a database table.

5.51.2.11 #define RTPS_LOG_DB_CREATE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 12)

Failed to create database entry for route.

May have exceeded DomainParticipantQos.resource_limits.matching_writer_reader_pair_allocation, DataWriterQos.writer_resource_limits.max_remote_readers, or DataReaderQos.reader_resource_limits.max_remote_writers

5.51.2.12 #define RTPS_LOG_DB_INSERT_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 13)

Failed to insert entry into route table.

Failure reason given by database return code (dbrc)

5.51.2.13 #define RTPS_LOG_DB_DELETE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 15)

Failed to delete entry from route table.

Failure reason given by database return code (dbrc)

5.51.2.14 #define RTPS_LOG_DB_CREATE_BIND_ENTRY_EC (RTPS_LOG_BASE + 16)

Failed to create database entry for bind.

If DomainParticipant, may have exceeded DomainParticipantQos.resource_limits.matching_reader_writer_pair_allocation. Failure reason given by database return code (dbrc).

5.51.2.15 #define RTPS_LOG_DB_INSERT_BIND_ENTRY_EC (RTPS_LOG_BASE + 17)

Failed to insert an entry into the bind table.

Failure reason given by database return code (dbrc)

5.51.2.16 #define RTPS_LOG_DB_REMOVE_BIND_ENTRY_EC (RTPS_LOG_BASE + 18)

Failed to remove an entry from the bind table.

Failure reason given by database return code (dbrc)

5.51.2.17 #define RTPS_LOG_DB_REMOVE_EXTERNAL_BIND_ENTRY_EC (RTPS_LOG_BASE + 19)

Failed to remove an entry from the external bind table.

Failure reason given by database return code (dbrc)

5.51.2.18 #define RTPS_LOG_DB_DELETE_EXTERNAL_BIND_ENTRY_EC (RTPS_LOG_BASE + 20)

Failed to delete entry from external bind table.

Failure reason given by database return code (dbrc)

5.51.2.19 #define RTPS_LOG_SEND_EC (RTPS_LOG_BASE + 21)

Failed to send a packet due to a lower module failing to send the packet.

5.51.2.20 #define RTPS_LOG_RECEIVE_EC (RTPS_LOG_BASE + 22)

Failed to receive a packet due to a higher module failing to receive the packet.

5.51.2.21 #define RTPS_LOG_ROUTE_PACKET_EC (RTPS_LOG_BASE + 23)

Failed to send a packet, when routing to a lower module or peer.

5.51.2.22 #define RTPS_LOG_FORWARD_UPSTREAM_EC (RTPS_LOG_BASE + 24)

Failed to receive a packet, when forwarding to a higher upstream module.

5.51.2.23 #define RTPS_LOG_BAD_PARAMETER_EC (RTPS_LOG_BASE + 25)

Bad parameter to an RTPS interface function.

5.51.2.24 #define RTPS_LOG_NOT_ENABLED_EC (RTPS_LOG_BASE + 26)

Failed because interface is not enabled.

5.51.2.25 #define RTPS_LOG_READER_UNSUPPORTED_EC (RTPS_LOG_BASE + 27)

RTPS reader does not support send.

5.51.2.26 #define RTPS_LOG_INTERFACE_MISMATCH_EC (RTPS_LOG_BASE + 28)

Upstream interface does not match source or destination of packet being sent or received.

5.51.2.27 #define RTPS_LOG_FULL_SEND_WINDOW_EC (RTPS_LOG_BASE + 29)

Failed to send due to reaching maximum send window size.

5.51.2.28 #define RTPS_LOG_NETIO_PACKET_SET_TAIL_EC (RTPS_LOG_BASE + 31)

Failed to set tail of packet to send.

5.51.2.29 #define RTPS_LOG_FUNC_UNSUPPORTED_EC (RTPS_LOG_BASE + 32)

RTPS interface does not support this function.

5.51.2.30 #define RTPS_LOG_EXCEEDED_LIMIT_TRANSPORTS_EC (RTPS_LOG_BASE + 33)

Out of transport entries.

Exceeded either DataWriterQos.writer_resource_limits.max_remote_readers or DataReaderQos.reader_resource_limits.max_remote_writers

5.51.2.31 #define RTPS_LOG_EXCEEDED_LIMIT_PEERS_EC (RTPS_LOG_BASE + 34)

Out of peer entries.

Exceeded either DataWriterQos.writer_resource_limits.max_remote_readers or DataReaderQos.reader_resource-_limits.max_remote_writers

5.51.2.32 #define RTPS_LOG_ASSERT_PEER_EC (RTPS_LOG_BASE + 35)

Failed to assert a remote writer or reader.

5.51.2.33 #define RTPS_LOG_ASSERT_TRANSPORT_EC (RTPS_LOG_BASE + 36)

Failed to assert a downstream transport.

5.51.2.34 #define RTPS_LOG_FIND_EXTERNAL_INTERFACE_EC (RTPS_LOG_BASE + 37)

Failed to lookup an external interface.

5.51.2.35 #define RTPS_LOG_NONEXISTENT_ROUTE_EC (RTPS_LOG_BASE + 38)

Failed to delete a nonexistent route.

5.51.2.36 #define RTPS_LOG_NONEXISTENT_BIND_EC (RTPS_LOG_BASE + 39)

Failed to remove a nonexistent bind entry.

5.51.2.37 #define RTPS_LOG_NONEXISTENT_EXTERNAL_BIND_EC (RTPS_LOG_BASE + 40)

Failed to remove a nonexistent external bind entry.

5.51.2.38 #define RTPS_LOG_STALE_ACK_EPOCH__EC (RTPS_LOG_BASE + 41)

Dropped an ACKNACK submessage with an old epoch.

5.51.2.39 #define RTPS_LOG_STALE_HB_EPOCH_EC (RTPS_LOG_BASE + 42)

Dropped a HEARTBEAT submessage with an old epoch.

5.51.2.40 #define RTPS_LOG_ACK_EC (RTPS_LOG_BASE + 43)

Failed an acknack() upstream.

5.51.2.41 #define RTPS_LOG_REQUEST_EC (RTPS_LOG_BASE + 45)

Failed a request() upstream.

5.51.2.42 #define RTPS_LOG_RETURN_LOAN_EC (RTPS_LOG_BASE + 46)

Failed a return_loan() upstream.

5.51.2.43 #define RTPS_LOG_NETIO_PACKET_SET_HEAD_EC (RTPS_LOG_BASE + 47)

Failed to set the head of a packet.

5.51.2.44 #define RTPS_LOG_SEND_HEARTBEAT_EC (RTPS_LOG_BASE + 48)

Failed to send a HEARTBEAT submessage.

5.51.2.45 #define RTPS_LOG_SHIFT_BITMAP_EC (RTPS_LOG_BASE + 49)

Failed to shift a bitmap.

5.51.2.46 #define RTPS_LOG_FULLY_ACKED_READER_EC (RTPS_LOG_BASE + 50)

A Reader is fully acknowledged and does not need to respond to a final HEARTBEAT.

5.51.2.47 #define RTPS_LOG_DATA_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 51)

Dropping a DATA submessage whose sequence number is outside of a Reader's receive window.

5.51.2.48 #define RTPS_LOG_DATA_ALREADY_RECEIVED_EC (RTPS_LOG_BASE + 52)

Dropping a DATA submessage whose sequence number was previously received.

5.51.2.49 #define RTPS_LOG_INTERFACE_NOT_ENABLED_EC (RTPS_LOG_BASE + 53)

Failed to receive a message because interface is not enabled.

5.51.2.50 #define RTPS_LOG_INVALID_PACKET_EC (RTPS_LOG_BASE + 54)

Dropped a message with an invalid packet header.

5.51.2.51 #define RTPS_LOG_UNSUPPORTED_INTERFACE_EC (RTPS_LOG_BASE + 55)

Received a message on an unsupported interface.

5.51.2.52 #define RTPS_LOG_UNKNOWN_SUBMESSAGE_EC (RTPS_LOG_BASE + 56)

Received a submessage with an unknown ID.

5.51.2.53 #define RTPS_LOG_PROCESS_ACKNACK_EC (RTPS_LOG_BASE + 57)

Failed to process received ACKNACK submessage.

5.51.2.54 #define RTPS_LOG_PROCESS_DATA_EC (RTPS_LOG_BASE + 58)

Failed to process received DATA submessage.

5.51.2.55 #define RTPS_LOG_PROCESS_GAP_EC (RTPS_LOG_BASE + 59)

Failed to process received GAP submessage.

5.51.2.56 #define RTPS_LOG_PROCESS_HEARTBEAT_EC (RTPS_LOG_BASE + 60)

Failed to process received HEARTBEAT submessage.

5.51.2.57 #define RTPS_LOG_CREATE_HB_EVENT_EC (RTPS_LOG_BASE + 61)

Failed to create periodic HEARTBEAT event.

5.51.2.58 #define RTPS_LOG_CREATE_EVENT_TIMER_EC (RTPS_LOG_BASE + 62)

Failed to create periodic HEARTBEAT event timer.

5.51.2.59 #define RTPS_LOG_SEQ_NUM_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 63)

Failed to set bitmap bit due to out-of-range sequence number.

5.51.2.60 #define RTPS_LOG_FIRST_SN_GREATER_LAST_SN_EC (RTPS_LOG_BASE + 65)

Invalid range of sequence numbers to fill in bitmap.

5.51.2.61 #define RTPS_LOG_BITCOUNT_OUT_OF_BOUNDS_EC (RTPS_LOG_BASE + 66)

Deserialized an invalid out-of-bounds bitcount for a bitmap.

5.51.2.62 #define RTPS_LOG_SHIFT_SN_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 67)

Failed to shift bitmap due to invalid starting sequence number.

5.51.2.63 #define RTPS_LOG_SERIALIZE_HOST_ID_EC (RTPS_LOG_BASE + 68)

Failed to serialize host ID of GUID.

5.51.2.64 #define RTPS_LOG_SERIALIZE_APP_ID_EC (RTPS_LOG_BASE + 69)

Failed to serialize app ID of GUID.

5.51.2.65 #define RTPS_LOG_SERIALIZE_INSTANCE_ID_EC (RTPS_LOG_BASE + 70)

Failed to serialized instance ID of GUID.

5.51.2.66 #define RTPS_LOG_SERIALIZE_OBJECT_ID_EC (RTPS_LOG_BASE + 71)

Failed to serialize object ID of GUID.

5.51.2.67 #define RTPS_LOG_DESERIALIZE_HOST_ID_EC (RTPS_LOG_BASE + 72)

Failed to deserialize host ID of GUID.

5.51.2.68 #define RTPS_LOG_DESERIALIZE_APP_ID_EC (RTPS_LOG_BASE + 73)

Faild to deserialize app ID of GUID.

5.51.2.69 #define RTPS_LOG_DESERIALIZE_INSTANCE_ID_EC (RTPS_LOG_BASE + 74)

Failed to deserialize instance ID of GUID.

5.51.2.70 #define RTPS_LOG_DESERIALIZE_OBJECT_ID_EC (RTPS_LOG_BASE + 75)

Failed to deserialize object ID of GUID.

5.51.2.71 #define RTPS_LOG_DB_CREATE_EXT_BIND_ENTRY_EC (RTPS_LOG_BASE + 76)

Failed to create database entry for external bind table.

If DomainParticipant, may have exceeded DomainParticipantQos.resource_limits.matching_reader_writer_pair_-allocation.

5.51.2.72 #define RTPS_LOG_BITMAP_SET_BIT_EC (RTPS_LOG_BASE + 77)

Failed to set bit in bitmap.

5.51.2.73 #define RTPS_LOG_UNSUPPORTED_INFO_REPLY_EC (RTPS_LOG_BASE + 78)

Received and dropped currently unsupported INFO_REPLY submessage.

5.51.2.74 #define RTPS_LOG_UNSUPPORTED_INFO_REPLY_IP4_EC (RTPS_LOG_BASE + 79)

Received and dropped currently unsupported INFO_REPLY_IP4 submessage.

5.51.2.75 #define RTPS_LOG_UNSUPPORTED_INFO_SRC_EC (RTPS_LOG_BASE + 80)

Received and dropped currently unsupported INFO_SRC submessage.

5.51.2.76 #define RTPS_LOG_DELETE_INDEX_EC (RTPS_LOG_BASE + 81)

Failed to delete index of a database table.

5.51.2.77 #define RTPS_LOG_DELETE_TABLE_EC (RTPS_LOG_BASE + 82)

Failed to delete a database table.

5.51.2.78 #define RTPS_LOG_DB_LOCK_EC (RTPS_LOG_BASE + 83)

Failed to take database lock.

5.51.2.79 #define RTPS_LOG_DB_UNLOCK_EC (RTPS_LOG_BASE + 84)

Failed to give database lock.

5.51.2.80 #define RTPS_LOG_CREATE_BIND_TABLE_EC (RTPS_LOG_BASE + 85)

Failed to create a database table for storing RTPS bind info.

5.51.2.81 #define RTPS_LOG_STATUS_CHANGE_EC (RTPS_LOG_BASE + 86)

Failed to update status for reliable reader activity changed.

5.51.2.82 #define RTPS_LOG_SET_GAP_EC (RTPS_LOG_BASE + 87)

Failed to update status for reliable reader activity changed.

5.51.2.83 #define RTPS_LOG_WINDOW_POOL_ALLOC_EC (RTPS_LOG_BASE + 88)

Out of memory to allocate send window pool.

5.51.2.84 #define RTPS_LOG_GET_WINDOW_BUFFER_EC (RTPS_LOG_BASE + 89)

Failed to get buffer from send window pool.

5.51.2.85 #define RTPS_LOG_INSERT_WINDOW_FULL_EC (RTPS_LOG_BASE + 90)

Cannot insert sample into full window.

5.51.2.86 #define RTPS_LOG_WINDOW_INSERT_EC (RTPS_LOG_BASE + 91)

Failed to insert sample into send window.

5.51.2.87 #define RTPS_LOG_TIMER_DELETE_TIMEOUT_EC (RTPS_LOG_BASE + 92)

Failed to delete a timeout event.

5.51.2.88 #define RTPS_LOG_BUFFERPOOL_DELETE_EC (RTPS_LOG_BASE + 93)

Failed to delete a buffer pool.

5.51.2.89 #define RTPS_LOG_DIRECT_SEND_EC (RTPS_LOG_BASE + 94)

Failed to send a message to a specific peer.

5.51.2.90 #define RTPS_LOG_PACKET_INITIALIZE_EC (RTPS_LOG_BASE + 95)

RTPS writer failed to initialize a packet.

5.51.2.91 #define RTPS_LOG_WRITER_REQUEST_SENT_HISTORY_EC (RTPS_LOG_BASE + 96)

RTPS reliable writer failed to request and resend history.

5.51.2.92 #define RTPS_LOG_BITMAP_SHIFT_EC (RTPS_LOG_BASE + 97)

Failed to shift bitmap to new lead sequence number.

5.51.2.93 #define RTPS_LOG_WINDOW_ADVANCE_EC (RTPS_LOG_BASE + 98)

Send window of a remote reader failed to advance.

5.51.2.94 #define RTPS_LOG_WINDOW_ADVANCE_UNACKED_EC (RTPS_LOG_BASE + 99)

Failed when advancing window ahead of unacknowledged sequence number.

5.51.2.95 #define RTPS_LOG_LOOKUP_PEER_EC (RTPS_LOG_BASE + 100)

Failed to find peer that should exist.

5.51.2.96 #define RTPS_LOG_READER_SEND_ACKNACK_EC (RTPS_LOG_BASE + 101)

RTPS reader failed to send an ACKNACK message.

5.51.2.97 #define RTPS_LOG_FINALIZE_INTERFACE_EC (RTPS_LOG_BASE + 102)

Failed to finalize an RTPS interface.

5.51.2.98 #define RTPS_LOG_INDEXER_REMOVE_ENTRY_EC (RTPS_LOG_BASE + 103)

Failed to remove index entry for external interface upon deleting RTPS interface.

5.51.2.99 #define RTPS_LOG_LOOKUP_EXT_BIND_ENTRY_EC (RTPS_LOG_BASE + 104)

Failed when looking up external bind entry from database.

5.51.2.100 #define RTPS_LOG_SEQ_INIT_EC (RTPS_LOG_BASE + 105)

Failed to initialize a local sequence.

5.51.2.101 #define RTPS_LOG_SEQ_MAX_EC (RTPS_LOG_BASE + 106)

Failed to set maximum of a local sequence.

5.51.2.102 #define RTPS_LOG_SEQ_LEN_EC (RTPS_LOG_BASE + 107)

Failed to set length of a local sequence.

5.51.2.103 #define RTPS_LOG_INTF_MODE_UNDEF_EC (RTPS_LOG_BASE + 108)

Initializing an RTPS interface with undefined mode.

5.51.2.104 #define RTPS_LOG_DELETE_BUFFER_POOL_EC (RTPS_LOG_BASE + 109)

Failed to delete buffer pool.

May still have buffers in use, thus cannot delete pool.

5.51.2.105 #define RTPS_LOG_CREATE_BUFFER_POOL_EC (RTPS_LOG_BASE + 110)

Failed to create buffer pool.

May have insufficient memory to allocate pool.

5.51.2.106 #define RTPS_LOG_CREATE_PEERS_INDEX_EC (RTPS_LOG_BASE + 111)

Failed to create index of peers.

System may have insufficient memory to allocate new index.

5.51.2.107 #define RTPS_LOG_DELETE_INDEXER_EC (RTPS_LOG_BASE + 112)

Failed to delete indexer.

5.51.2.108 #define RTPS_LOG_GET_PEER_BUFFER_EC (RTPS_LOG_BASE + 113)

Failed to get buffer from peer buffer pool.

5.51.2.109 #define RTPS_LOG_DB_CREATE_DIRECT_INDEX_EC (RTPS_LOG_BASE + 114)

Failed to create database index for direct sends.

5.51.2.110 #define RTPS_LOG_DB_CREATE_GROUP_INDEX_EC (RTPS_LOG_BASE + 115)

Failed to create database index for group sends.

5.51.2.111 #define RTPS_LOG_NETIO_PACKET_INSUFFICIENT_BUFFER_EC (RTPS_LOG_BASE + 116)

Insufficient buffer in packet.

Receive buffer does not have enough space for a valid RTPS message

5.51.2.112 #define RTPS_LOG_PROCESS_DATA_BATCH_EC (RTPS_LOG_BASE + 117)

Failed to process received DATA BATCH submessage.

This indicates an invalid DATA_BATCH submessage

5.51.2.113 #define RTPS_LOG_PROCESS_HEARTBEAT_BATCH_EC (RTPS_LOG_BASE + 118)

Failed to process received HEARTBEAT_BATCH submessage.

This indicates an invalid HEARTBEAT_BATCH submessage

5.51.2.114 #define RTPS_LOG_NEGATIVE_RESERVATION_COUNT_EC (RTPS_LOG_BASE + 119)

The reservation count it zero.

The reservation count should never be less than 0, this is an internal error.

5.51.2.115 #define RTPS_LOG_DB_CREATE_SELECTED_GROUP_INDEX_EC (RTPS_LOG_BASE + 148)

Failed to create database index for the selected route group.

5.51.2.116 #define RTPS_LOG_CHECKSUM_MISSING_CHECKSUM_LENGTH_EC (RTPS_LOG_BASE + 149)

A header extension was received, but the checksum length is not present.

5.51.2.117 #define RTPS_LOG_CHECKSUM_UNSUPPORTED_EC (RTPS_LOG_BASE + 150)

Unsupported checksum bits received, message dropped.

5.51.2.118 #define RTPS_LOG_CHECKSUM_LENGTH_DESERIALIZE_ERROR_EC (RTPS_LOG_BASE + 151)

Checksum length could not be deserialized.

5.51.2.119 #define RTPS_LOG_CHECKSUM_INCONSISTENT_LENGTH_EC (RTPS_LOG_BASE + 152)

The payload length is less than the deserialized checksum length.

5.51.2.120 #define RTPS_LOG_CHECKSUM_CHECKSUM_DESERIALIZE_ERROR_EC (RTPS_LOG_BASE + 153)

Checksum length could not be deserialized.

5.51.2.121 #define RTPS_LOG_CHECKSUM_CHECKSUM_ERROR_EC (RTPS_LOG_BASE + 154)

The checksum is invalid.

5.51.2.122 #define RTPS_LOG_CHECKSUM_INVALID_CRC32_FLAGS_EC (RTPS_LOG_BASE + 155)

The CRC32 submsg flags are invalid.

5.51.2.123 #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM128_EC (RTPS_LOG_BASE + 200)

The property tries to override the BUILTIN128 checksum without setting allow_builtin_override to TRUE.

5.51.2.124 #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM32_EC (RTPS_LOG_BASE + 201)

The property tries to override the BUILTIN32 checksum without setting allow_builtin_override to TRUE.

5.51.2.125 #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM64_EC (RTPS_LOG_BASE + 202)

The property tries to override the BUILTIN64 checksum without setting allow_builtin_override to TRUE.

5.51.2.126 #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN128_EC (RTPS_LOG_BASE + 203)

The property override of the BUILTIN128 checksum is invalid. It is not legal to reuse any of the built-in functions.

5.51.2.127 #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN32_EC (RTPS_LOG_BASE + 204)

The property override of the BUILTIN32 checksum is invalid. It is not legal to reuse any of the built-in functions functions.

5.51.2.128 #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN64_EC (RTPS_LOG_BASE + 205)

The property override of the BUILTIN64 checksum is invalid. It is not legal to reuse any of the built-in functions functions.

5.51.2.129 #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM256_EC (RTPS_LOG_BASE + 206)

The property specifies an invalid custom256 checksum function.

5.51.2.130 #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM32_EC (RTPS_LOG_BASE + 207)

The property specifies an invalid custom32 checksum function.

5.51.2.131 #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM64_EC (RTPS_LOG_BASE + 208)

The property specifies an invalid custom64 checksum function.

5.51.2.132 #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM128_EC (RTPS_LOG_BASE + 209)

The property specifies an invalid custom128 checksum function.

5.51.2.133 #define RTPS_LOG_CHECKSUM_CHECKSUM_CALCULATION_FAILED_EC (RTPS_LOG_BASE + 210)

RTPS_Interface_calculate_checksum() failed to calculate the checksum.

5.51.2.134 #define RTPS_LOG_UNSUPPORTED_MANDTORY_HDR_EXT_PID_EC (RTPS_LOG_BASE + 211)

The HeaderExtension contained a mandatory PID that was not understood.

5.51.2.135 #define RTPS_LOG_INVALID_HDR_EXT_PID_LIST_EC (RTPS_LOG_BASE + 212)

An invalid HeaderExtension PID was detected.

5.51.2.136 #define RTPS_LOG_INVALID_HDR_EXT_PID_ERROR_EC (RTPS_LOG_BASE + 213)

Failed to decode the HeaderExtension PID list.

5.51.2.137 #define RTPS_LOG_INCONSISTENT_HDR_EXT_LENGTH_ERROR_EC (RTPS_LOG_BASE + 214)

The HeaderExtension length is isconsistent with the decoded encoded HeaderExtension length.

5.51.2.138 #define RTPS_LOG_MESSAGE_EXCEED_LENGTH_ERROR_EC (DDSC_LOG_BASE + 215)

A RTPS message which was longer than the indicated message length in the RTPS header was received. This causes the rest of the RTPS message to be invalidated and the remaining submessages to be ignored.

5.52 DDS-Specific Primitive Types

Basic DDS value types for use in user data types.

Macros

- #define DDS_BOOLEAN_TRUE
  Defines TRUE value of IDL boolean data type.
- #define DDS_BOOLEAN_FALSE
  Defines FALSE value of IDL boolean data type.

Typedefs

- typedef CDR_Char DDS_Char
  Defines IDL char data type.
- typedef CDR_Wchar DDS_Wchar
  Defines IDL wchar data type.
- typedef CDR_Octet DDS_Octet
  Defines IDL octet data type.
- typedef CDR_Short DDS_Short
  Defines IDL short data type.
- typedef CDR_UnsignedShort DDS_UnsignedShort
  Defines IDL unsigned short data type.
- typedef CDR_Long DDS_Long
  Defines IDL long data type.
- typedef CDR_UnsignedLong DDS_UnsignedLong
  Defines IDL unsigned long data type.
- typedef CDR_LongLong DDS_LongLong
  Defines IDL long long data type.
- typedef CDR_UnsignedLongLong DDS_UnsignedLongLong
  Defines IDL unsigned long long data type.
- typedef CDR_Float DDS_Float
  Defines IDL float data type.
- typedef CDR_Double DDS_Double
  Defines IDL double data type.
- typedef CDR_LongDouble DDS_LongDouble
  Defines IDL long double data type.
- typedef CDR_Boolean DDS_Boolean
  Defines IDL boolean data type.
- typedef CDR_Enum DDS_Enum
  Defines IDL enum data type.
- typedef CDR_String DDS_String
  IDL `string` type representation.
- typedef CDR_Wstring DDS_Wstring
  Defines IDL wstring data type.

5.52.1 Detailed Description

Basic DDS value types for use in user data types. As part of the finalization of the DDS standard, a number of DDS-specific primitive types will be introduced. By using these types, you will ensure that your data is serialized consistently across platforms even if the built-in types have different sizes on those platforms.

In this version of RTI Connext Micro, the DDS primitive types are defined using the OMG's Common Data Representation (CDR) standard.

5.52.2 Macro Definition Documentation

5.52.2.1 #define DDS_BOOLEAN_TRUE

Defines TRUE value of IDL boolean data type.

5.52.2.2 #define DDS_BOOLEAN_FALSE

Defines FALSE value of IDL boolean data type.

5.52.3 Typedef Documentation

5.52.3.1 typedef CDR_Char DDS_Char

Defines IDL char data type.

An 8-bit quantity that encodes a single byte charater from any byte-oriented code set.

5.52.3.2 typedef CDR_Wchar DDS_Wchar

Defines IDL wchar data type.

An 16-bit quantity that encodes a wide character from any character set.

5.52.3.3 typedef CDR_Octet DDS_Octet

Defines IDL octet data type.

An 8-bit quantity that is guaranteed not to undergo any conversion when transmitted by the middleware.

5.52.3.4 typedef CDR_Short DDS_Short

Defines IDL short data type.

A 16-bit signed short integer value.

5.52.3.5 typedef CDR_UnsignedShort DDS_UnsignedShort

Defines IDL unsigned short data type.

A 16-bit unsigned short integer value.

5.52.3.6 typedef CDR_Long DDS_Long

Defines IDL long data type.

A 32-bit signed long integer value.

5.52.3.7 typedef CDR_UnsignedLong DDS_UnsignedLong

Defines IDL unsigned long data type.

A 32-bit unsigned long integer value.

5.52.3.8 typedef CDR_LongLong DDS_LongLong

Defines IDL long long data type.

A 64-bit signed long long integer value.

Since some architectures do not support long long (8 bytes long), RTI has defined character arrays that match the expected size of these types.

5.52.3.9 typedef CDR_UnsignedLongLong DDS_UnsignedLongLong

Defines IDL unsigned long long data type.

An 64-bit unsigned long long integer value.

Since some architectures do not support (unsigned) long long (8 byte long), RTI has defined character arrays that match the expected size of these types.

5.52.3.10 typedef CDR_Float DDS_Float

Defines IDL float data type.

A 32-bit floating point value.

5.52.3.11 typedef CDR_Double DDS_Double

Defines IDL double data type.

A 64-bit floating point value.

5.52.3.12 typedef CDR_LongDouble DDS_LongDouble

Defines IDL long double data type.

A 16-byte floating point value.

5.52.3.13 typedef CDR_Boolean DDS_Boolean

Defines IDL boolean data type.

An 8-bit boolean value that is used to denote a data item that can only take one of the values DDS_BOOLEAN_TRUE (1) or DDS_BOOLEAN_FALSE (0).

5.52.3.14 typedef CDR_Enum DDS_Enum

Defines IDL enum data type.

Encoded as unsigned long value. The first enum identifier has the numeric value zero (0). Successive enum identifiers take ascending numeric values, in order of declaration from left to right.

5.52.3.15 typedef CDR_String DDS_String

IDL `string` type representation.

Has a range of ISO Latin-1 (except ASCII NULL) and a variable length.

5.52.3.16 typedef CDR_Wstring DDS_Wstring

Defines IDL wstring data type.

Each character composing the string is a 32-bit quantity that encodes a wide character from any character set.

5.53 Participant Built-in Topic

Builtin topic for accessing information about the DomainParticipants discovered by RTI Connext Micro.

Data Structures

- struct DDS_ParticipantBuiltinTopicData

  <<*cert*>> *Object representing a remote DomainParticipant.*

Macros

- #define DDS_PARTICIPANT_TOPIC_NAME DDS_PARTICIPANT_BUILTIN_TOPIC_NAME

  <<*cert*>> *Participant topic name*

Functions

- DDS_Boolean DDS_ParticipantBuiltinTopicData_initialize (struct DDS_ParticipantBuiltinTopicData *self)

  <<*cert*>> *Initialize a DDS_ParticipantBuiltinTopicData object.*
- DDS_Boolean DDS_ParticipantBuiltinTopicData_copy (struct DDS_ParticipantBuiltinTopicData *out, const struct DDS_ParticipantBuiltinTopicData *in)

  <<*cert*>> *Copy a DDS_ParticipantBuiltinTopicData object.*
- DDS_Boolean DDS_ParticipantBuiltinTopicData_is_equal (const struct DDS_ParticipantBuiltinTopicData *left, const struct DDS_ParticipantBuiltinTopicData *right)

  <<*cert*>> *Compare two DDS_ParticipantBuiltinTopicData objects for equality.*

5.53.1 Detailed Description

Builtin topic for accessing information about the DomainParticipants discovered by RTI Connext Micro.

5.53.2 Macro Definition Documentation

5.53.2.1 #define DDS_PARTICIPANT_TOPIC_NAME DDS_PARTICIPANT_BUILTIN_TOPIC_NAME

<<*cert*>> Participant topic name

Topic name of participant builtin topic data DataReader.

See also

DDS_ParticipantBuiltinTopicData

5.53.3 Function Documentation

5.53.3.1 DDS_Boolean DDS_ParticipantBuiltinTopicData_initialize (struct DDS_ParticipantBuiltinTopicData * *self*)

<<*cert*>> Initialize a DDS_ParticipantBuiltinTopicData object.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

DDS_BOOLEAN_TRUE on success,DDS_BOOLEAN_FALSE on failure.

5.53.3.2 DDS_Boolean DDS_ParticipantBuiltinTopicData_copy (struct DDS_ParticipantBuiltinTopicData * *out*, const struct DDS_ParticipantBuiltinTopicData * *in*)

<<*cert*>> Copy a DDS_ParticipantBuiltinTopicData object.

Parameters

out	*out*	The DDS_ParticipantBuiltinTopicData object being copied to. The object must have been initialized with DDS_ParticipantBuiltinTopicData_initialize.
in	*in*	The DDS_ParticipantBuiltinTopicData object being copied from.

Returns

DDS_BOOLEAN_TRUE on success,DDS_BOOLEAN_FALSE on failure.

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.53.3.3 DDS_Boolean DDS_ParticipantBuiltinTopicData_is_equal (const struct DDS_ParticipantBuiltinTopicData * *left*, const struct DDS_ParticipantBuiltinTopicData * *right*)

<<*cert*>> Compare two DDS_ParticipantBuiltinTopicData objects for equality.

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

5.54 Publication Built-in Topic

Builtin topic for configuring information about the Publications to be discovered by RTI Connext Micro.

Data Structures

- struct DDS_PublicationBuiltinTopicData

  Object describing a remote Publication.

Macros

- #define DDS_PUBLICATION_TOPIC_NAME DDS_PUBLICATION_BUILTIN_TOPIC_NAME

  Publication topic name.

Functions

- DDS_Boolean DDS_PublicationBuiltinTopicData_initialize (struct DDS_PublicationBuiltinTopicData *self)

  <<cert>> Initialize a DDS_PublicationBuiltinTopicData object.
- DDS_Boolean DDS_PublicationBuiltinTopicData_copy (struct DDS_PublicationBuiltinTopicData *out, const struct DDS_PublicationBuiltinTopicData *in)

  <<cert>> Copy a DDS_PublicationBuiltinTopicData object.
- DDS_Boolean DDS_PublicationBuiltinTopicData_is_equal (const struct DDS_PublicationBuiltinTopicData *left, const struct DDS_PublicationBuiltinTopicData *right)

  <<cert>> Compare two DDS_PublicationBuiltinTopicData objects for equality

5.54.1 Detailed Description

Builtin topic for configuring information about the Publications to be discovered by RTI Connext Micro.

5.54.2 Macro Definition Documentation

5.54.2.1 #define DDS_PUBLICATION_TOPIC_NAME DDS_PUBLICATION_BUILTIN_TOPIC_NAME

Publication topic name.

Topic name of publication builtin topic data DataReader.

See also

DDS_PublicationBuiltinTopicData

5.54.3 Function Documentation

5.54.3.1 DDS_Boolean DDS_PublicationBuiltinTopicData_initialize (struct DDS_PublicationBuiltinTopicData * *self*)

<<cert>> Initialize a DDS_PublicationBuiltinTopicData object.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

DDS_BOOLEAN_TRUE on success,DDS_BOOLEAN_FALSE on failure.

5.54.3.2 DDS_Boolean DDS_PublicationBuiltinTopicData_copy (struct DDS_PublicationBuiltinTopicData * *out,* const struct DDS_PublicationBuiltinTopicData * *in*)

<<*cert*>> Copy a DDS_PublicationBuiltinTopicData object.

Parameters

out	*out*	The DDS_PublicationBuiltinTopicData object being copied to. The object must have been initialized with DDS_PublicationBuiltinTopicData_initialize.
in	*in*	The DDS_PublicationBuiltinTopicData object being copied from.

Returns

DDS_BOOLEAN_TRUE on success,DDS_BOOLEAN_FALSE on failure.

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.54.3.3 DDS_Boolean DDS_PublicationBuiltinTopicData_is_equal (const struct DDS_PublicationBuiltinTopicData * *left,* const struct DDS_PublicationBuiltinTopicData * *right*)

<<*cert*>> Compare two DDS_PublicationBuiltinTopicData objects for equality

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

5.55 Subscription Built-in Topic

Builtin topic for configuring information about the Subscriptions to be discovered by RTI Connext Micro.

Data Structures

- struct DDS_SubscriptionBuiltinTopicData

  Entry created when a DDS_DataReader is discovered in association with its Subscriber.

Macros

- #define DDS_SUBSCRIPTION_TOPIC_NAME DDS_SUBSCRIPTION_BUILTIN_TOPIC_NAME

  Subscription topic name.

Functions

- DDS_Boolean DDS_SubscriptionBuiltinTopicData_initialize (struct DDS_SubscriptionBuiltinTopicData *self)

  <<cert>> Initialize a DDS_SubscriptionBuiltinTopicData object.
- DDS_Boolean DDS_SubscriptionBuiltinTopicData_copy (struct DDS_SubscriptionBuiltinTopicData *out, const struct DDS_SubscriptionBuiltinTopicData *in)

  <<cert>> Copy a DDS_SubscriptionBuiltinTopicData object.
- DDS_Boolean DDS_SubscriptionBuiltinTopicData_is_equal (const struct DDS_SubscriptionBuiltinTopicData *left, const struct DDS_SubscriptionBuiltinTopicData *right)

  <<cert>> Compare two DDS_SubscriptionBuiltinTopicData objects for equality

5.55.1 Detailed Description

Builtin topic for configuring information about the Subscriptions to be discovered by RTI Connext Micro.

5.55.2 Macro Definition Documentation

5.55.2.1 #define DDS_SUBSCRIPTION_TOPIC_NAME DDS_SUBSCRIPTION_BUILTIN_TOPIC_NAME

Subscription topic name.

Topic name of subscription builtin topic data DataReader.

See also

DDS_SubscriptionBuiltinTopicData

5.55.3 Function Documentation

5.55.3.1 DDS_Boolean DDS_SubscriptionBuiltinTopicData_initialize (struct DDS_SubscriptionBuiltinTopicData * *self*)

<<cert>> Initialize a DDS_SubscriptionBuiltinTopicData object.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

DDS_BOOLEAN_TRUE on success,DDS_BOOLEAN_FALSE on failure.

5.55.3.2 DDS_Boolean DDS_SubscriptionBuiltinTopicData_copy (struct DDS_SubscriptionBuiltinTopicData * *out*, const struct DDS_SubscriptionBuiltinTopicData * *in*)

<<*cert*>> Copy a DDS_SubscriptionBuiltinTopicData object.

Parameters

out	*out*	The DDS_SubscriptionBuiltinTopicData object being copied to. The object must have been initialized with DDS_SubscriptionBuiltinTopicData_initialize.
in	*in*	The DDS_SubscriptionBuiltinTopicData object being copied from.

Returns

DDS_BOOLEAN_TRUE on success,DDS_BOOLEAN_FALSE on failure.

MT Safety:

UNSAFE. This operation is not thread safe. This operation does not protect the source or destination from being modified by another thread while the source is being copied to the destination.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.55.3.3 DDS_Boolean DDS_SubscriptionBuiltinTopicData_is_equal (const struct DDS_SubscriptionBuiltinTopicData * *left*, const struct DDS_SubscriptionBuiltinTopicData * *right*)

<<*cert*>> Compare two DDS_SubscriptionBuiltinTopicData objects for equality

Parameters

in	*left*	The left side of the comparison.
in	*right*	The right side of the comparison.

Returns

DDS_BOOLEAN_TRUE if left is equal to right, otherwise DDS_BOOLEAN_FALSE.

MT Safety:

UNSAFE. This operation does not protect against the left or right side from being modified by another thread while the comparison is made.

5.56 DISCOVERY

<<eXtension>> Specifies the attributes required to discover participants in the domain.

Data Structures

- struct DDS_DiscoveryQosPolicy

  <<cert>> *Specifies the attributes required to discover participants in the domain.*

Variables

- const char *const DDS_DISCOVERY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DiscoveryQosPolicy.

5.56.1 Detailed Description

<<eXtension>> Specifies the attributes required to discover participants in the domain.

5.56.2 Variable Documentation

5.56.2.1 const char* const DDS_DISCOVERY_QOS_POLICY_NAME

Stringified human-readable name for DDS_DiscoveryQosPolicy.

5.57 Time Support

Time and duration types and defines.

Data Structures

- struct DDS_Time_t
  Type for time representation.
- struct DDS_Duration_t
  Type for duration representation.

Macros

- #define DDS_TIME_ZERO
  The default instant in time: zero seconds and zero nanoseconds.

Variables

- const DDS_Long DDS_TIME_INVALID_SEC
  A sentinel indicating an invalid second of time.
- const DDS_UnsignedLong DDS_TIME_INVALID_NSEC
  A sentinel indicating an invalid nano-second of time.
- struct DDS_Time_t DDS_TIME_INVALID
  A sentinel indicating an invalid time.
- const DDS_Long DDS_DURATION_INFINITE_SEC
  An infinite second period of time.
- const DDS_UnsignedLong DDS_DURATION_INFINITE_NSEC
  An infinite nano-second period of time.
- struct DDS_Duration_t DDS_DURATION_INFINITE
  An infinite period of time.
- const DDS_Long DDS_DURATION_ZERO_SEC
  A zero-length second period of time.
- const DDS_UnsignedLong DDS_DURATION_ZERO_NSEC
  A zero-length nano-second period of time.
- struct DDS_Duration_t DDS_DURATION_ZERO
  A zero-length period of time.

5.57.1 Detailed Description

Time and duration types and defines.

5.57.2 Macro Definition Documentation

5.57.2.1 #define DDS_TIME_ZERO

The default instant in time: zero seconds and zero nanoseconds.

5.57.3 Variable Documentation

5.57.3.1 const DDS_Long DDS_TIME_INVALID_SEC

A sentinel indicating an invalid second of time.

5.57.3.2 const DDS_UnsignedLong DDS_TIME_INVALID_NSEC

A sentinel indicating an invalid nano-second of time.

5.57.3.3 struct DDS_Time_t DDS_TIME_INVALID

A sentinel indicating an invalid time.

5.57.3.4 const DDS_Long DDS_DURATION_INFINITE_SEC

An infinite second period of time.

5.57.3.5 const DDS_UnsignedLong DDS_DURATION_INFINITE_NSEC

An infinite nano-second period of time.

5.57.3.6 struct DDS_Duration_t DDS_DURATION_INFINITE

An infinite period of time.

5.57.3.7 const DDS_Long DDS_DURATION_ZERO_SEC

A zero-length second period of time.

5.57.3.8 const DDS_UnsignedLong DDS_DURATION_ZERO_NSEC

A zero-length nano-second period of time.

5.57.3.9 struct DDS_Duration_t DDS_DURATION_ZERO

A zero-length period of time.

5.58 GUID Support

GUID type and defines.

Data Structures

- struct DDS_GUID_t

  Type for GUID (Global Unique Identifier) representation.

Variables

- struct DDS_GUID_t DDS_GUID_UNKNOWN

  Unknown GUID.

5.58.1 Detailed Description

GUID type and defines.

5.58.2 Variable Documentation

5.58.2.1 struct DDS_GUID_t DDS_GUID_UNKNOWN

Unknown GUID.

5.59 Sequence Number Support

Sequence number type and defines.

Data Structures

- struct DDS_SequenceNumber_t

  Type for sequence number representation.

5.59.1 Detailed Description

Sequence number type and defines.

5.60 Return Codes

Types of return codes.

Enumerations

- enum DDS_ReturnCode_t {
  DDS_RETCODE_OK, DDS_RETCODE_ERROR, DDS_RETCODE_UNSUPPORTED, DDS_RETCODE_BAD_PARAMETER,
  DDS_RETCODE_PRECONDITION_NOT_MET, DDS_RETCODE_OUT_OF_RESOURCES, DDS_RETCODE_NOT_ENABLED, DDS_RETCODE_IMMUTABLE_POLICY,
  DDS_RETCODE_INCONSISTENT_POLICY, DDS_RETCODE_ALREADY_DELETED, DDS_RETCODE_TIMEOUT, DDS_RETCODE_NO_DATA,
  DDS_RETCODE_ILLEGAL_OPERATION }

  Type for return codes.

5.60.1 Detailed Description

Types of return codes.

5.60.2 Standard Return Codes

Any operation with return type DDS_ReturnCode_t may return DDS_RETCODE_OK DDS_RETCODE_ERROR or DDS_RETCODE_ILLEGAL_OPERATION. Any operation that takes one or more input parameters may additionally return DDS_RETCODE_BAD_PARAMETER. Any operation on an object created from any of the factories may additionally return DDS_RETCODE_ALREADY_DELETED. Any operation that is stated as optional may additionally return DDS_RETCODE_UNSUPPORTED.

Thus, the standard return codes are:

- DDS_RETCODE_ERROR
- DDS_RETCODE_ILLEGAL_OPERATION
- DDS_RETCODE_ALREADY_DELETED
- DDS_RETCODE_BAD_PARAMETER
- DDS_RETCODE_UNSUPPORTED

Operations that may return any of the additional return codes will state so explicitly.

5.60.3 Enumeration Type Documentation

5.60.3.1 enum DDS_ReturnCode_t

Type for return codes.

Errors are modeled as operation return codes of this type.

Enumerator:

DDS_RETCODE_OK Successful return.

DDS_RETCODE_ERROR Generic, unspecified error.

DDS_RETCODE_UNSUPPORTED Unsupported operation. Can only returned by operations that are unsupported.

DDS_RETCODE_BAD_PARAMETER Illegal parameter value. The value of the parameter that is passed in has illegal value. Things that falls into this category includes NULL parameters and parameter values that are out of range.

DDS_RETCODE_PRECONDITION_NOT_MET A pre-condition for the operation was not met. The system is not in the expected state when the function is called, or the parameter itself is not in the expected state when the function is called.

DDS_RETCODE_OUT_OF_RESOURCES RTI Connext Micro ran out of the resources needed to complete the operation.

DDS_RETCODE_NOT_ENABLED Operation invoked on a DDS_Entity that is not yet enabled.

DDS_RETCODE_IMMUTABLE_POLICY Application attempted to modify an immutable QoS policy.

DDS_RETCODE_INCONSISTENT_POLICY Application specified a set of QoS policies that are not consistent with each other.

DDS_RETCODE_ALREADY_DELETED The object target of this operation has already been deleted.

DDS_RETCODE_TIMEOUT The operation timed out.

DDS_RETCODE_NO_DATA Indicates a transient situation where the operation did not return any data but there is no inherent error.

DDS_RETCODE_ILLEGAL_OPERATION The operation was called under improper circumstances. An operation was invoked on an inappropriate object or at an inappropriate time. This return code is similar to DDS_RETCODE_PRECONDITION_NOT_MET, except that there is no precondition that could be changed to make the operation succeed.

5.61 Status Kinds

Kinds of communication status.

Macros

- #define DDS_STATUS_MASK_NONE

  No bits are set.
- #define DDS_STATUS_MASK_ALL

  All bits are set.

Typedefs

- typedef DDS_UnsignedLong DDS_StatusMask

  A bit-mask (list) of concrete status types, i.e. DDS_StatusKind[].

Enumerations

- enum DDS_StatusKind {
  DDS_INCONSISTENT_TOPIC_STATUS, DDS_OFFERED_DEADLINE_MISSED_STATUS, DDS_REQUESTED_DEADLINE_MISSED_STATUS, DDS_OFFERED_INCOMPATIBLE_QOS_STATUS,
  DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS, DDS_SAMPLE_LOST_STATUS, DDS_SAMPLE_REJECTED_STATUS, DDS_DATA_ON_READERS_STATUS,
  DDS_DATA_AVAILABLE_STATUS, DDS_LIVELINESS_LOST_STATUS, DDS_LIVELINESS_CHANGED_STATUS, DDS_PUBLICATION_MATCHED_STATUS,
  DDS_SUBSCRIPTION_MATCHED_STATUS, DDS_INSTANCE_REPLACED_STATUS, DDS_RELIABLE_READER_ACTIVITY_CHANGED_STATUS }

  Type for status kinds.

5.61.1 Detailed Description

Kinds of communication status.

Entity:

DDS_Entity

QoS:

QoS Policies

Listener:

DDS_Listener

Each concrete DDS_Entity is associated with a set of Status objects whose value represents the communication status of that entity. Each status value can be accessed with a corresponding function on the DDS_Entity.

RTI Connext Micro supports a subset of these statuses.

When these status values change, the proper DDS_Listener objects are invoked to asynchronously inform the application.

An application is notified of communication status by means of the DDS_Listener mechanism.

The statuses may be classified into:

- *read communication statuses*: i.e., those that are related to arrival of data, namely DDS_DATA_ON_READERS_STATUS and DDS_DATA_AVAILABLE_STATUS.

- *plain communication statuses*: i.e., all the others.

Read communication statuses are treated slightly differently than the others because they don't change independently. In other words, at least two changes will appear at the same time (DDS_DATA_ON_READERS_STATUS and DDS_DATA_AVAILABLE_STATUS) and even several of the last kind may be part of the set. This 'grouping' has to be communicated to the application.

For each plain communication status, there is a corresponding structure to hold the status value. These values contain the information related to the change of status, as well as information related to the statuses themselves (e.g., contains cumulative counts).

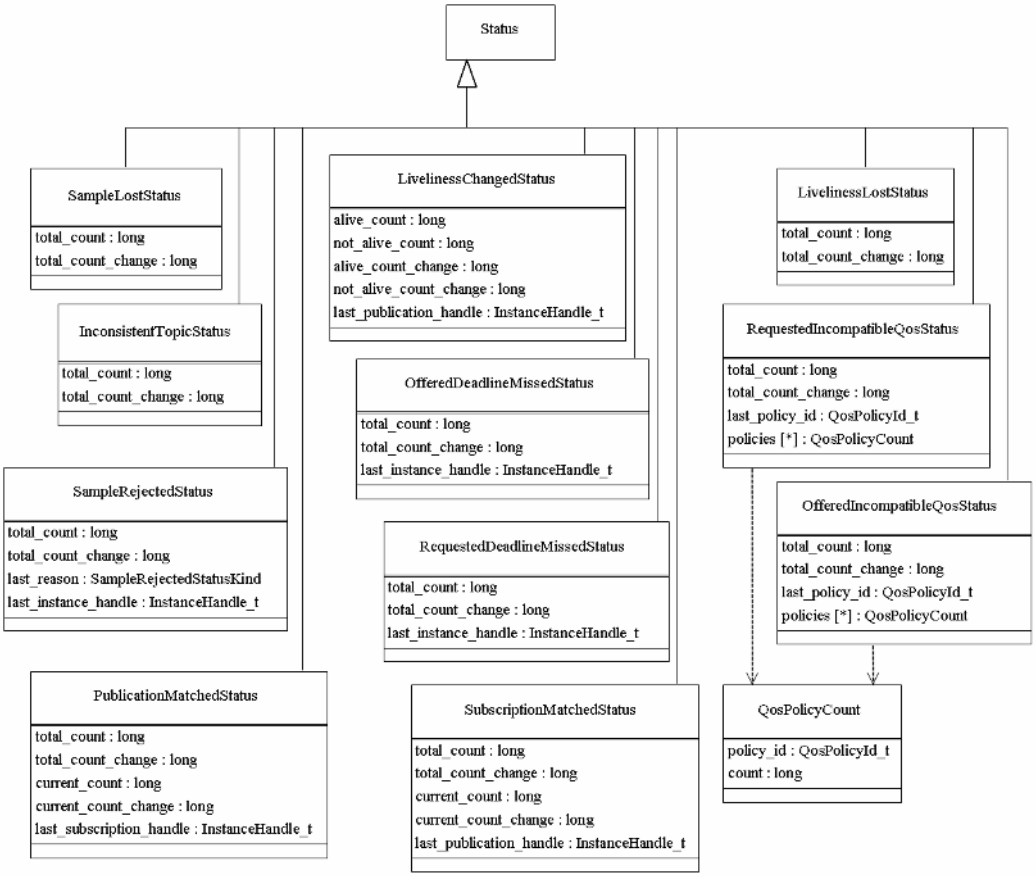

Figure 6: Status Values

5.61.2 Changes in Status

Associated with each one of an DDS_Entity's communication status is a logical `StatusChangedFlag`. This flag indicates whether that particular communication status has changed since the last time the status was read by the application. The way the status changes is slightly different for the Plain Communication Status and the Read Communication status.

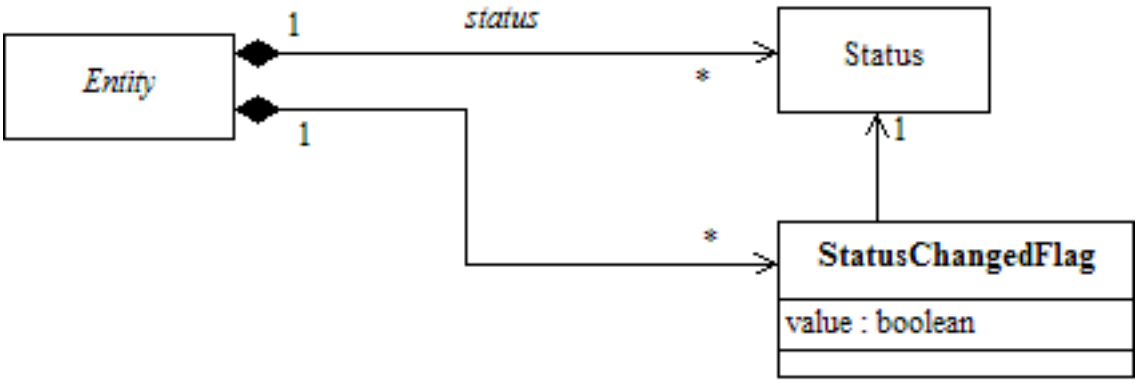

Figure 7: `StatusChangedFlag` indicates if status has changed

5.61.2.1 Changes in plain communication status

For the plain communication status, the `StatusChangedFlag` flag is initially set to FALSE. It becomes TRUE whenever the plain communication status changes and it is reset to DDS_BOOLEAN_FALSE each time the application accesses the plain communication status via the proper `get_<plain communication status>()` operation on the DDS_Entity.

The communication status is also reset to FALSE whenever the associated listener operation is called as the listener implicitly accesses the status which is passed as a parameter to the operation. The fact that the status is reset prior to calling the listener means that if the application calls the get_<plain communication status> from inside the listener it will see the status already reset.

An exception to this rule is when the associated listener is the 'nil' listener. The 'nil' listener is treated as a NOOP and the act of calling the 'nil' listener does not reset the communication status.

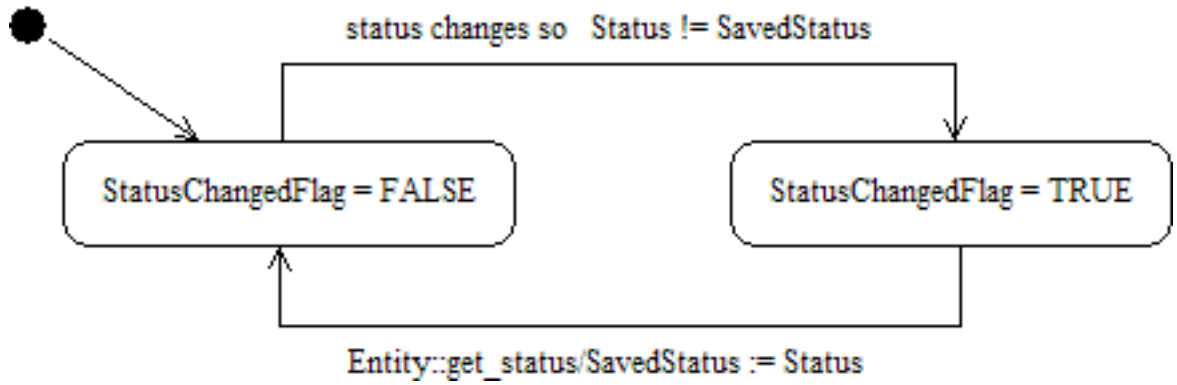

Figure 8: Changes in `StatusChangedFlag` for plain communication status

5.61.2.2 Changes in read communication status

For the read communication status, the `StatusChangedFlag` flag is initially set to FALSE. The `StatusChangedFlag` becomes TRUE when either a data-sample arrives or else the DDS_ViewStateKind, DDS_SampleStateKind, or DDS_InstanceStateKind of any existing sample changes for any reason other than a call to FooDataReader_read_next_sample, or FooDataReader_take_next_sample. Specifically any of the following events will cause the `StatusChangedFlag` to become TRUE:

- The arrival of new data.
- A change in the DDS_InstanceStateKind of a contained instance. This can be caused by either:
  - The arrival of the notification that an instance has been disposed by:

    * the DDS_DataWriter that owns it if OWNERSHIP QoS kind= DDS_EXCLUSIVE_OWNERSHIP_QOS
    * or by any DDS_DataWriter if OWNERSHIP QoS kind= DDS_SHARED_OWNERSHIP_QOS
  - The loss of liveliness of the DDS_DataWriter of an instance for which there is no other DDS_DataWriter.
  - The arrival of the notification that an instance has been unregistered by the only DDS_DataWriter that is known to be writing the instance.

Depending on the kind of `StatusChangedFlag`, the flag transitions to FALSE again as follows:

- The DDS_DATA_AVAILABLE_STATUS `StatusChangedFlag` becomes FALSE when either the corresponding listener operation (on_data_available) is called or the read_next_sample or take_next_sample operation is called on the associated DDS_DataReader.
- The DDS_DATA_ON_READERS_STATUS `StatusChangedFlag` becomes FALSE when any of the following events occurs:
  - The corresponding listener operation (on_data_on_readers) is called.
  - The on_data_available listener operation is called on any DDS_DataReader belonging to the DDS_Subscriber.
  - The read or take operation (or their variants) is called on any DDS_DataReader belonging to the DDS_Subscriber.

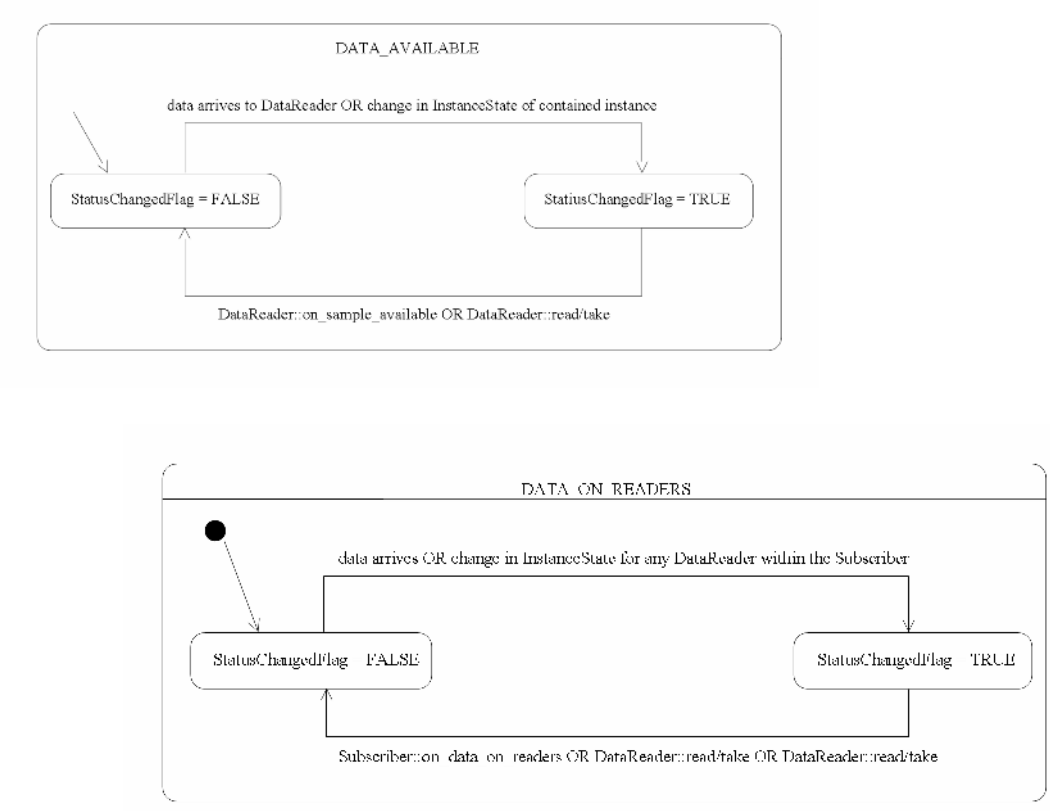

Figure 9: Changes in `StatusChangedFlag` for read communication status

See also

DDS_Listener

5.61.3 Macro Definition Documentation

5.61.3.1 #define DDS_STATUS_MASK_NONE

No bits are set.

5.61.3.2 #define DDS_STATUS_MASK_ALL

All bits are set.

5.61.4 Typedef Documentation

5.61.4.1 typedef DDS_UnsignedLong DDS_StatusMask

A bit-mask (list) of concrete status types, i.e. DDS_StatusKind[].

The bit-mask is an efficient and compact representation of a fixed-length list of DDS_StatusKind values.

Bits in the mask correspond to different statuses. You can choose which changes in status will trigger a callback by setting the corresponding status bits in this bit-mask and installing callbacks for each of those statuses.

The bits that are true indicate that the listener will be called back for changes in the corresponding status.

For example:

```
DDS_StatusMask mask = DDS_REQUESTED_DEADLINE_MISSED_STATUS
      |
                                    DDS_DATA_AVAILABLE_STATUS
     ;
                datareader->set_listener(listener, mask);
```

or

```
DDS_StatusMask mask = DDS_REQUESTED_DEADLINE_MISSED_STATUS
      |
                                    DDS_DATA_AVAILABLE_STATUS
     ;
                datareader = subscriber->create_datareader(topic,
                                                DDS_DATAREADER_QOS_DEFAULT
     ,
                                                listener, mask);
```

5.61.5 Enumeration Type Documentation

5.61.5.1 enum DDS_StatusKind

Type for *status* kinds.

Each concrete DDS_Entity is associated with a set of `*Status` objects whose values represent the communication status of that DDS_Entity.

The communication statuses whose changes can be communicated to the application depend on the DDS_Entity.

Each status value can be accessed with a corresponding function on the DDS_Entity. The changes on these status values cause activation of the corresponding trigger invocation of the proper DDS_Listener objects to asynchronously inform the application.

See also

DDS_Entity, DDS_Listener

Enumerator:

DDS_INCONSISTENT_TOPIC_STATUS Another topic exists with the same name but different characteristics.

Entity:

DDS_Topic

Status:

DDS_InconsistentTopicStatus

Listener:

DDS_TopicListener

DDS_OFFERED_DEADLINE_MISSED_STATUS The deadline that the DDS_DataWriter has committed through its DDS_DeadlineQosPolicy was not respected for a specific instance.

Entity:

DDS_DataWriter

QoS:

DEADLINE

Status:

DDS_OfferedDeadlineMissedStatus

Listener:

DDS_DataWriterListener

DDS_REQUESTED_DEADLINE_MISSED_STATUS The deadline that the DDS_DataReader was expecting through its DDS_DeadlineQosPolicy was not respected for a specific instance.

Entity:

DDS_DataReader

QoS:

DEADLINE

Status:

DDS_RequestedDeadlineMissedStatus

Listener:

DDS_DataReaderListener

DDS_OFFERED_INCOMPATIBLE_QOS_STATUS A QosPolicy value was incompatible with what was requested.

Entity:

DDS_DataWriter

Status:

DDS_OfferedIncompatibleQosStatus

Listener:

DDS_DataWriterListener

DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS A QosPolicy value was incompatible with what is offered.

Entity:

DDS_DataReader

Status:

DDS_RequestedIncompatibleQosStatus

Listener:

DDS_DataReaderListener

DDS_SAMPLE_LOST_STATUS A sample has been lost (i.e. was never received).

Entity:

DDS_Subscriber

Status:

DDS_SampleLostStatus

Listener:

DDS_SubscriberListener

DDS_SAMPLE_REJECTED_STATUS A (received) sample has been rejected.

Entity:

DDS_DataReader

QoS:

RESOURCE_LIMITS

Status:

DDS_SampleRejectedStatus

Listener:

DDS_DataReaderListener

DDS_DATA_ON_READERS_STATUS New data is available.

Entity:

DDS_Subscriber

Listener:

DDS_SubscriberListener

DDS_DATA_AVAILABLE_STATUS One or more new data samples have been received.

Entity:

DDS_DataReader

Listener:

DDS_DataReaderListener

DDS_LIVELINESS_LOST_STATUS The liveliness that the DDS_DataWriter has committed to through its DDS_LivelinessQosPolicy was not respected, thus DDS_DataReader entities will consider the DDS_DataWriter as no longer alive.

Entity:

DDS_DataWriter

QoS:

LIVELINESS

Status:

DDS_LivelinessLostStatus

Listener:

DDS_DataWriterListener

DDS_LIVELINESS_CHANGED_STATUS The liveliness of one or more DDS_DataWriter that were writing instances read through the DDS_DataReader has changed. Some DDS_DataWriter have become alive or not_alive.

Entity:

DDS_DataReader

QoS:

LIVELINESS

Status:

DDS_LivelinessChangedStatus

Listener:

DDS_DataReaderListener

DDS_PUBLICATION_MATCHED_STATUS The DDS_DataWriter has found DDS_DataReader that matches the DDS_Topic and has compatible QoS.

Entity:

DDS_DataWriter

Status:

DDS_PublicationMatchedStatus

Listener:

DDS_DataWriterListener

DDS_SUBSCRIPTION_MATCHED_STATUS The DDS_DataReader has found DDS_DataWriter that matches the DDS_Topic and has compatible QoS.

Entity:

DDS_DataReader

Status:

DDS_SubscriptionMatchedStatus

Listener:

DDS_DataReaderListener

DDS_INSTANCE_REPLACED_STATUS The DDS_DataReader has replaced an instance.

Entity:

DDS_DataReader

Status:

DDS_DataReaderInstanceReplacedStatus

Listener:

DDS_DataReaderListener

DDS_RELIABLE_READER_ACTIVITY_CHANGED_STATUS The DDS_DataWriter has detected a change of activity in a reliable DDS_DataReader.

Entity:

DDS_DataWriter

Status:

DDS_ReliableReaderActivityChangedStatus

Listener:

DDS_DataWriterListener

5.62 QoS Policies

Quality of Service (QoS) policies.

Modules

- DISCOVERY

  <<eXtension>> Specifies the attributes required to discover participants in the domain.
- DEADLINE

  Expresses the maximum duration (deadline) within which an instance is expected to be updated.
- OWNERSHIP

  Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated.
- OWNERSHIP_STRENGTH

  Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key).
- LIVELINESS

  Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive.
- RELIABILITY

  Indicates the level of reliability offered/requested by RTI Connext Micro.
- HISTORY

  Specifies the behavior of RTI Connext Micro in the case where the value of an instance changes (one or more times) before it can be successfully communicated to one or more existing subscribers.
- DURABILITY

  <<eXtension>> Specifies the Durability properties used by a DDS_DataWriter and/or DDS_DataReader.
- RESOURCE_LIMITS

  Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS.
- PRESENTATION

  Specifies how the samples representing changes to data instances are presented to a subscribing application.
- DESTINATION_ORDER

  <<eXtension>> Specifies the DestinationOrder policy.
- ENTITY_FACTORY

  A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types.
- Extended Qos Support

  <<eXtension>> Types and defines used in extended QoS policies.
- SYSTEM_RESOURCE_LIMITS

  <<eXtension>> Specifies the system-specific resources that RTI Connext Micro can use.
- WIRE_PROTOCOL

  <<eXtension>> Specifies the wire protocol related attributes for the DDS_DomainParticipant.
- DATA_READER_PROTOCOL

  <<eXtension>> Specifies the DataReader-specific protocol QoS.
- DATA_WRITER_PROTOCOL

  <<eXtension>> Specifies the DataWriter-specific protocol QoS.
- TRANSPORT

  <<eXtension>> <<cert>> Specify transport settings
- DOMAIN_PARTICIPANT_RESOURCE_LIMITS

  <<eXtension>> Specifies the DomainParticipant-specific resources that RTI Connext Micro can use.
- ENTITY_NAME

  <<eXtension>> Name of the entity.

Data Structures

- struct RTI_ManagementQosPolicy

  <<cert>> Policy used determine how DDS_DataReader, DDS_DataWriter, DDS_Publisher, and DDS_Subscriber is managed.

Enumerations

- enum DDS_QosPolicyId_t {
  DDS_INVALID_QOS_POLICY_ID , DDS_PRESENTATION_QOS_POLICY_ID, DDS_DEADLINE_QOS_POLICY_ID , DDS_OWNERSHIP_QOS_POLICY_ID,
  DDS_OWNERSHIPSTRENGTH_QOS_POLICY_ID, DDS_LIVELINESS_QOS_POLICY_ID , DDS_RELIABILITY_QOS_POLICY_ID , DDS_HISTORY_QOS_POLICY_ID ,
  DDS_ENTITYFACTORY_QOS_POLICY_ID }

  Type to identify `QosPolicies`.

5.62.1 Detailed Description

Quality of Service (QoS) policies. Data Distribution Service (DDS) relies on the use of QoS. A QoS is a set of characteristics that controls some aspect of the behavior of DDS. A QoS is comprised of individual QoS policies (objects conceptually deriving from an *abstract* `QosPolicy` class).

RTI Connext Micro supports a subset of these QoS.

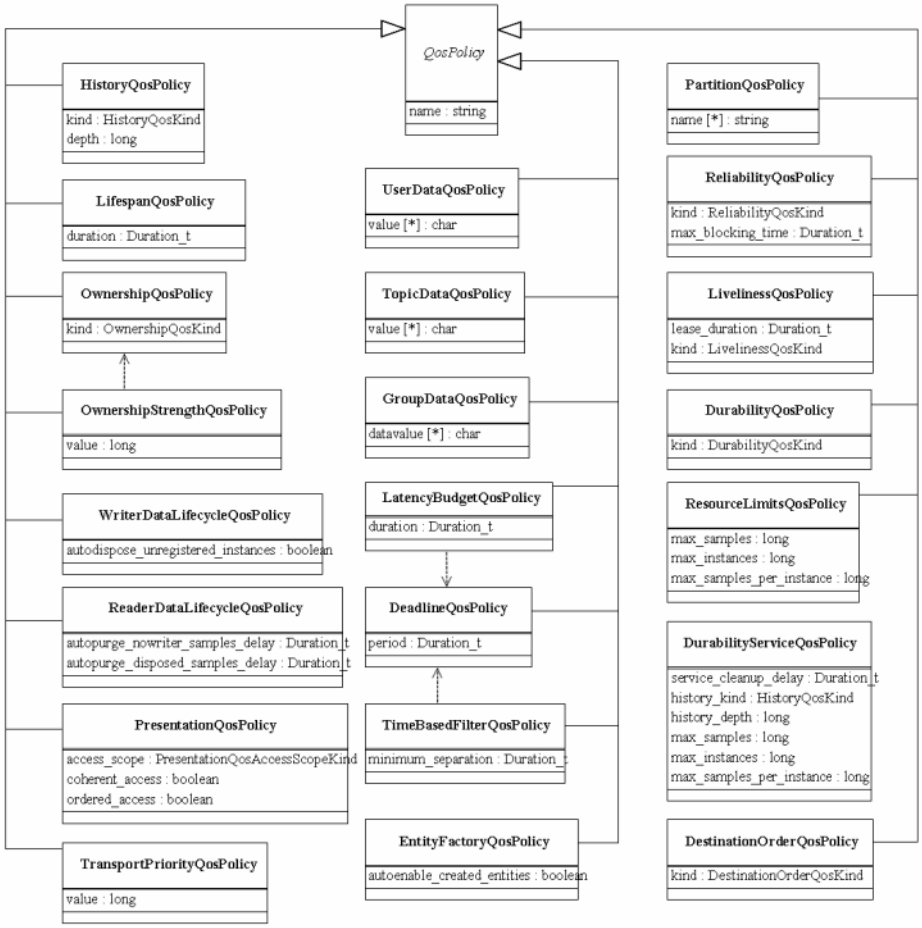

Figure 10: DDS QoS policies

The QosPolicy provides the basic mechanism for an application to specify quality of service parameters. It has an attribute name that is used to uniquely identify each `QosPolicy`.

QosPolicy implementation is comprised of a name, an ID, and a type. The type of a `QosPolicy` value may be atomic, such as an integer or float, or compound (a structure). Compound types are used whenever multiple parameters must be set coherently to define a consistent value for a QosPolicy.

QoS (i.e., a list of `QosPolicy` objects) may be associated with all DDS_Entity objects in the system such as DDS_Topic, DDS_DataWriter, DDS_DataReader, DDS_Publisher, DDS_Subscriber, and DDS_DomainParticipant.

5.62.2 Specifying QoS on entities

QosPolicies can be set programmatically when an DDS_Entity is created, or modified with the DDS_Entity's set_qos (abstract) function.

To customize a DDS_Entity's QoS before creating the entity, the correct pattern is:

- First, initialize a QoS object with the appropriate INITIALIZER constant.
- Call the relevant get_<entity>_default_qos() method.
- Modify the QoS values as desired.
- Finally, create the entity.

Each `QosPolicy` is treated independently from the others. This approach has the advantage of being very extensible. However, there may be cases where several policies are in conflict. Consistency checking is performed each time the policies are modified via the set_qos (abstract) operation, or when the DDS_Entity is created.

When a policy is changed after being set to a given value, it is not required that the new value be applied instantaneously; RTI Connext Micro is allowed to apply it after a transition phase. In addition, some `QosPolicy` have immutable semantics, meaning that they can only be specified either at DDS_Entity creation time or else prior to calling the DDS_Entity_enable operation on the entity.

Each DDS_Entity can be configured with a list of `QosPolicy` objects. However, not all QosPolicies are supported by each DDS_Entity. For instance, a DDS_DomainParticipant supports a different set of QosPolicies than a DDS_Topic or a DDS_Publisher.

5.62.3 QoS compatibility

In several cases, for communications to occur properly (or efficiently), a QosPolicy on the publisher side must be compatible with a corresponding policy on the subscriber side. For example, if a DDS_Subscriber requests to receive data reliably while the corresponding DDS_Publisher defines a best-effort policy, communication will not happen as requested.

To address this issue and maintain the desirable decoupling of publication and subscription as much as possible, the QosPolicy specification follows the **subscriber-requested, publisher-offered pattern**.

In this pattern, the subscriber side can specify a "requested" value for a particular `QosPolicy`. The publisher side specifes an "offered" value for that `QosPolicy`. RTI Connext Micro will then determine whether the value requested by the subscriber side is compatible with what is offered by the publisher side. If the two policies are compatible, then communication will be established. If the two policies are not compatible, RTI Connext Micro will not establish communications between the two DDS_Entity objects and will record this fact by means of the DDS_OFFERED_INCOMPATIBLE_QOS_STATUS on the publisher end and DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS on the subscriber end. The application can detect this fact by means of a DDS_Listener.

The following **properties** are defined on a QosPolicy.

- **RxO property**

  The QosPolicy objects that need to be set in a compatible manner between the **publisher** and **subscriber** end are indicated by the setting of the RxO property:

  - RxO = **YES** indicates that the policy can be set both at the publishing and subscribing ends and the values must be set in a compatible manner. In this case the compatible values are explicitly defined.
  - RxO = **NO** indicates that the policy can be set both at the publishing and subscribing ends but the two settings are independent. That is, all combinations of values are compatible.
  - RxO = **N/A** indicates that the policy can only be specified at either the publishing or the subscribing end, but not at both ends. So compatibility does not apply.

- **Changeable property**

  Determines whether a QosPolicy can be changed.

  NO – policy can only be specified at DDS_Entity creation time.

  UNTIL ENABLE – policy can only be changed before the DDS_Entity is enabled.

  YES – policy can be changed at any time.

5.62.4 Enumeration Type Documentation

5.62.4.1 enum DDS_QosPolicyId_t

Type to identify `QosPolicies`.

Note that the value of these constants disagree with the values of the corresponding parameter IDs in the RTPS protocol. This conflict is unavoidable since these values are given in the DDS specification, which is not tied to RTPS.

Enumerator:

DDS_INVALID_QOS_POLICY_ID Identifier for an invalid QoS policy.

DDS_PRESENTATION_QOS_POLICY_ID Identifier for DDS_PresentationQosPolicy.

DDS_DEADLINE_QOS_POLICY_ID Identifier for DDS_DeadlineQosPolicy.

DDS_OWNERSHIP_QOS_POLICY_ID Identifier for DDS_OwnershipQosPolicy.

DDS_OWNERSHIPSTRENGTH_QOS_POLICY_ID Identifier for DDS_OwnershipStrengthQosPolicy.

DDS_LIVELINESS_QOS_POLICY_ID Identifier for DDS_LivelinessQosPolicy.

DDS_RELIABILITY_QOS_POLICY_ID Identifier for DDS_ReliabilityQosPolicy.

DDS_HISTORY_QOS_POLICY_ID Identifier for DDS_HistoryQosPolicy.

DDS_ENTITYFACTORY_QOS_POLICY_ID Identifier for DDS_EntityFactoryQosPolicy.

5.63 DEADLINE

Expresses the maximum duration (deadline) within which an instance is expected to be updated.

Data Structures

- struct DDS_DeadlineQosPolicy

  <<cert>> Expresses the maximum duration (deadline) within which an instance is expected to be updated.

Variables

- const char *const DDS_DEADLINE_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DeadlineQosPolicy.

5.63.1 Detailed Description

Expresses the maximum duration (deadline) within which an instance is expected to be updated.

5.63.2 Variable Documentation

5.63.2.1 const char* const DDS_DEADLINE_QOS_POLICY_NAME

Stringified human-readable name for DDS_DeadlineQosPolicy.

5.64 OWNERSHIP

Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated.

Data Structures

- struct DDS_OwnershipQosPolicy

  <<cert>> Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated.

Enumerations

- enum DDS_OwnershipQosPolicyKind { DDS_SHARED_OWNERSHIP_QOS, DDS_EXCLUSIVE_OWNERSHIP_QOS }

  Kinds of ownership.

Variables

- const char *const DDS_OWNERSHIP_QOS_POLICY_NAME

  Stringified human-readable name for DDS_OwnershipQosPolicy.

5.64.1 Detailed Description

Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated.

5.64.2 Enumeration Type Documentation

5.64.2.1 enum DDS_OwnershipQosPolicyKind

Kinds of ownership.

QoS:

DDS_OwnershipQosPolicy

Enumerator:

DDS_SHARED_OWNERSHIP_QOS **[default]** Indicates shared ownership for each instance. Multiple writers are allowed to update the same instance and all the updates are made available to the readers. In other words there is no concept of an owner for the instances.

This is the **default** behavior if the OWNERSHIP policy is not specified or supported.

DDS_EXCLUSIVE_OWNERSHIP_QOS Indicates each instance can only be owned by one DDS_DataWriter, but the owner of an instance can change dynamically. The selection of the owner is controlled by the setting of the OWNERSHIP_STRENGTH policy. The owner is always set to be the highest-strength DDS_DataWriter object among the ones currently active (as determined by the LIVELINESS).

5.64.3 Variable Documentation

5.64.3.1 const char* const DDS_OWNERSHIP_QOS_POLICY_NAME

Stringified human-readable name for DDS_OwnershipQosPolicy.

5.65 OWNERSHIP_STRENGTH

Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key).

Data Structures

- struct DDS_OwnershipStrengthQosPolicy

  <<cert>> Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key).

Variables

- const char *const DDS_OWNERSHIPSTRENGTH_QOS_POLICY_NAME

  Stringified human-readable name for DDS_OwnershipStrengthQosPolicy.

5.65.1 Detailed Description

Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key).

5.65.2 Variable Documentation

5.65.2.1 const char* const DDS_OWNERSHIPSTRENGTH_QOS_POLICY_NAME

Stringified human-readable name for DDS_OwnershipStrengthQosPolicy.

5.66 LIVELINESS

Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive.

Data Structures

- struct DDS_LivelinessQosPolicy

  <<cert>> Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive.

Enumerations

- enum DDS_LivelinessQosPolicyKind { DDS_AUTOMATIC_LIVELINESS_QOS, DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS, DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS }

  Kinds of liveliness.

Variables

- const char *const DDS_LIVELINESS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_LivelinessQosPolicy.

5.66.1 Detailed Description

Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive.

5.66.2 Enumeration Type Documentation

5.66.2.1 enum DDS_LivelinessQosPolicyKind

Kinds of liveliness.

QoS:

DDS_LivelinessQosPolicy

Enumerator:

DDS_AUTOMATIC_LIVELINESS_QOS **[default]** The infrastructure will automatically signal liveliness for the DDS_DataWriter (s) at least as often as required by the DDS_DataWriter (S) `lease_duration`. A DDS_DataWriter with this setting does not need to take any specific action in order to be considered 'alive.' The DDS_DataWriter is only 'not alive' when the participant to which it belongs terminates (gracefully or not), or when there is a network problem that prevents the current participant from contacting that remote participant.

Only DDS_DURATION_INFINITE is supported for `lease_duration`. Finite `lease_duration` is not supported with this liveliness setting.

DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS RTI Connext Micro will assume that as long as at least one DDS_DataWriter belonging to the DDS_DomainParticipant (or the DDS_DomainParticipant itself) has asserted its liveliness, then the other DataWriters belonging to that same DDS_DomainParticipant are also alive. Only DDS_DURATION_INFINITE is supported for `lease_duration`. Finite `lease_duration` is not supported with this liveliness setting.

The user application takes responsibility to signal liveliness to RTI Connext Micro by calling DDS_DataWriter_assert_liveliness, or FooDataWriter_write on any DDS_DataWriter belonging to the DDS_DomainParticipant.

DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS RTI Connext Micro will only assume liveliness of the DDS_DataWriter if the application has asserted liveliness of that DDS_DataWriter itself. The user application takes responsibility to signal liveliness to RTI Connext Micro using the DDS_DataWriter_assert_liveliness function, or by writing some data.

5.66.3 Variable Documentation

5.66.3.1 const char* const DDS_LIVELINESS_QOS_POLICY_NAME

Stringified human-readable name for DDS_LivelinessQosPolicy.

5.67 RELIABILITY

Indicates the level of reliability offered/requested by RTI Connext Micro.

Data Structures

- struct DDS_ReliabilityQosPolicy
  <<cert>> Indicates the level of reliability offered/requested by RTI Connext Micro.

Enumerations

- enum DDS_ReliabilityQosPolicyKind { DDS_BEST_EFFORT_RELIABILITY_QOS, DDS_RELIABLE_RELIABILITY_QOS }
  Kinds of reliability.

Variables

- const char *const DDS_RELIABILITY_QOS_POLICY_NAME
  Stringified human-readable name for DDS_ReliabilityQosPolicy.

5.67.1 Detailed Description

Indicates the level of reliability offered/requested by RTI Connext Micro.

5.67.2 Enumeration Type Documentation

5.67.2.1 enum DDS_ReliabilityQosPolicyKind

Kinds of reliability.

QoS:

DDS_ReliabilityQosPolicy

Enumerator:

DDS_BEST_EFFORT_RELIABILITY_QOS Indicates that it is acceptable to not retry propagation of any samples. Presumably new values for the samples are generated often enough that it is not necessary to re-send or acknowledge any samples.
[default]

DDS_RELIABLE_RELIABILITY_QOS Specifies RTI Connext Micro will attempt to deliver all samples in its history. Missed samples may be retried. In steady-state (no modifications communicated via the DDS_DataWriter) the middleware guarantees that all samples in the DDS_DataWriter history will eventually be delivered to all the DDS_DataReader objects (subject to timeouts that indicate loss of communication with a particular DDS_Subscriber).
Outside steady state the HISTORY and RESOURCE_LIMITS policies will determine how samples become part of the history and whether samples can be discarded from it.
NOTE: The RTPS spec defines reliability as 0x03 to comply with RTPS spec. However, RTI Connext Core uses 0x3, as well as others.

5.67.3 Variable Documentation

5.67.3.1 const char* const DDS_RELIABILITY_QOS_POLICY_NAME

Stringified human-readable name for DDS_ReliabilityQosPolicy.

5.68 HISTORY

Specifies the behavior of RTI Connext Micro in the case where the value of an instance changes (one or more times) before it can be successfully communicated to one or more existing subscribers.

Data Structures

- struct DDS_HistoryQosPolicy

  <<cert>> Specifies the behavior of RTI Connext Micro in the case where the value of a sample changes (one or more times) before it can be successfully communicated to one or more existing subscribers.

Enumerations

- enum DDS_HistoryQosPolicyKind { DDS_KEEP_LAST_HISTORY_QOS }

  Kinds of history.

Variables

- const char *const DDS_HISTORY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_HistoryQosPolicy.

5.68.1 Detailed Description

Specifies the behavior of RTI Connext Micro in the case where the value of an instance changes (one or more times) before it can be successfully communicated to one or more existing subscribers.

5.68.2 Enumeration Type Documentation

5.68.2.1 enum DDS_HistoryQosPolicyKind

Kinds of history.

QoS:

DDS_HistoryQosPolicy

Enumerator:

DDS_KEEP_LAST_HISTORY_QOS **[default]** Keep the last `depth` samples. On the publishing side, RTI Connext Micro will only attempt to keep the most recent `depth` samples of each instance of data (identified by its key) managed by the DDS_DataWriter.

On the subscribing side, the DDS_DataReader will only attempt to keep the most recent `depth` samples received for each instance (identified by its key) until the application takes them via the DDS_DataReader 's **take()** operation.

5.68.3 Variable Documentation

5.68.3.1 const char* const DDS_HISTORY_QOS_POLICY_NAME

Stringified human-readable name for DDS_HistoryQosPolicy.

5.69 DURABILITY

<<*eXtension*>> Specifies the Durability properties used by a DDS_DataWriter and/or DDS_DataReader.

Data Structures

- struct DDS_DurabilityQosPolicy

  Durability properties that affect the life-cycle of published data.

Enumerations

- enum DDS_DurabilityQosPolicyKind { DDS_VOLATILE_DURABILITY_QOS, DDS_TRANSIENT_LOCAL_DURABILITY_QOS }

  Enumerated values used to describe the durability of published data.

Variables

- const char *const DDS_DURABILITY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DurabilityQosPolicy.

5.69.1 Detailed Description

<<*eXtension*>> Specifies the Durability properties used by a DDS_DataWriter and/or DDS_DataReader.

5.69.2 Enumeration Type Documentation

5.69.2.1 enum DDS_DurabilityQosPolicyKind

Enumerated values used to describe the durability of published data.

Enumerator:

DDS_VOLATILE_DURABILITY_QOS Volatile durability.

DDS_TRANSIENT_LOCAL_DURABILITY_QOS Transient local durability.

5.69.3 Variable Documentation

5.69.3.1 const char* const DDS_DURABILITY_QOS_POLICY_NAME

Stringified human-readable name for DDS_DurabilityQosPolicy.

5.70 RESOURCE_LIMITS

Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS.

Data Structures

- struct DDS_ResourceLimitsQosPolicy

  <<cert>> Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS.

Variables

- const char *const DDS_RESOURCELIMITS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_ResourceLimitsQosPolicy.
- const DDS_Long DDS_LENGTH_UNLIMITED

  A special value indicating an unlimited quantity.
- const DDS_Long DDS_LENGTH_AUTO

  A special value indicating the resource-limit is determined based on other settings.

5.70.1 Detailed Description

Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS.

5.70.2 Variable Documentation

5.70.2.1 const char* const DDS_RESOURCELIMITS_QOS_POLICY_NAME

Stringified human-readable name for DDS_ResourceLimitsQosPolicy.

5.70.2.2 const DDS_Long DDS_LENGTH_UNLIMITED

A special value indicating an unlimited quantity.

5.70.2.3 const DDS_Long DDS_LENGTH_AUTO

A special value indicating the resource-limit is determined based on other settings.

Some resource-limits can be derived from other limits using rules. However, the rules are usually conservative and typically the resulting limit is larger than needed. Thus, while this is a useful value when starting development, it should be adjusted properly when the exact limits are known.

5.71 PRESENTATION

Specifies how the samples representing changes to data instances are presented to a subscribing application.

Data Structures

- struct DDS_PresentationQosPolicy

  Specifies how the samples representing changes to data instances are presented to a subscribing application.

Enumerations

- enum DDS_PresentationQosPolicyAccessScopeKind { DDS_INSTANCE_PRESENTATION_QOS, DDS_TOPIC_PRESENTATION_QOS, DDS_GROUP_PRESENTATION_QOS, DDS_HIGHEST_OFFERED_PRESENTATION_QOS }

  Kinds of presentation "access scope".

5.71.1 Detailed Description

Specifies how the samples representing changes to data instances are presented to a subscribing application.

5.71.2 Enumeration Type Documentation

5.71.2.1 enum DDS_PresentationQosPolicyAccessScopeKind

Kinds of presentation "access scope".

Access scope determines the largest scope spanning the entities for which the order and coherency of changes can be preserved.

QoS:

DDS_PresentationQosPolicy

Enumerator:

DDS_INSTANCE_PRESENTATION_QOS **[default]** for DDS_DataReader. Scope spans only a single instance. Indicates that changes to one instance need not be coherent nor ordered with respect to changes to any other instance. In other words, order and coherent changes apply to each instance separately.

DDS_TOPIC_PRESENTATION_QOS **[default]** for DDS_DataWriter. Scope spans all instances within the same DDS_DataWriter (or DDS_DataReader), but not across instances in different DDS_DataWriter (or DDS_DataReader entities.

DDS_GROUP_PRESENTATION_QOS Scope spans all instances belonging to DDS_DataWriter (or DDS_DataReader) entities within the same DDS_Publisher (or DDS_Subscriber).

DDS_HIGHEST_OFFERED_PRESENTATION_QOS This value only applies to a DDS_Subscriber. The DDS_Subscriber will use the access scope specified by each remote DDS_Publisher.

5.72 DESTINATION_ORDER

<<eXtension>> Specifies the DestinationOrder policy.

Data Structures

- struct DDS_DestinationOrderQosPolicy

  Destination order policies that affect the ordering of subsribed data.

Enumerations

- enum DDS_DestinationOrderQosPolicyKind { DDS_BY_RECEPTION_TIMESTAMP_DESTINATIONORDER_QOS, DDS_BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS }

  Enumerated values used to describe the destination order of published data.

5.72.1 Detailed Description

<<eXtension>> Specifies the DestinationOrder policy.

5.72.2 Enumeration Type Documentation

5.72.2.1 enum DDS_DestinationOrderQosPolicyKind

Enumerated values used to describe the destination order of published data.

Enumerator:

DDS_BY_RECEPTION_TIMESTAMP_DESTINATIONORDER_QOS destination ordering by reception timestemp.

Entity:

DDS_DataReader, DDS_DataWriter

Properties:

RxO = YES;
Changeable = NO

[default] Indicates that data is ordered based on the reception time at each DDS_DataReader.

Since each subscriber may receive the data at different times there is no guarantee that the changes will be seen in the same order. Consequently, it is possible for each subscriber to end up with a different final value for the data.

DDS_BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS destination ordering by source timestemp.

Entity:

DDS_DataWriter

Properties:

RxO = YES;
Changeable = NO

Indicates that data is ordered based on the source time provided by the source application.

A consistent final value is guaranteed for the data in all subscribers.

5.73 DataReader Resource Limits

<<eXtension>> Specifies the DataReader-specific resources that RTI Connext Micro can use.

Data Structures

- struct DDS_DataReaderResourceLimitsQosPolicy

  <<cert>> *Resource limits that apply only to DDS_DataReader instances.*

Enumerations

- enum DDS_DataReaderResourceLimitsInstanceReplacementKind { DDS_NO_INSTANCE_REPLACEMENT_QOS, DDS_REPLACE_OLDEST_INSTANCE_REPLACEMENT_QOS }

  Kinds of instance replacement.

Variables

- const char *const DDS_DATAREADERRESOURCELIMITS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DataReaderResourceLimitsQosPolicy.

5.73.1 Detailed Description

<<eXtension>> Specifies the DataReader-specific resources that RTI Connext Micro can use.

5.73.2 Enumeration Type Documentation

5.73.2.1 enum DDS_DataReaderResourceLimitsInstanceReplacementKind

Kinds of instance replacement.

QoS:

DDS_DataReaderResourceLimitsQosPolicy

Enumerator:

DDS_NO_INSTANCE_REPLACEMENT_QOS No instances will be replaced when a new instance is detected and resources are exhausted.

DDS_REPLACE_OLDEST_INSTANCE_REPLACEMENT_QOS Replace the oldest instance when out of instance resources. When a new instance is detected and resources are exhausted the oldest instance will be removed even if samples with a not read state exists for the instance. Thus, this is a forceful removal of samples and the instance. The freed instance resources will be re-used for the newly detected instance. If there are outstanding loans for the replaced instance the samples can not be freed until returned. However, when the loan is returned the samples are returned back to the free sample pool. Note that the instance will not be rejected.

5.73.3 Variable Documentation

5.73.3.1 const char* const DDS_DATAREADERRESOURCELIMITS_QOS_POLICY_NAME

Stringified human-readable name for DDS_DataReaderResourceLimitsQosPolicy.

5.74 DataWriter Resource Limits

<<eXtension>> Specifies the DataWriter-specific resources that RTI Connext Micro can use.

Data Structures

- struct DDS_DataWriterResourceLimitsQosPolicy
  <<cert>> *Resource limits that apply only to DDS_DataWriter instances.*

Variables

- const char *const DDS_DATAWRITERRESOURCELIMITS_QOS_POLICY_NAME
  Stringified human-readable name for DDS_DataWriterResourceLimitsQosPolicy.

5.74.1 Detailed Description

<<eXtension>> Specifies the DataWriter-specific resources that RTI Connext Micro can use.

5.74.2 Variable Documentation

5.74.2.1 const char* const DDS_DATAWRITERRESOURCELIMITS_QOS_POLICY_NAME

Stringified human-readable name for DDS_DataWriterResourceLimitsQosPolicy.

5.75 ENTITY_FACTORY

A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types.

Data Structures

- struct DDS_EntityFactoryQosPolicy
  <<cert>> A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types.

Variables

- const char *const DDS_ENTITYFACTORY_QOS_POLICY_NAME
  Stringified human-readable name for DDS_EntityFactoryQosPolicy.

5.75.1 Detailed Description

A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types.

5.75.2 Variable Documentation

5.75.2.1 const char* const DDS_ENTITYFACTORY_QOS_POLICY_NAME

Stringified human-readable name for DDS_EntityFactoryQosPolicy.

5.76 Extended Qos Support

<<eXtension>> Types and defines used in extended QoS policies.

Modules

- DataReader Resource Limits

  <<eXtension>> Specifies the DataReader-specific resources that RTI Connext Micro can use.
- DataWriter Resource Limits

  <<eXtension>> Specifies the DataWriter-specific resources that RTI Connext Micro can use.

Data Structures

- struct DDS_RtpsReliableReaderProtocol_t

  Qos related to reliable reader protocol defined in RTPS.
- struct DDS_RtpsReliableWriterProtocol_t

  <<cert>> QoS related to the reliable writer protocol defined in RTPS.

5.76.1 Detailed Description

<<eXtension>> Types and defines used in extended QoS policies.

5.77 SYSTEM_RESOURCE_LIMITS

<<eXtension>> Specifies the system-specific resources that RTI Connext Micro can use.

Data Structures

- struct DDS_SystemResourceLimitsQosPolicy
  <<cert>> *Resource limits that apply only to DDS_DomainParticipantFactory*

Variables

- const char *const DDS_SYSTEMRESOURCELIMITS_QOS_POLICY_NAME
  Stringified human-readable name for DDS_SystemResourceLimitsQosPolicy.

5.77.1 Detailed Description

<<eXtension>> Specifies the system-specific resources that RTI Connext Micro can use.

5.77.2 Variable Documentation

5.77.2.1 const char* const DDS_SYSTEMRESOURCELIMITS_QOS_POLICY_NAME

Stringified human-readable name for DDS_SystemResourceLimitsQosPolicy.

5.78 WIRE_PROTOCOL

<<*eXtension*>> Specifies the wire protocol related attributes for the DDS_DomainParticipant.

Data Structures

- struct DDS_RtpsWellKnownPorts_t

  RTPS well-known port mapping configuration.
- struct DDS_WireProtocolQosPolicy

  <<*cert*>> *Specifies the wire-protocol-related attributes for the DDS_DomainParticipant.*

Macros

- #define DDS_CHECKSUM_NONE (0)

  Bitmap indicating no checksum selection.
- #define DDS_CHECKSUM_AUTO (0xffffU)

  Bitmap indicating automatic checksum selection.
- #define DDS_CHECKSUM_BUILTIN32 (0x1U)

  Bitmap for selecting the builtin 32bit checksum.
- #define DDS_CHECKSUM_BUILTIN64 (0x2U)

  Bitmap for selecting the builtin 64bit checksum.
- #define DDS_CHECKSUM_BUILTIN128 (0x4U)

  Bitmap for selecting the builtin 128bit checksum.

Typedefs

- typedef DDS_UnsignedShort DDS_ChecksumKind_t

  Datatype for checksum kind.
- typedef DDS_UnsignedShort DDS_ChecksumKindMask_t

  Datatype for an OR'ing for DDS_ChecksumKind_t .

Variables

- struct DDS_RtpsWellKnownPorts_t DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WELL_KNOWN_PORTS

  Assign to use well-known port mappings which are compatible with previous versions of the RTI Connext Micro middleware.
- struct DDS_RtpsWellKnownPorts_t DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS

  Assign to use well-known port mappings which are compliant with OMG's DDS Interoperability Wire Protocol.
- const char ∗const DDS_WIREPROTOCOL_QOS_POLICY_NAME

  Stringified human-readable name for DDS_WireProtocolQosPolicy.

5.78.1 Detailed Description

<<*eXtension*>> Specifies the wire protocol related attributes for the DDS_DomainParticipant.

5.78.2 Macro Definition Documentation

5.78.2.1 #define DDS_CHECKSUM_NONE (0)

Bitmap indicating no checksum selection.

QoS:

DDS_WireProtocolQosPolicy

5.78.2.2 #define DDS_CHECKSUM_AUTO (0xffffU)

Bitmap indicating automatic checksum selection.

QoS:

DDS_WireProtocolQosPolicy

5.78.2.3 #define DDS_CHECKSUM_BUILTIN32 (0x1U)

Bitmap for selecting the builtin 32bit checksum.

QoS:

DDS_WireProtocolQosPolicy

5.78.2.4 #define DDS_CHECKSUM_BUILTIN64 (0x2U)

Bitmap for selecting the builtin 64bit checksum.

QoS:

DDS_WireProtocolQosPolicy

5.78.2.5 #define DDS_CHECKSUM_BUILTIN128 (0x4U)

Bitmap for selecting the builtin 128bit checksum.

QoS:

DDS_WireProtocolQosPolicy

5.78.3 Typedef Documentation

5.78.3.1 typedef DDS_UnsignedShort DDS_ChecksumKind_t

Datatype for checksum kind.

QoS:

DDS_WireProtocolQosPolicy

5.78.3.2 typedef DDS_UnsignedShort DDS_ChecksumKindMask_t

Datatype for an OR'ing for DDS_ChecksumKind_t .

QoS:

DDS_WireProtocolQosPolicy

5.78.4 Variable Documentation

5.78.4.1 struct DDS_RtpsWellKnownPorts_t DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WELL_KNOWN_PORTS

Assign to use well-known port mappings which are compatible with previous versions of the RTI Connext Micro middleware.

Assign DDS_WireProtocolQosPolicy::rtps_well_known_ports to this value to remain compatible with previous versions of the RTI Connext Micro middleware that used fixed port mappings.

The following are the `rtps_well_known_ports` values for DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WELL_KNOWN_PORTS:

`port_base` = 7400

`domain_id_gain` = 10

`participant_id_gain` = 1000

`builtin_multicast_port_offset` = 2

`builtin_unicast_port_offset` = 0

`user_multicast_port_offset` = 1

`user_unicast_port_offset` = 3

These settings are *not* compliant with OMG's DDS Interoperability Wire Protocol. To comply with the specification, please use DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS.

See also

DDS_WireProtocolQosPolicy::rtps_well_known_ports
DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS

5.78.4.2 struct DDS_RtpsWellKnownPorts_t DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS

Assign to use well-known port mappings which are compliant with OMG's DDS Interoperability Wire Protocol.

Assign DDS_WireProtocolQosPolicy::rtps_well_known_ports to this value to use well-known port mappings which are compliant with OMG's DDS Interoperability Wire Protocol.

The following are the `rtps_well_known_ports` values for DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS:

`port_base` = 7400

`domain_id_gain` = 250

`participant_id_gain` = 2

`builtin_multicast_port_offset` = 0

`builtin_unicast_port_offset` = 10

`user_multicast_port_offset` = 1

`user_unicast_port_offset` = 11

Assuming a maximum port number of 65535 (UDPv4), the above settings enable the use of about 230 domains with up to 120 Participants per node per domain.

These settings are *not* backwards compatible with previous versions of the RTI Connext Micro middleware that used fixed port mappings. For backwards compability, please use DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WELL_KNOWN_PORTS.

See also

DDS_WireProtocolQosPolicy::rtps_well_known_ports
DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WELL_KNOWN_PORTS

5.78.4.3 const char* const DDS_WIREPROTOCOL_QOS_POLICY_NAME

Stringified human-readable name for DDS_WireProtocolQosPolicy.

5.79 DATA_READER_PROTOCOL

<<eXtension>> Specifies the DataReader-specific protocol QoS.

Data Structures

- struct DDS_DataReaderProtocolQosPolicy
  <<eXtension>> *Protocol that applies only to DDS_DataReader instances.*

Variables

- const char *const DDS_DATAREADERPROTOCOL_QOS_POLICY_NAME
  Stringified human-readable name for DDS_DataReaderProtocolQosPolicy.

5.79.1 Detailed Description

<<eXtension>> Specifies the DataReader-specific protocol QoS.

5.79.2 Variable Documentation

5.79.2.1 const char* const DDS_DATAREADERPROTOCOL_QOS_POLICY_NAME

Stringified human-readable name for DDS_DataReaderProtocolQosPolicy.

5.80 DATA_WRITER_PROTOCOL

<<eXtension>> Specifies the DataWriter-specific protocol QoS.

Data Structures

- struct DDS_DataWriterProtocolQosPolicy
  <<eXtension>> <<cert>> Protocol that applies only to DDS_DataWriter instances.

Variables

- const char *const DDS_DATAWRITERPROTOCOL_QOS_POLICY_NAME
  Stringified human-readable name for DDS_DataWriterProtocolQosPolicy.

5.80.1 Detailed Description

<<eXtension>> Specifies the DataWriter-specific protocol QoS.

5.80.2 Variable Documentation

5.80.2.1 const char* const DDS_DATAWRITERPROTOCOL_QOS_POLICY_NAME

Stringified human-readable name for DDS_DataWriterProtocolQosPolicy.

5.81 TRANSPORT

<<eXtension>> <<cert>> Specify transport settings

Data Structures

- struct DDS_TransportQosPolicy
  <<eXtension>> <<cert>> Settings for available transports

Variables

- const char *const DDS_TRANSPORT_QOS_POLICY_NAME
  Stringified human-readable name for DDS_TransportQosPolicy.

5.81.1 Detailed Description

<<eXtension>> <<cert>> Specify transport settings

5.81.2 Variable Documentation

5.81.2.1 const char* const DDS_TRANSPORT_QOS_POLICY_NAME

Stringified human-readable name for DDS_TransportQosPolicy.

5.82 DOMAIN_PARTICIPANT_RESOURCE_LIMITS

<<eXtension>> Specifies the DomainParticipant-specific resources that RTI Connext Micro can use.

Data Structures

- struct DDS_DomainParticipantResourceLimitsQosPolicy
  <<cert>> *Resource limits that apply only to DDS_DomainParticipant instances.*

Variables

- const char *const DDS_DOMAINPARTICIPANTRESOURCELIMITS_QOS_POLICY_NAME
  Stringified human-readable name for DDS_DomainParticipantResourceLimitsQosPolicy.

5.82.1 Detailed Description

<<eXtension>> Specifies the DomainParticipant-specific resources that RTI Connext Micro can use.

5.82.2 Variable Documentation

5.82.2.1 const char* const DDS_DOMAINPARTICIPANTRESOURCELIMITS_QOS_POLICY_NAME

Stringified human-readable name for DDS_DomainParticipantResourceLimitsQosPolicy.

5.83 Conditions and WaitSets

DDS_Condition and DDS_WaitSet and related items.

Typedefs

- typedef struct DDS_ConditionImpl DDS_Condition

  <<interface>> <<cert>> Root class for all the conditions that may be attached to a DDS_WaitSet.
- typedef struct DDS_GuardConditionImpl DDS_GuardCondition

  <<interface>> <<cert>> A specific DDS_Condition whose `trigger_value` is completely under the control of the application.
- typedef struct DDS_StatusConditionImpl DDS_StatusCondition

  <<interface>> <<cert>> A specific DDS_Condition that is associated with each DDS_Entity.
- typedef struct DDS_WaitSetImpl DDS_WaitSet

  <<interface>> <<cert>> Allows an application to wait until one or more of the attached DDS_Condition objects has a `trigger_value` of DDS_BOOLEAN_TRUE or else until the timeout expires.

Functions

- DDS_Boolean DDS_Condition_get_trigger_value (DDS_Condition *self)

  <<cert>> Retrieve the `trigger_value`.
- DDS_Condition * DDS_GuardCondition_as_condition (DDS_GuardCondition *guard_cond)

  <<cert>> Access a DDS_GuardCondition's supertype instance.
- DDS_GuardCondition * DDS_GuardCondition_new (void)

  <<cert>> No argument constructor.
- DDS_ReturnCode_t DDS_GuardCondition_set_trigger_value (DDS_GuardCondition *self, DDS_Boolean value)

  <<cert>> Set the guard condition trigger value.
- DDS_Condition * DDS_StatusCondition_as_condition (DDS_StatusCondition *statusCondition)

  <<cert>> Access a DDS_StatusCondition's supertype instance.
- DDS_StatusMask DDS_StatusCondition_get_enabled_statuses (DDS_StatusCondition *self)

  <<cert>> Get the list of statuses enabled on an DDS_Entity.
- DDS_ReturnCode_t DDS_StatusCondition_set_enabled_statuses (DDS_StatusCondition *self, DDS_StatusMask mask)

  <<cert>> This operation defines the list of communication statuses that determine the `trigger_value` of the DDS_StatusCondition.
- DDS_Entity * DDS_StatusCondition_get_entity (DDS_StatusCondition *self)

  <<cert>> Get the DDS_Entity associated with the DDS_StatusCondition.
- DDS_WaitSet * DDS_WaitSet_new (void)

  <<cert>> Create a new, empty waitset.
- DDS_ReturnCode_t DDS_WaitSet_wait (DDS_WaitSet *self, struct DDS_ConditionSeq *active_conditions, const struct DDS_Duration_t *timeout)

  <<cert>> Allows an application thread to wait for the occurrence of certain conditions.
- DDS_ReturnCode_t DDS_WaitSet_attach_condition (DDS_WaitSet *self, DDS_Condition *cond)

  <<cert>> Attaches a DDS_Condition to the DDS_WaitSet.
- DDS_ReturnCode_t DDS_WaitSet_get_conditions (DDS_WaitSet *self, struct DDS_ConditionSeq *attached_conditions)

  <<cert>> Retrieves the list of attached DDS_Condition (s).

5.83.1 Detailed Description

DDS_Condition and DDS_WaitSet and related items.

5.83.2 Typedef Documentation

5.83.2.1 typedef struct DDS_ConditionImpl DDS_Condition

<<interface>> *<<cert>>* Root class for all the conditions that may be attached to a DDS_WaitSet.

This basic class is specialised in two classes:

DDS_GuardCondition and DDS_StatusCondition.

A DDS_Condition has a `trigger_value` that can be DDS_BOOLEAN_TRUE or DDS_BOOLEAN_FALSE and is set automatically by RTI Connext Micro.

See also

DDS_WaitSet

5.83.2.2 typedef struct DDS_GuardConditionImpl DDS_GuardCondition

<<interface>> *<<cert>>* A specific DDS_Condition whose `trigger_value` is completely under the control of the application.

The DDS_GuardCondition provides a way for an application to manually wake up a DDS_WaitSet. This is accomplished by attaching the DDS_GuardCondition to the DDS_WaitSet and then setting the `trigger_value` by means of the DDS_GuardCondition_set_trigger_value operation.

See also

DDS_WaitSet

5.83.2.3 typedef struct DDS_StatusConditionImpl DDS_StatusCondition

<<interface>> *<<cert>>* A specific DDS_Condition that is associated with each DDS_Entity.

The `trigger_value` of the DDS_StatusCondition depends on the communication status of that entity (e.g., arrival of data, loss of information, etc.), 'filtered' by the set of `enabled_statuses` on the DDS_StatusCondition.

This release supports the following subset of communication statuses that can be assocated with a DDS_StatusCondition :

DDS_DATA_AVAILABLE_STATUS

See also

Status Kinds
DDS_WaitSet, DDS_Condition
DDS_Listener

5.83.2.4 typedef struct DDS_WaitSetImpl DDS_WaitSet

<<interface>> *<<cert>>* Allows an application to wait until one or more of the attached DDS_Condition objects has a `trigger_value` of DDS_BOOLEAN_TRUE or else until the timeout expires.

5.83.3 Usage

DDS_Condition (s) (in conjunction with wait-sets) provide an alternative mechanism to allow the middleware to communicate communication status changes (including arrival of data) to the application.

This mechanism is wait-based. Its general use pattern is as follows:

- The application indicates which relevant information it wants to get by creating DDS_Condition objects (DDS_StatusCondition) and attaching them to a DDS_WaitSet.

- It then waits on that DDS_WaitSet until the `trigger_value` of one or several DDS_Condition objects become DDS_BOOLEAN_TRUE.

- It then uses the result of the wait (i.e., `active_conditions`, the list of DDS_Condition objects with `trigger_value` == DDS_BOOLEAN_TRUE) to actually get the information:
  - by calling DDS_Entity_get_status_changes and then `get_<communication_status>`() on the relevant DDS_Entity, if the condition is a DDS_StatusCondition and the status changes, refer to plain communication status;
  - by calling DDS_Entity_get_status_changes and then FooDataReader_read() or FooDataReader_take on the relevant DDS_DataReader, if the condition is a DDS_StatusCondition and the status changes refers to DDS_DATA_AVAILABLE_STATUS;

Usually the first step is done in an initialization phase, while the others are put in the application main loop.

As there is no extra information passed from the middleware to the application when a wait returns (only the list of triggered DDS_Condition objects), DDS_Condition objects are meant to embed all that is needed to react properly when enabled. In particular, DDS_Entity-related conditions are related to exactly one DDS_Entity and cannot be shared.

The result of a DDS_WaitSet_wait operation depends on the state of the DDS_WaitSet, which in turn depends on whether at least one attached DDS_Condition has a `trigger_value` of DDS_BOOLEAN_TRUE. If the wait operation is called on DDS_WaitSet with state BLOCKED, it will block the calling thread. If wait is called on a DDS_WaitSet with state UNBLOCKED, it will return immediately. In addition, when the DDS_WaitSet transitions from BLOCKED to UNBLOCKED it wakes up any threads that had called wait on it.

A key aspect of the DDS_Condition/DDS_WaitSet mechanism is the setting of the `trigger_value` of each DDS_Condition.

5.83.4 Trigger State of a ::DDS_StatusCondition

The `trigger_value` of a DDS_StatusCondition is the boolean OR of the ChangedStatusFlag of all the communication statuses (see Status Kinds) to which it is sensitive. That is, `trigger_value` == DDS_BOOLEAN_FALSE only if all the values of the ChangedStatusFlags are DDS_BOOLEAN_FALSE.

The sensitivity of the DDS_StatusCondition to a particular communication status is controlled by the list of `enabled_statuses` set on the condition by means of the DDS_StatusCondition_set_enabled_statuses operation.

5.83.5 Trigger State of a ::DDS_GuardCondition

The `trigger_value` of a DDS_GuardCondition is completely controlled by the application via the operation DDS_GuardCondition_set_trigger_value.

See also

Status Kinds
DDS_StatusCondition, DDS_GuardCondition
DDS_Listener

5.83.6 Function Documentation

5.83.6.1 DDS_Boolean DDS_Condition_get_trigger_value (DDS_Condition * *self*)

<<cert>> Retrieve the `trigger_value`.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

the trigger value.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.83.6.2 DDS_Condition* DDS_GuardCondition_as_condition (DDS_GuardCondition * *guard_cond*)

<<*cert*>> Access a DDS_GuardCondition's supertype instance.

Parameters

guard_cond	<<*in*>> Cannot be NULL.

5.83.6.3 DDS_GuardCondition* DDS_GuardCondition_new (void)

<<*cert*>> No argument constructor.

Construct a new guard condition on the heap.

Returns

A new condition with trigger value DDS_BOOLEAN_FALSE, or NULL if a condition could not be allocated.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.83.6.4 DDS_ReturnCode_t DDS_GuardCondition_set_trigger_value (DDS_GuardCondition * *self*, DDS_Boolean *value*)

<<*cert*>> Set the guard condition trigger value.

Parameters

self	<<*in*>> Cannot be NULL.
value	<<*in*>> the new trigger value.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.83.6.5 DDS_Condition* DDS_StatusCondition_as_condition (DDS_StatusCondition * *statusCondition*)

<<*cert*>> Access a DDS_StatusCondition's supertype instance.

Parameters

statusCondition	<<*in*>> Cannot be NULL.

5.83.6.6 DDS_StatusMask DDS_StatusCondition_get_enabled_statuses (DDS_StatusCondition * *self*)

<<*cert*>> Get the list of statuses enabled on an DDS_Entity.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

list of enabled statuses.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.83.6.7 DDS_ReturnCode_t DDS_StatusCondition_set_enabled_statuses (DDS_StatusCondition * *self*, DDS_StatusMask *mask*)

<<*cert*>> This operation defines the list of communication statuses that determine the `trigger_value` of the DDS_StatusCondition.

This operation may change the `trigger_value` of the DDS_StatusCondition.

DDS_WaitSet objects' behavior depends on the changes of the `trigger_value` of their attached conditions. Therefore, any DDS_WaitSet to which the DDS_StatusCondition is attached is potentially affected by this operation.

If this function is not invoked, the default list of enabled statuses includes all the statuses.

Parameters

self	<<*in*>> Cannot be NULL.
mask	<<*in*>> the list of enables statuses (see Status Kinds)

Returns

One of the Standard Return Codes

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.83.6.8 DDS_Entity* DDS_StatusCondition_get_entity (DDS_StatusCondition * *self*)

<<*cert*>> Get the DDS_Entity associated with the DDS_StatusCondition.

There is exactly one DDS_Entity associated with each DDS_StatusCondition.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

DDS_Entity associated with the DDS_StatusCondition.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.83.6.9 DDS_WaitSet* DDS_WaitSet_new (void)

<<cert>> Create a new, empty waitset.

Create a new waitset. The waitset does not have a factory, so for this reason the waitset is allocated from the heap. A waitset maintains a list of conditions that have been attached with DDS_WaitSet_attach.

Returns

A new, empty waitset on success or NULL on failure.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.83.6.10 DDS_ReturnCode_t DDS_WaitSet_wait (DDS_WaitSet * *self*, struct DDS_ConditionSeq * *active_conditions*, const struct DDS_Duration_t * *timeout*)

<<cert>> Allows an application thread to wait for the occurrence of certain conditions.

If none of the conditions attached to the DDS_WaitSet have a `trigger_value` of DDS_BOOLEAN_TRUE, the wait operation will block suspending the calling thread.

The result of the wait operation is the list of all the attached conditions that have a `trigger_value` of DDS_BOOLEAN_TRUE (i.e., the conditions that unblocked the wait).

Note: The resolution of the `timeout` period is constrained by the resolution of the system clock.

The wait operation takes a `timeout` argument that specifies the maximum duration for the wait. If this duration is exceeded and none of the attached DDS_Condition objects is DDS_BOOLEAN_TRUE, wait will return with the return code DDS_RETCODE_TIMEOUT. In this case, the resulting list of conditions will be empty.

Note: The resolution of the `timeout` period is constrained by the resolution of the system clock.

DDS_RETCODE_TIMEOUT will *not* be returned when the `timeout` duration is exceeded if attached DDS_Condition objects are DDS_BOOLEAN_TRUE, or in the case of a DDS_WaitSet waiting for more than one trigger event, if one or more trigger events have occurred.

It is not allowable for for more than one application thread to be waiting on the same DDS_WaitSet. If the wait operation is invoked on a DDS_WaitSet that already has a thread blocking on it, the operation will return immediately with the value DDS_RETCODE_PRECONDITION_NOT_MET.

Parameters

active_conditions	*<<inout>>* a valid non-NULL DDS_ConditionSeq object. Note that RTI Connext Micro will not allocate a new object if `active_conditions` is NULL; the function will return DDS_RETCODE_PRECONDITION_NOT_MET.
self	*<<in>>* Cannot be NULL.
timeout	*<<in>>* a wait timeout

Exceptions

One	of the Standard Return Codes or DDS_RETCODE_PRECONDITION_NOT_MET or DDS_RETCODE_TIMEOUT.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.83.6.11 DDS_ReturnCode_t DDS_WaitSet_attach_condition (DDS_WaitSet * *self*, DDS_Condition * *cond*)

<<cert>> Attaches a DDS_Condition to the DDS_WaitSet.

It is possible to attach a DDS_Condition on a DDS_WaitSet that is currently being waited upon (via the wait operation). In this case, if the DDS_Condition has a `trigger_value` of DDS_BOOLEAN_TRUE, then attaching the condition will unblock the DDS_WaitSet.

Parameters

self	*<<in>>* Cannot be NULL.
cond	*<<in>>* Contition to be attached.

Exceptions

One	of the Standard Return Codes, or DDS_RETCODE_OUT_OF_RESOURCES.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.83.6.12 DDS_ReturnCode_t DDS_WaitSet_get_conditions (DDS_WaitSet * *self*, struct DDS_ConditionSeq * *attached_conditions*)

<<cert>> Retrieves the list of attached DDS_Condition (s).

Parameters

self	*<<in>>* Cannot be NULL.
attached_conditions	*<<inout>>* a DDS_ConditionSeq object where the list of attached conditions will be returned

Exceptions

One	of the Standard Return Codes, or DDS_RETCODE_PRECONDITION_NOT_MET.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.84 Entity Support

DDS_Entity, DDS_Listener and related items.

Data Structures

- struct DDS_Listener

  <<interface>> Abstract base class for all Listener interfaces.

Macros

- #define DDS_Listener_INITIALIZER

  Initialize the DDS_Listener::listener_data pointer to NULL.

Typedefs

- typedef struct DDS_EntityImpl DDS_Entity

  <<interface>> Abstract base class for all the DDS objects that support QoS policies, and a listener.
- typedef struct DDS_DomainEntityImpl DDS_DomainEntity

  <<interface>> Abstract base class for all DDS entities except for the DDS_DomainParticipant.

Functions

- DDS_ReturnCode_t DDS_Entity_enable (DDS_Entity *self)

  <<cert>> Enables the DDS_Entity.
- DDS_InstanceHandle_t DDS_Entity_get_instance_handle (DDS_Entity *self)

  <<cert>> Allows access to the DDS_InstanceHandle_t associated with the DDS_Entity.
- DDS_StatusCondition * DDS_Entity_get_statuscondition (DDS_Entity *self)

  <<cert>> Return the DDS_StatusCondition associated with a particular DDS_Entity.
- DDS_StatusMask DDS_Entity_get_status_changes (DDS_Entity *self)

  <<cert>> Retrieves the list of communication statuses in the DDS_Entity that are triggered.

5.84.1 Detailed Description

DDS_Entity, DDS_Listener and related items. DDS_Entity subtypes are created and destroyed by factory objects. With the exception of DDS_DomainParticipant, whose factory is DDS_DomainParticipantFactory, all DDS_Entity factory objects are themselves DDS_Entity subtypes as well.

Important: all DDS_Entity delete operations are inherently thread-unsafe. The user must take extreme care that a given DDS_Entity is not destroyed in one thread while being used concurrently (including being deleted concurrently) in another thread. An operation's effect in the presence of the concurrent deletion of the operation's target DDS_Entity is undefined.

5.84.2 Macro Definition Documentation

5.84.2.1 #define DDS_Listener_INITIALIZER

Initialize the DDS_Listener::listener_data pointer to NULL.

5.84.3 Typedef Documentation

5.84.3.1 typedef struct DDS_EntityImpl DDS_Entity

<<interface>> Abstract base class for all the DDS objects that support QoS policies, and a listener.

All operations except for set_qos(), get_qos(), set_listener(), get_listener() and enable(), may return the value DDS_RETCODE_NOT_ENABLED.

QoS:

QoS Policies

Status:

Status Kinds

Listener:

DDS_Listener

5.84.4 Abstract operations

Each derived entity provides the following operations specific to its role in RTI Connext Micro.

5.84.4.1 set_qos (abstract)

This operation sets the QoS policies of the DDS_Entity.

This operation must be provided by each of the derived DDS_Entity classes (DDS_DomainParticipant, DDS_Topic, DDS_Publisher, DDS_DataWriter, DDS_Subscriber, and DDS_DataReader) so that the policies that are meaningful to each DDS_Entity can be set.

Precondition

Certain policies are immutable (see QoS Policies): they can only be set at DDS_Entity creation time or before the entity is enabled. If `set_qos()` is invoked after the DDS_Entity is enabled and it attempts to change the value of an immutable policy, the operation will fail and return DDS_RETCODE_IMMUTABLE_POLICY.
Certain values of QoS policies can be incompatible with the settings of the other policies. The `set_qos()` operation will also fail if it specifies a set of values that, once combined with the existing values, would result in an inconsistent set of policies. In this case, the operation will fail and return DDS_RETCODE_INCONSISTENT_POLICY.
If the application supplies a non-default value for a QoS policy that is not supported by the implementation of the service, the set_qos operation will fail and return DDS_RETCODE_UNSUPPORTED.

Postcondition

The existing set of policies is only changed if the `set_qos()` operation succeeds. This is indicated by a return code of DDS_RETCODE_OK. In all other cases, none of the policies are modified.

Each derived DDS_Entity class (DDS_DomainParticipant, DDS_Topic, DDS_Publisher, DDS_DataWriter, DDS_Subscriber, DDS_DataReader) has a corresponding special value of the QoS (DDS_PARTICIPANT_QOS_DEFAULT, DDS_PUBLISHER_QOS_DEFAULT, DDS_SUBSCRIBER_QOS_DEFAULT, DDS_TOPIC_QOS_DEFAULT, DDS_DATAWRITER_QOS_DEFAULT, DDS_DATAREADER_QOS_DEFAULT). This special value may be used as a parameter to the `set_qos` operation to indicate that the QoS of the DDS_Entity should be changed to match the current default QoS set in the DDS_Entity's factory. The operation set_qos cannot modify the immutable QoS, so a successful return of the operation indicates that the mutable QoS for the Entity has been modified to match the current default for the DDS_Entity's factory.

The set of policies specified in the `qos` parameter are applied on top of the existing QoS, replacing the values of any policies previously set.

Possible error codes returned in addition to Standard Return Codes : DDS_RETCODE_IMMUTABLE_POLICY, or DDS_RETCODE_INCONSISTENT_POLICY.

5.84.4.2 get_qos (abstract)

This operation allows access to the existing set of QoS policies for the DDS_Entity. This operation must be provided by each of the derived DDS_Entity classes (DDS_DomainParticipant, DDS_Topic, DDS_Publisher, DDS_DataWriter, DDS_Subscriber, and DDS_DataReader), so that the policies that are meaningful to each DDS_Entity can be retrieved.

Possible error codes are Standard Return Codes.

5.84.4.3 set_listener (abstract)

This operation installs a DDS_Listener on the DDS_Entity. The listener will only be invoked on the changes of communication status indicated by the specified `mask`.

This operation must be provided by each of the derived DDS_Entity classes (DDS_DomainParticipant, DDS_Topic, DDS_Publisher, DDS_DataWriter, DDS_Subscriber, and DDS_DataReader), so that the listener is of the concrete type suitable to the particular DDS_Entity.

It is permitted to use NULL as the value of the listener. The NULL listener behaves as if the `mask` is DDS_STATUS_MASK_NONE.

Postcondition

Only one listener can be attached to each DDS_Entity. If a listener was already set, the operation `set_listener()` will replace it with the new one. Consequently, if the value NULL is passed for the listener parameter to the set_listener operation, any existing listener will be removed.

5.84.4.4 get_listener (abstract)

This operation allows access to the existing DDS_Listener attached to the DDS_Entity.

This operation must be provided by each of the derived DDS_Entity classes (DDS_DomainParticipant, DDS_Topic, DDS_Publisher, DDS_DataWriter, DDS_Subscriber, and DDS_DataReader) so that the listener is of the concrete type suitable to the particular DDS_Entity.

If no listener is installed on the DDS_Entity, this operation will return NULL.

5.84.4.5 typedef struct DDS_DomainEntityImpl DDS_DomainEntity

<<interface>> Abstract base class for all DDS entities except for the DDS_DomainParticipant.

Its sole purpose is to *conceptually* express that DDS_DomainParticipant is a special kind of DDS_Entity that acts as a container of all other DDS_Entity but itself cannot contain other DDS_DomainParticipant.

5.84.5 Function Documentation

5.84.5.1 DDS_ReturnCode_t DDS_Entity_enable (DDS_Entity * *self*)

<<cert>> Enables the DDS_Entity.

This operation enables the Entity. Entity objects can be created either enabled or disabled. This is controlled by the value of the ENTITY_FACTORY QoS policy on the corresponding factory for the DDS_Entity.

By default, ENTITY_FACTORY is set so that it is not necessary to explicitly call DDS_Entity_enable on newly created entities.

The DDS_Entity_enable operation is idempotent. Calling enable on an already enabled Entity returns OK and has no effect.

If a DDS_Entity has not yet been enabled, the following kinds of operations may be invoked on it:

- set or get the QoS policies (including default QoS policies) and listener
- 'factory' operations

- 'lookup' operations

Other operations may explicitly state that they may be called on disabled entities; those that do not will return the error DDS_RETCODE_NOT_ENABLED.

It is legal to delete an DDS_Entity that has not been enabled by calling the proper operation on its factory.

Entities created from a factory that is disabled are created disabled, regardless of the setting of the DDS_EntityFactoryQosPolicy.

Calling enable on an Entity whose factory is not enabled will fail and return DDS_RETCODE_PRECONDITION_NOT_MET.

If DDS_EntityFactoryQosPolicy::autoenable_created_entities is TRUE, the enable operation on a factory will automatically enable all entities created from that factory.

Listeners associated with an entity are not called until the entity is enabled.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

One of the Standard Return Codes, Standard Return Codes or DDS_RETCODE_PRECONDITION_NOT_MET.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.84.5.2 DDS_InstanceHandle_t DDS_Entity_get_instance_handle (DDS_Entity * *self*)

<<cert>> Allows access to the DDS_InstanceHandle_t associated with the DDS_Entity.

This operation returns the DDS_InstanceHandle_t that represents the DDS_Entity.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

the instance handle associated with this entity.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.84.5.3 DDS_StatusCondition* DDS_Entity_get_statuscondition (DDS_Entity * *self*)

<<cert>> Return the DDS_StatusCondition associated with a particular DDS_Entity.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

The DDS_StatusCondition that belongs to the specified DDS_Entity.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

5.84.5.4 DDS_StatusMask DDS_Entity_get_status_changes (DDS_Entity * *self*)

<<*cert*>> Retrieves the list of communication statuses in the DDS_Entity that are triggered.

That is, the list of statuses whose value has changed since the last time the application read the status using the get_*_status() function.

When the entity is first created or if the entity is not enabled, all communication statuses are in the "untriggered" state so the list returned by the get_status_changes operation will be empty.

The list of statuses returned by the get_status_changes operation refers to the status that are triggered on the Entity itself and does not include statuses that apply to contained entities.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

list of communication statuses in the DDS_Entity that are triggered.

MT Safety:

This operation is thread safe. However, note that the arguments are not protected from being modified by other threads and must not be modified until the call returns.

See also

Status Kinds

5.85 ENTITY_NAME

<<*eXtension*>> Name of the entity.

Data Structures

- struct DDS_EntityNameQosPolicy
  <<*cert*>> *Policy used to name an entity.*

Macros

- #define DDS_ENTITYNAME_QOS_NAME_MAX 255
  Constant determining the maximum length of the name string contained in a DDS_EntityNameQosPolicy instance.

Functions

- DDS_Boolean DDS_EntityNameQosPolicy_set_name (struct DDS_EntityNameQosPolicy *const self, const char *const name)
  Set the EntityName policy.

Variables

- const char *const DDS_ENTITYNAME_QOS_POLICY_NAME
  Stringified human-readable name for DDS_EntityNameQosPolicy.
- const char *const DDS_AUTO_NAME_ENTITY
  Constant used to request that an entity be assigned an automatically generated name.

5.85.1 Detailed Description

<<*eXtension*>> Name of the entity.

5.85.2 Macro Definition Documentation

5.85.2.1 #define DDS_ENTITYNAME_QOS_NAME_MAX 255

Constant determining the maximum length of the name string contained in a DDS_EntityNameQosPolicy instance.

5.85.3 Function Documentation

5.85.3.1 DDS_Boolean DDS_EntityNameQosPolicy_set_name (struct DDS_EntityNameQosPolicy *const *self*, const char *const *name*)

Set the EntityName policy.

Set the entity name Qos policy. The maximum allowed string length is DDS_ENTITYNAME_QOS_NAME_MAX (excluding the NUL character).

Parameters

self	<<*in*>> Cannot be NULL.
name	<<*in*>> The entity name to assign to the policy

Returns

DDS_BOOLEAN_TRUE on success, DDS_BOOLEAN_FALSE on failure

5.85.4 Variable Documentation

5.85.4.1 const char* const DDS_ENTITYNAME_QOS_POLICY_NAME

Stringified human-readable name for DDS_EntityNameQosPolicy.

5.85.4.2 const char* const DDS_AUTO_NAME_ENTITY

Constant used to request that an entity be assigned an automatically generated name.

5.86 DDS_C

DDS C. ModuleID = 7.

Macros

- #define DDSC_LOG_INVALID_DURATION_EC (DDSC_LOG_BASE + 1)

  The specified duration is not valid.
- #define DDSC_LOG_INVALID_PARTICIPANT_NAME_EC (DDSC_LOG_BASE + 2)

  An invalid participant name was specified with an unknown GUID.
- #define DDSC_LOG_INVALID_PARTICIPANT_GUID_PREFIX_EC (DDSC_LOG_BASE + 3)

  An invalid participant GUID prefix was specified, typically the GUID prefix does not match an already detected participant.
- #define DDSC_LOG_SYS_GETTIME_EC (DDSC_LOG_BASE + 4)

  A call to OSAPI_System_get_time failed.
- #define DDSC_LOG_GET_NEXT_OBJECT_ID_EC (DDSC_LOG_BASE + 5)

  Failed to get the next automatically generated object ID for an entity's GUID. This typically means the object id pool has been exhausted.
- #define DDSC_LOG_SET_ENTITY_NAME_EC (DDSC_LOG_BASE + 6)

  Failed to set name string for a DDS entity in the DomainParticipantQos entity_name policy.
- #define DDSC_LOG_SYS_GET_HOSTNAME_EC (DDSC_LOG_BASE + 7)

  A call to OSAPI_System_get_hostname failed.
- #define DDSC_LOG_IO_SNPRINTF_FAILED_EC (DDSC_LOG_BASE + 8)

  A call to OSAPI_Stdio_snprintf failed. Typically this means the destination buffer was too small.
- #define DDSC_LOG_TOPIC_FIND_EC (DDSC_LOG_BASE + 9)

  Failed to find a topic created by a DomainParticipant.
- #define DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_NAME_EC (DDSC_LOG_BASE + 10)

  Endpoint discovery failed because the name of the remote participant parent for an endpoint was not found.
- #define DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_KEY_EC (DDSC_LOG_BASE + 11)

  Endpoint discovery failed because the key of the remote participant parent for an endpoint was not found. Note that the key is logged as 4 integers in host endianess format.
- #define DDSC_LOG_REMOTE_PARTICIPANT_KEY_NOT_EQUAL_EC (DDSC_LOG_BASE + 12)

  Failed endpoint discovery when key does not match the remote participant.
- #define DDSC_LOG_INVALID_ENDPOINT_GUID_EC (DDSC_LOG_BASE + 13)

  Failed endpoint discovery due to an invalid or unknown endpoint GUID.
- #define DDSC_LOG_ENDPOINT_NOT_CHILD_OF_PARTICIPANT_EC (DDSC_LOG_BASE + 14)

  Failed endpoint discovery when an endpoint is determined to belong to a different remote participant.
- #define DDSC_LOG_PARTICIPANT_DOES_NOT_EXIST_EC (DDSC_LOG_BASE + 15)

  Failed participant discovery because a remote participant that should have been already asserted locally was not found.
- #define DDSC_LOG_REFRESH_REM_PARTICIPANT_EC (DDSC_LOG_BASE + 16)

  Did not find a remote participant when asserting participant liveliness for it. Note that the key is logged as 4 integers in host endianess format.
- #define DDSC_LOG_REFRESH_REM_PARTICIPANT_TIMEOUT_EC (DDSC_LOG_BASE + 17)

  Failed to assert participant liveliness to a remote participant.
- #define DDSC_LOG_PARTICIPANT_LOOKUP_EC (DDSC_LOG_BASE + 18)

  Failed to find a remote participant as previously discovered.
- #define DDSC_LOG_REMOVE_PUBLICATION_EC (DDSC_LOG_BASE + 19)

  Failed to remove resources for a remote publication.
- #define DDSC_LOG_REMOVE_SUBSCRIPTION_EC (DDSC_LOG_BASE + 20)

  Failed to remove resources for a remote subscription.
- #define DDSC_LOG_FIND_PUBLICATION_PARENT_EC (DDSC_LOG_BASE + 21)

Cannot determine the participant of a remote publication.

- #define DDSC_LOG_FIND_SUBSCRIPTION_PARENT_EC (DDSC_LOG_BASE + 22)

  Cannot determine the participant of a remote subscription.
- #define DDSC_LOG_MAX_PARTICIPANT_ID_REACHED_EC (DDSC_LOG_BASE + 23)

  Failed to create DomainParticipant due to running out of participant IDs. This is typically caused by all UDP ports being used.
- #define DDSC_LOG_RESERVE_LOCATORS_EC (DDSC_LOG_BASE + 24)

  Failed to reserve endpoint locators.
- #define DDSC_LOG_TIMER_CREATE_TIMEOUT_EC (DDSC_LOG_BASE + 25)

  Failed to create a timeout.
- #define DDSC_LOG_ILLEGAL_OBJECTID_EC (DDSC_LOG_BASE + 26)

  Illegal object id specified.
- #define DDSC_LOG_TOPIC_NARROW_EC (DDSC_LOG_BASE + 27)

  Failed to narrow a TopicDescription to the named Topic.
- #define DDSC_LOG_FAILED_UPDATE_STATUS_CONDITION_EC (DDSC_LOG_BASE + 28)

  Failed to update state of StatusCondition.
- #define DDSC_LOG_WS_REMOVE_COND_REFERENCE_EC (DDSC_LOG_BASE + 29)

  Failed to remove a condition reference from a waitset.
- #define DDSC_LOG_WS_ADD_COND_REFERENCE_EC (DDSC_LOG_BASE + 30)

  Failed to add a condition reference to a waitset.
- #define DDSC_LOG_RELEASE_META_MC_EC (DDSC_LOG_BASE + 31)

  Failed to release resources for multicast discovery locators.
- #define DDSC_LOG_RELEASE_META_UC_EC (DDSC_LOG_BASE + 32)

  Failed to release resources for unicast discovery locators.
- #define DDSC_LOG_RELEASE_USER_MC_EC (DDSC_LOG_BASE + 33)

  Failed to release resources for multicast user locators.
- #define DDSC_LOG_RELEASE_USER_UC_EC (DDSC_LOG_BASE + 34)

  Failed to release resources for unicast user locators.
- #define DDSC_LOG_INVALID_DOMAINID_EC (DDSC_LOG_BASE + 35)

  The domain ID specified exceeds what is allowed based on the parameters specified in dp_qos.protocol.rtps_well_known_ports.
- #define DDSC_LOG_RESOLVE_LOCATORS_PRIORITY_EC (DDSC_LOG_BASE + 36)

  Failed to resolve locators based on priority.
- #define DDSC_LOG_DATABASE_CREATE_EC (DDSC_LOG_BASE + 100)

  Failed to create database.
- #define DDSC_LOG_DATABASE_DELETE_EC (DDSC_LOG_BASE + 101)

  Failed to delete database.
- #define DDSC_LOG_TABLE_CREATE_EC (DDSC_LOG_BASE + 102)

  Failed to create database table of the specified name.
- #define DDSC_LOG_TABLE_INUSE_EC (DDSC_LOG_BASE + 103)

  Failed to delete a database table because it is not empty.
- #define DDSC_LOG_TABLE_DELETE_EC (DDSC_LOG_BASE + 104)

  Failed to delete a database table.
- #define DDSC_LOG_TABLE_SELECT_EC (DDSC_LOG_BASE + 105)

  A selection operation failed on the specified database table.
- #define DDSC_LOG_CREATE_INDEX_EC (DDSC_LOG_BASE + 106)

  Failed to create an index on a database table.
- #define DDSC_LOG_DELETE_INDEX_EC (DDSC_LOG_BASE + 107)

  Failed to delete an index on a database table.
- #define DDSC_LOG_DB_CURSOR_INVALIDATED_EC (DDSC_LOG_BASE + 108)

  A database table cursor was invalidated while in use.

- #define DDSC_LOG_SUBSCRIPTION_RECORD 1

  Database record for a Subscription.
- #define DDSC_LOG_PUBLICATION_RECORD 2

  Database record for a Publication.
- #define DDSC_LOG_PARTICIPANT_RECORD 3

  Database record for a Remote Participant.
- #define DDSC_LOG_TOPIC_RECORD 4

  Database record for a Topic.
- #define DDSC_LOG_DATAWRITER_RECORD 5

  Database record for a DataWriter.
- #define DDSC_LOG_DATAREADER_RECORD 6

  Database record for a DataReader.
- #define DDSC_LOG_PUBLISHER_RECORD 7

  Database record for a Publisher.
- #define DDSC_LOG_SUBSCRIBER_RECORD 8

  Database record for a Subscriber.
- #define DDSC_LOG_ROUTE_RECORD 9

  Database record for a route record.
- #define DDSC_LOG_BIND_RECORD 10

  Database record for a bind record.
- #define DDSC_LOG_TYPE_RECORD 11

  Database record for a type plugin.
- #define DDSC_LOG_STRING_RECORD 14

  String record.
- #define DDSC_LOG_RECORD_CREATE_EC (DDSC_LOG_BASE + 109)

  Failed to create a database record of the specified kind.
- #define DDSC_LOG_RECORD_DELETE_EC (DDSC_LOG_BASE + 110)

  Failed to delete a database record of the specified kind.
- #define DDSC_LOG_RECORD_INSERT_EC (DDSC_LOG_BASE + 111)

  Failed to insert a database record of the specified kind.
- #define DDSC_LOG_RECORD_ERROR_EC (DDSC_LOG_BASE + 112)

  Unknown error for database record of the specified kind.
- #define DDSC_LOG_RECORD_EXISTS_EC (DDSC_LOG_BASE + 113)

  A database record of the specified kind already exists.
- #define DDSC_LOG_RECORD_LOOKUP_EC (DDSC_LOG_BASE + 114)

  A lookup of a database record of the specified kind failed.
- #define DDSC_LOG_RECORD_NOT_EXISTS_EC (DDSC_LOG_BASE + 115)

  A database record of the specified kind does not exist.
- #define DDSC_LOG_RECORD_SELECT_EC (DDSC_LOG_BASE + 116)

  A database select on the specified record kind failed.
- #define DDSC_LOG_RECORD_REMOVE_EC (DDSC_LOG_BASE + 117)

  Removal a database record of the specified kind failed.
- #define DDSC_LOG_RECORD_INITIALIZE_EC (DDSC_LOG_BASE + 118)

  A database record of the specified kind could not be initialized.
- #define DDSC_LOG_RECORD_FINALIZE_EC (DDSC_LOG_BASE + 119)

  A database record of the specified kind could not be finalized.
- #define DDSC_LOG_PUBLICATIONDATA_OBJECT 1

  DDS PublicationBuiltinTopicData object.
- #define DDSC_LOG_SUBSCRIPTIONDATA_OBJECT 2

  DDS SubscriptionBuiltinTopicData object.
- #define DDSC_LOG_PARTICIPANTDATA_OBJECT 3

DDS ParticipantBuiltinTopicData object.

- #define DDSC_LOG_DATAREADERQOS_OBJECT 4

  DDS DataReaderQos object.
- #define DDSC_LOG_DATAWRITERQOS_OBJECT 5

  DDS DataWriterQos object.
- #define DDSC_LOG_TOPICQOS_OBJECT 6

  DDS TopicQos object.
- #define DDSC_LOG_PUBLISHERQOS_OBJECT 7

  DDS PublisherQos object.
- #define DDSC_LOG_SUBSCRIBERQOS_OBJECT 8

  DDS SubscriberQos object.
- #define DDSC_LOG_PARTICIPANTQOS_OBJECT 9

  DDS DomainParticipantQos object.
- #define DDSC_LOG_ENTITY_OBJECT 10

  DDS Entity object.
- #define DDSC_LOG_PARTICIPANTFACTORYQOS_OBJECT 11

  DDS DomainParticipantFactoryQos object.
- #define DDSC_LOG_TYPE_OBJECT 12

  A DDS Type object.
- #define DDSC_LOG_BINDRESOLVER_OBJECT 13

  A NETIO BindResolver object.
- #define DDSC_LOG_ROUTERESOLVER_OBJECT 14

  A NETIO RouteResolver object.
- #define DDSC_LOG_ADDRESSRESOLVER_OBJECT 15

  A NETIO AddressResolver object.
- #define DDSC_LOG_DATAWRITERIO_OBJECT 16

  A NETIO DataWriterInterface object.
- #define DDSC_LOG_DATAREADERIO_OBJECT 17

  A NETIO DataReaderInterface object.
- #define DDSC_LOG_LOG_OBJECT 18

  A OSAPI Log object.
- #define DDSC_LOG_SYSTEM_OBJECT 19

  A OSAPI system object.
- #define DDSC_LOG_MUTEX_OBJECT 20

  A OSAPI mutex object.
- #define DDSC_LOG_CONDREF_OBJECT 21

  A DDS Waitset condition reference object.
- #define DDSC_LOG_WSREF_OBJECT 22

  A DDS Condition waitset reference object.
- #define DDSC_LOG_MD5STREAM_OBJECT 23

  A MD5 stream object.
- #define DDSC_LOG_CONDITION_OBJECT 24

  A DDS Condition object.
- #define DDSC_LOG_RT_OBJECT 25

  A RT Condition object.
- #define DDSC_LOG_PARTICIPANT_POOL_OBJECT 26

  A Participant pool object.
- #define DDSC_LOG_TIMER_OBJECT 27

  A OSAPI_Timer object.
- #define DDSC_LOG_TIMEROUT_OBJECT 28

  A OSAPI_Timer timeout object.

- #define DDSC_LOG_TOPIC_OBJECT 29

  A DDS Topic object.
- #define DDSC_LOG_WAITSET_OBJECT 30

  A DDS WaitSet object.
- #define DDSC_LOG_UUID_OBJECT 31

  A UUID object.
- #define DDSC_LOG_STRING_OBJECT 40

  A String object.
- #define DDSC_LOG_STRINGMANAGER_OBJECT 43

  A String Manager object.
- #define DDSC_LOG_OBJECT_INITIALIZE_EC (DDSC_LOG_BASE + 200)

  Out of resources to initialize object of the specified kind.
- #define DDSC_LOG_OBJECT_ALLOCATE_EC (DDSC_LOG_BASE + 201)

  Out of resources to allocate an object of the specified kind.
- #define DDSC_LOG_OBJECT_FINALIZE_EC (DDSC_LOG_BASE + 202)

  Failed to finalize object of specified kind.
- #define DDSC_LOG_OBJECT_DELETE_EC (DDSC_LOG_BASE + 203)

  Failed to delete object of specified kind.
- #define DDSC_LOG_OBJECT_COPY_EC (DDSC_LOG_BASE + 204)

  Failed to copy object of specified kind.
- #define DDSC_LOG_OBJECT_REFCOUNT_EC (DDSC_LOG_BASE + 205)

  Failed to delete/finalize an object because other objects are referencing it.
- #define DDSC_LOG_OBJECT_GET_PROPERTY_EC (DDSC_LOG_BASE + 206)

  Failed to get the object properties.
- #define DDSC_LOG_OBJECT_SET_PROPERTY_EC (DDSC_LOG_BASE + 207)

  Failed to set the object properties.
- #define DDSC_LOG_OBJECT_EMPTY_EC (DDSC_LOG_BASE + 208)

  An object is empty, typically applies only to buffer-pool objects.
- #define DDSC_LOG_OBJECT_INUSECOUNT_EC (DDSC_LOG_BASE + 209)

  Failed to delete/finalize an object because other objects are using it.
- #define DDSC_LOG_OBJECT_CREATE_EC (DDSC_LOG_BASE + 210)

  Failed to create an object.
- #define DDSC_LOG_NETMASK_SEQUENCE 1

  A NETIO Netmask sequence.
- #define DDSC_LOG_ROUTE_SEQUENCE 2

  A NETIO Route sequence.
- #define DDSC_LOG_RESERVED_SEQUENCE 3

  A NETIO Reserved address sequence.
- #define DDSC_LOG_ENABLED_USER_TRANSPORT_SEQUENCE 4

  A DDS sequence of enabled user NETIO transports.
- #define DDSC_LOG_ENABLED_TRANSPORT_SEQUENCE 5

  A DDS sequence of enabled transports.
- #define DDSC_LOG_ENABLED_DISCVOERY_TRANSPORT_SEQUENCE 6

  A DDS sequence of enabled discovery transports.
- #define DDSC_LOG_INITIAL_PEER_SEQUENCE 7

  A DDS sequence of initial peers.
- #define DDSC_LOG_DESTINATION_SEQUENCE 8

  A NETIO sequence of destinations to send to.
- #define DDSC_LOG_METAUNICAST_SEQUENCE 9

  A DDS sequence of meta-traffic unicast locators.
- #define DDSC_LOG_METAMULTICAST_SEQUENCE 10

A DDS sequence of meta-traffic multicast locators.

- #define DDSC_LOG_USERUNICAST_SEQUENCE 11

  A DDS sequence of user-traffic unicast locators.
- #define DDSC_LOG_USERMULTICAST_SEQUENCE 12

  A DDS sequence of user-traffic multicast locators.
- #define DDSC_LOG_READTAKE_SEQUENCE 13

  A DDS sequence used in a read/take call.
- #define DDSC_LOG_SEQUENCE_SETMAX_EC (DDSC_LOG_BASE + 300)

  Failed to set the maximum length of a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_SETLENGTH_EC (DDSC_LOG_BASE + 301)

  Failed to set the length of a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_GETREF_EC (DDSC_LOG_BASE + 302)

  Failed to get a reference at the specified index for a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_INITIALIZE_EC (DDSC_LOG_BASE + 303)

  Failed to initialize a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_FINALIZE_EC (DDSC_LOG_BASE + 304)

  Failed to finalize a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_COPY_EC (DDSC_LOG_BASE + 305)

  Failed to copy a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_INVALID_EC (DDSC_LOG_BASE + 306)

  The sequence of the specified kind was invalid in the context it is used.
- #define DDSC_LOG_DISCOVERY_COMPONENT 1

  A component of discovery kind.
- #define DDSC_LOG_RTPS_COMPONENT 2

  A component of RTPS kind.
- #define DDSC_LOG_READERHISTORY_COMPONENT 3

  A component of reader-history kind.
- #define DDSC_LOG_WRITERHISTORY_COMPONENT 4

  A component of writer-history kind.
- #define DDSC_LOG_DATAREADERIO_COMPONENT 5

  A component of DataReaderInterface kind.
- #define DDSC_LOG_DATAWRITERIO_COMPONENT 6

  A component of DataWriterInterface kind.
- #define DDSC_LOG_NETIO_COMPONENT 7

  A component of NETIO kind.
- #define DDSC_LOG_TRANSPORT_COMPONENT 8

  A component of transport kind.
- #define DDSC_LOG_COMPONENT_LOOKUP_EC (DDSC_LOG_BASE + 400)

  Did not find a component factory with the given name in the registry.
- #define DDSC_LOG_COMPONENT_CREATE_EC (DDSC_LOG_BASE + 401)

  Could not create a component of the specified kind using the specified factory.
- #define DDSC_LOG_COMPONENT_DELETE_EC (DDSC_LOG_BASE + 402)

  Could not delete a component of the specified kind using the specified factory.
- #define DDSC_LOG_HISTORY_RESOURCE 1

  Exceeded resource limits for writer history queue.
- #define DDSC_LOG_SAMPLE_RESOURCES 2

  Failed to get resource for new sample due to resource limit.
- #define DDSC_LOG_PARTICIPANT_RESOURCES 3

  Failed to allocate a new participant.
- #define DDSC_LOG_RESOURCE_EXCEEDED_EC (DDSC_LOG_BASE + 500)

  Could not allocate a resource of the specified kind.

- #define DDSC_LOG_TOPIC_QOS 1

  The TopicQos kind.
- #define DDSC_LOG_DATAREADER_QOS 2

  The DataReaderQos kind.
- #define DDSC_LOG_DATAWRITER_QOS 3

  The DataWriterQos kind.
- #define DDSC_LOG_SUBSCRIBER_QOS 4

  The SubscriberQos kind.
- #define DDSC_LOG_PUBLISHER_QOS 5

  The PublisherQos kind.
- #define DDSC_LOG_PARTICIPANT_QOS 6

  The DomainParticipantQos kind.
- #define DDSC_LOG_PARTICIPANTFACTORY_QOS 7

  The DomainParticipantFactoryQos kind.
- #define DDSC_LOG_DEFAULTTOPIC_QOS 8

  The default TopicQos kind.
- #define DDSC_LOG_DEFAULTDATAREADER_QOS 9

  The default DataReaderQos kind.
- #define DDSC_LOG_DEFAULTDATAWRITER_QOS 10

  The default DataWriterQos kind.
- #define DDSC_LOG_DEFAULTSUBSCRIBER_QOS 11

  The default SubscriberQos kind.
- #define DDSC_LOG_DEFAULTPUBLISHER_QOS 12

  The default PublisherQos kind.
- #define DDSC_LOG_DEFAULTPARTICIPANT_QOS 13

  The default PublisherQos kind.
- #define DDSC_LOG_DEADLINE_QOS_POLICY 14

  The deadline qos policy kind.
- #define DDSC_LOG_LIVELINESS_QOS_POLICY 15

  The liveliness qos policy kind.
- #define DDSC_LOG_HISTORY_QOS_POLICY 16

  The history qos policy kind.
- #define DDSC_LOG_RESOURCE_LIMIT_QOS_POLICY 17

  The resource limits qos policy kind.
- #define DDSC_LOG_PROTOCOL_QOS_POLICY 18

  The protocol qos policy kind.
- #define DDSC_LOG_TYPE_SUPPORT_QOS_POLICY 19

  The type-support qos policy kind.
- #define DDSC_LOG_RELIABILITY_QOS_POLICY 20

  The reliability qos policy kind.
- #define DDSC_LOG_DURABILITY_QOS_POLICY 21

  The durability qos policy kind.
- #define DDSC_LOG_OWNERSHIP_QOS_POLICY 22

  The ownership qos policy kind.
- #define DDSC_LOG_OWNERSHIP_STRENGTH_QOS_POLICY 23

  The ownership-strength qos policy kind.
- #define DDSC_LOG_TRANSPORT_QOS_POLICY 24

  The transport qos policy kind.
- #define DDSC_LOG_PARTICIPANT_ID_QOS_POLICY 25

  The participant id qos policy kind.
- #define DDSC_LOG_HEARTBEATS_QOS_POLICY 26

The heartbeat qos policy kind.

- #define DDSC_LOG_DATAWRITER_RESOURCE_QOS_POLICY 27

  The DataWriterQos writer_resource_limits policy.
- #define DDSC_LOG_DATAREADER_RESOURCE_QOS_POLICY 28

  The DataReaderQos reader_resource_limits policy.
- #define DDSC_LOG_DESTINATION_ORDER_POLICY 29

  The DestinationOrderQos destination_order policy.
- #define DDSC_LOG_USER_TRAFFIC_QOS_POLICY 30

  The UserTraffic policy.
- #define DDSC_LOG_DISCOVERY_TRAFFIC_QOS_POLICY 31

  The Discovery.enabled_transport policy.
- #define DDSC_LOG_QOS_INCONSISTENT_POLICY_EC (DDSC_LOG_BASE + 501)

  An inconsistent Qos policy for the specified Qos kind was found.
- #define DDSC_LOG_QOS_INCONSISTENT_POLICIES_EC (DDSC_LOG_BASE + 502)

  Inconsistency between two Qos policies for the specified Qos kind was found.
- #define DDSC_LOG_QOS_INCONSISTENT_EC (DDSC_LOG_BASE + 503)

  Failed to create an entity or set a qos due to inconsistent policy.
- #define DDSC_LOG_QOS_COPY_EC (DDSC_LOG_BASE + 504)

  Failed to copy a Qos of the specified kind.
- #define DDSC_LOG_QOS_INITIALIZE_EC (DDSC_LOG_BASE + 505)

  Failed to initialize a Qos of the specified kind.
- #define DDSC_LOG_QOS_FINALIZE_EC (DDSC_LOG_BASE + 506)

  Failed to finalize a Qos of the specified kind.
- #define DDSC_LOG_QOS_SET_EC (DDSC_LOG_BASE + 507)

  Failed to set a Qos of the specified kind.
- #define DDSC_LOG_QOS_SET_ON_ENABLED_EC (DDSC_LOG_BASE + 508)

  Failed to set a Qos of the specified kind because the entity is already enabled.
- #define DDSC_LOG_QOS_IMMUTABLE_EC (DDSC_LOG_BASE + 509)

  Failed to set a Qos of the specified kind the immutable Qos policies have been changed.
- #define DDSC_LOG_QOS_CHANGED_EC (DDSC_LOG_BASE + 510)

  A discovered Qos changed (the entity already existed)
- #define DDSC_LOG_QOS_GET_EC (DDSC_LOG_BASE + 511)

  Failed to get a Qos of the specified kind.
- #define DDSC_LOG_TOPIC_LISTENER 1

  The topic listener kind.
- #define DDSC_LOG_DATAREADER_LISTENER 2

  The datareader listener kind.
- #define DDSC_LOG_DATAWRITER_LISTENER 3

  The datawriter listener kind.
- #define DDSC_LOG_SUBSCRIBER_LISTENER 4

  The subscriber listener kind.
- #define DDSC_LOG_PUBLISHER_LISTENER 5

  The publisher listener kind.
- #define DDSC_LOG_PARTICIPANT_LISTENER 6

  The participant listener kind.
- #define DDSC_LOG_PARTICIPANTFACTORY_LISTENER 7

  The participant-factory listener kind.
- #define DDSC_LOG_LISTENER_INCONSISTENT_EC (DDSC_LOG_BASE + 512)

  Failed to create an entity due to inconsistent listener and status mask.
- #define DDSC_LOG_LISTENER_SET_EC (DDSC_LOG_BASE + 513)

  Failed to set the listener of the specified kind.

- #define DDSC_LOG_LISTENER_GET_EC (DDSC_LOG_BASE + 514)

  Failed to get the listener of the specified kind.
- #define DDSC_LOG_LISTENER_SET_ILLEGAL_NULL_EC (DDSC_LOG_BASE + 515)

  Illegal combination of NULL listener and non-NONE status mask when setting a listener for an Entity.
- #define DDSC_LOG_TOPIC_ENTITY 1

  The topic entity kind.
- #define DDSC_LOG_DATAWRITER_ENTITY 2

  The datawriter entity kind.
- #define DDSC_LOG_DATAREADER_ENTITY 3

  The datareader entity kind.
- #define DDSC_LOG_PUBLISHER_ENTITY 4

  The publisher entity kind.
- #define DDSC_LOG_SUBSCRIBER_ENTITY 5

  The subscriber entity kind.
- #define DDSC_LOG_PARTICIPANT_ENTITY 6

  The participant entity kind.
- #define DDSC_LOG_PUBLICATION_ENTITY 7

  The publication entity kind.
- #define DDSC_LOG_SUBSCRIPTION_ENTITY 8

  The subscription entity kind.
- #define DDSC_LOG_ENTITY_ENABLE_EC (DDSC_LOG_BASE + 600)

  Failed to enable an entity of the specified kind.
- #define DDSC_LOG_ENTITY_NOT_EMPTY_EC (DDSC_LOG_BASE + 601)

  Failed to an delete/finalize an entity of the specified kind because it is not empty.
- #define DDSC_LOG_ENTITY_FINALIZE_EC (DDSC_LOG_BASE + 602)

  Failed to finalize an entity of the specified kind.
- #define DDSC_LOG_ENTITY_INITIALIZE_EC (DDSC_LOG_BASE + 603)

  Failed to initialize an entity of the specified kind.
- #define DDSC_LOG_ENTITY_NOT_ENABLED_EC (DDSC_LOG_BASE + 604)

  An operation was attempted on an entity that is not enabled.
- #define DDSC_LOG_ENTITY_DIFFERENT_FACTORY_EC (DDSC_LOG_BASE + 605)

  Entities are in different factories.
- #define DDSC_LOG_DATAWRITER_CDR 1

  The datawriter CDR kind.
- #define DDSC_LOG_DATAREADER_CDR 2

  The datareader CDR kind.
- #define DDSC_LOG_CDR_POOL_ALLOC_EC (DDSC_LOG_BASE + 700)

  Failed to allocate a pool of the specified kind.
- #define DDSC_LOG_CDR_BUFFER_SET_EC (DDSC_LOG_BASE + 701)

  Failed to set the CDR buffer for a packet.
- #define DDSC_LOG_CDR_POOL_DELETE_EC (DDSC_LOG_BASE + 702)

  Failed to delete the CDR pool.
- #define DDSC_LOG_CDR_SERIALIZE_PID_EC (DDSC_LOG_BASE + 703)

  Failed to serialize a parameter ID.
- #define DDSC_LOG_CDR_SERIALIZE_PID_LENGTH_EC (DDSC_LOG_BASE + 704)

  Failed to serialize a parameter length.
- #define DDSC_LOG_CDR_SERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 705)

  Failed to serialize a key-hash.
- #define DDSC_LOG_CDR_SERIALIZE_DATA_EC (DDSC_LOG_BASE + 706)

  Failed to serialize payload data.
- #define DDSC_LOG_DESERIALIZE_BAD_PID_LENGTH_EC (DDSC_LOG_BASE + 707)

Deserialized an invalid parameter length for a specific parameter ID.

- #define DDSC_LOG_CDR_DESERIALIZE_PID_EC (DDSC_LOG_BASE + 708)

  Failed to deserialize the ID of an inline parameter.
- #define DDSC_LOG_CDR_DESERIALIZE_PID_LENGTH_EC (DDSC_LOG_BASE + 709)

  Failed to deserialize the length of an inline parameter.
- #define DDSC_LOG_CDR_INCREMENT_POS_EC (DDSC_LOG_BASE + 710)

  Failed to increment to the position of the next inline parameter.
- #define DDSC_LOG_CDR_SET_POS_EC (DDSC_LOG_BASE + 711)

  Failed to set the reception stream position.
- #define DDSC_LOG_CDR_DESERIALIZE_HEADER_EC (DDSC_LOG_BASE + 712)

  Failed to deserialize the encapsulation header.
- #define DDSC_LOG_CDR_DESERIALIZE_DATA_EC (DDSC_LOG_BASE + 713)

  Failed to deserialize CDR payload data.
- #define DDSC_LOG_CDR_INITIALIZE_SAMPLE_EC (DDSC_LOG_BASE + 714)

  Failed to initialize CDR sample.
- #define DDSC_LOG_CDR_FINALIZE_SAMPLE_EC (DDSC_LOG_BASE + 715)

  Failed to finalize sample.
- #define DDSC_LOG_CDR_SERIALIZE_STATUS_INFO_EC (DDSC_LOG_BASE + 716)

  Failed to serialize the status info parameter.
- #define DDSC_LOG_CDR_DESERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 717)

  Failed to deserialize a key-hash.
- #define DDSC_LOG_CDR_DESERIALIZE_KEY_EC (DDSC_LOG_BASE + 718)

  Failed to deserialize CDR payload key.
- #define DDSC_LOG_NETIO_ADD_ANON_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 800)

  Failed to add a route to an anonymous participant discovery datawriter.
- #define DDSC_LOG_NETIO_ADD_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 801)

  Failed to add a route to a topic from a datawriter.
- #define DDSC_LOG_NETIO_DELETE_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 802)

  Failed to delete a route to a topic.
- #define DDSC_LOG_NETIO_FORWARD_TOPIC_EC (DDSC_LOG_BASE + 803)

  Failed to forward a topic.
- #define DDSC_LOG_NETIO_NETIO_KIND 1

  Any NETIO interface kind, typically UDP.
- #define DDSC_LOG_INTRA_NETIO_KIND 2

  The intra interface kind.
- #define DDSC_LOG_RTPS_NETIO_KIND 3

  The RTPS interface kind.
- #define DDSC_LOG_DATAREADER_NETIO_KIND 4

  The DataReader interface kind.
- #define DDSC_LOG_DATAWRITER_NETIO_KIND 5

  The DataWriter interface kind.
- #define DDSC_LOG_NETIO_BIND_EXTERNAL_EC (DDSC_LOG_BASE + 804)

  Failed to bind two external interface of the specified kind.
- #define DDSC_LOG_NETIO_UNBIND_EXTERNAL_EC (DDSC_LOG_BASE + 805)

  Failed to unbind two external interfaces of the specified kind.
- #define DDSC_LOG_NETIO_BIND_EC (DDSC_LOG_BASE + 806)

  Failed to bind an interface to a peer interface.
- #define DDSC_LOG_NETIO_UNBIND_EC (DDSC_LOG_BASE + 807)

  Failed to unbind an interface from a peer interface.
- #define DDSC_LOG_NETIO_ADD_ROUTE_EC (DDSC_LOG_BASE + 808)

  Failed to add a route from an interface to a peer interface.

- #define DDSC_LOG_NETIO_DELETE_ROUTE_EC (DDSC_LOG_BASE + 809)

  Failed to delete a route from an interface to a peer interface.
- #define DDSC_LOG_NETIO_GET_EXTERNAL_INTF_EC (DDSC_LOG_BASE + 810)

  Failed to get an external interface for the specified interface kind.
- #define DDSC_LOG_NETIO_NO_ROUTE_EC (DDSC_LOG_BASE + 811)

  A DataReader failed a bind due to no existing route.
- #define DDSC_LOG_NETIO_ROUTE_LOOKUP_FAILED_EC (DDSC_LOG_BASE + 812)

  Lookup a route to a destination failed.
- #define DDSC_LOG_NETIO_GET_ROUTE_TABLE_FAILED_EC (DDSC_LOG_BASE + 813)

  Failed to get the route table for an interface.
- #define DDSC_LOG_NETIO_SEND_FAILED_EC (DDSC_LOG_BASE + 814)

  Failure when sending on an interface.
- #define DDSC_LOG_NETIO_SET_STATE_EC (DDSC_LOG_BASE + 815)

  Failed to set an interface state.
- #define DDSC_LOG_NETIO_PEER_LOOKUP_EC (DDSC_LOG_BASE + 816)

  Datawriter did not find a peer.
- #define DDSC_LOG_NETIO_FORCED_REMOVE_EC (DDSC_LOG_BASE + 817)

  Forced removal of sample downstream failed.
- #define DDSC_LOG_PACKET_INIT_EC (DDSC_LOG_BASE + 818)

  Failed to initialize a packet.
- #define DDSC_LOG_PACKET_SET_HEAD_EC (DDSC_LOG_BASE + 819)

  Failed to set the head of a packet.
- #define DDSC_LOG_PACKET_SET_TAIL_EC (DDSC_LOG_BASE + 820)

  Failed to set the tail of a packet.
- #define DDSC_LOG_NETIO_DISCOVERY_ENABLED_TRANSPORT_EC (DDSC_LOG_BASE + 821)

  Configured discovery enabled transport is not valid.
- #define DDSC_LOG_DW_ACKNACK_FAILED_EC (DDSC_LOG_BASE + 900)

  Failed to ACKNACK sample in the writer history.
- #define DDSC_LOG_DW_COMMIT_EC (DDSC_LOG_BASE + 901)

  Failed to commit a sample to the writer queue.
- #define DDSC_LOG_DW_KEYHASH_CREATE_EC (DDSC_LOG_BASE + 902)

  Failed to create keyhash of instance handle.
- #define DDSC_LOG_DW_ILLEGAL_KEY_KIND_EC (DDSC_LOG_BASE + 903)

  Failed a write due to an invalid key kind for the type being written.
- #define DDSC_LOG_DW_CREATE_TYPED_WRITER_EC (DDSC_LOG_BASE + 904)

  Failed to create a typed writer.
- #define DDSC_LOG_DW_HISTORY_REGISTER_KEY_EC (DDSC_LOG_BASE + 905)

  Failed to register the key of an instance.
- #define DDSC_LOG_DR_CREATE_TYPED_READER_EC (DDSC_LOG_BASE + 1000)

  Failed to create a typed datareader.
- #define DDSC_LOG_DR_COPY_DATA_SAMPLE_EC (DDSC_LOG_BASE + 1001)

  Failed to copy a sample upon reception, read, or take.
- #define DDSC_LOG_DR_COMMIT_SAMPLE_EC (DDSC_LOG_BASE + 1002)

  Failed to commit a sample to be made available to be read or taken.
- #define DDSC_LOG_DR_FILTER_ERROR_EC (DDSC_LOG_BASE + 1003)

  A datareader filter function failed.
- #define DDSC_LOG_DR_DESERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 1004)

  Failed to deserialize a key-hash parameter.
- #define DDSC_LOG_DR_GET_ENTRY_FAILED_EC (DDSC_LOG_BASE + 1005)

  Failed to get a Reader History entry for a received sample.
- #define DDSC_LOG_DR_COMMIT_ENTRY_EC (DDSC_LOG_BASE + 1006)

Failed to commit a receive sample to Reader History to be read or taken.

- #define DDSC_LOG_DR_UNREGISTER_KEY_EC (DDSC_LOG_BASE + 1007)

  A DataReader failed to unregister an instance.
- #define DDSC_LOG_DR_DISPOSE_KEY_EC (DDSC_LOG_BASE + 1008)

  A DataReader failed to dispose an instance.
- #define DDSC_LOG_DR_READ_TAKE_FAILURE_EC (DDSC_LOG_BASE + 1009)

  A call to a reader/take function failed.
- #define DDSC_LOG_DR_INSTANCE_TO_KEYHASH_EC (DDSC_LOG_BASE + 1010)

  Failed to create keyhash of instance handle.
- #define DDSC_LOG_TYPE_NAME_CMP_EC (DDSC_LOG_BASE + 1100)

  Two type names are incompatible.
- #define DDSC_LOG_TOPIC_NAME_CMP_EC (DDSC_LOG_BASE + 1101)

  Two topic names are incompatible.
- #define DDSC_LOG_TYPE_FUNCTION_GET_SERIALIZED_SAMPLE_MAX_SIZE 1

  The get_serialized_sample_max_size function kind.
- #define DDSC_LOG_TYPE_FUNCTION_SERIALIZE_DATA 2

  The serialize_data function kind.
- #define DDSC_LOG_TYPE_FUNCTION_DESERIALIZE_DATA 3

  The deserialize_data function kind.
- #define DDSC_LOG_TYPE_FUNCTION_CREATE_SAMPLE 4

  The create_sample function kind.
- #define DDSC_LOG_TYPE_FUNCTION_COPY_SAMPLE 5

  The copy_sample function kind.
- #define DDSC_LOG_TYPE_FUNCTION_DELETE_SAMPLE 6

  The delete_sample function kind.
- #define DDSC_LOG_TYPE_FUNCTION_GET_KEY_KIND 7

  The get_key_kind function kind.
- #define DDSC_LOG_TYPE_FUNCTION_INSTANCE_TO_KEYHASH 8

  The instance_to_keyhash function kind.
- #define DDSC_LOG_TYPE_FUNCTION_NULL_EC (DDSC_LOG_BASE + 1102)

  Invalid type plugin, The specified function pointer is NULL.
- #define DDSC_LOG_TOPIC_TOO_LONG_EC (DDSC_LOG_BASE + 1103)

  Failed to create a topic because the name exceeded the maximum length of 255 octets (excluding the terminating NUL)
- #define DDSC_LOG_TYPE_TOO_LONG_EC (DDSC_LOG_BASE + 1104)

  Failed to create a type because the name exceeded the maximum length of 255 octets (excluding the terminating NUL)
- #define DDSC_LOG_LOOKUP_TYPE_PLUGIN_EC (DDSC_LOG_BASE + 1105)

  A type-plugin for the given type could not be found.
- #define DDSC_LOG_TYPE_KEY_TYPE_EC (DDSC_LOG_BASE + 1106)

  Two types have incompatible keys.
- #define DDSC_LOG_TYPE_MANAGED_POOL_FAILURE_EC (DDSC_LOG_BASE + 1107)

  Managed pool function failed.
- #define DDSC_LOG_TYPE_GET_SAMPLE_ID_EC (DDSC_LOG_BASE + 1108)

  Managed pool failed to get sample id.
- #define DDSC_LOG_DISC_LOCAL_PARTICIPANT_ENABLED_EC (DDSC_LOG_BASE + 1200)

  Discovery plugin failed its update after a local DomainParticipant was enabled.
- #define DDSC_LOG_DISC_BEFORE_LOCAL_PARTICIPANT_CREATED_EC (DDSC_LOG_BASE + 1201)

  Discovery plugin failed its update before a local DomainParticipant was created.
- #define DDSC_LOG_DISC_AFTER_LOCAL_PARTICIPANT_CREATED_EC (DDSC_LOG_BASE + 1202)

Discovery plugin failed its update after a local DomainParticipant was created.

- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_ENABLED_EC (DDSC_LOG_BASE + 1203)

  Discovery plugin failed its update after a local DataReader was enabled.
- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_DELETED_EC (DDSC_LOG_BASE + 1204)

  Discovery plugin failed its update after a local DataReader was deleted.
- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_ENABLED_EC (DDSC_LOG_BASE + 1205)

  Discovery plugin failed its update after a local DataWriter was enabled.
- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_DELETED_EC (DDSC_LOG_BASE + 1206)

  Discovery plugin failed its update after a local DataWriter was deleted.
- #define DDSC_LOG_DISC_REMOTE_PARTICIPANT_EXPIRED_EC (DDSC_LOG_BASE + 1207)

  Discovery plugin failed its update after a remote DomainParticipant's liveliness expired.
- #define DDSC_LOG_DISC_ADD_PEER_EC (DDSC_LOG_BASE + 1208)

  Failed to add a peer with discovery plugin.
- #define DDSC_LOG_CHECKSUM_REQUIRED_EC (DDSC_LOG_BASE + 1301)

  A participant requires checksum, but the discovered participant does not send a checksum.
- #define DDSC_LOG_CHECKSUM_INCOMPATIBLE_EC (DDSC_LOG_BASE + 1302)

  A participant does not support checksum's sent by a discovered participant.
- #define DDSC_LOG_CHECKSUM_INCONSISTENT_CLASS_EC (DDSC_LOG_BASE + 1303)

  The checksum class id for a custom checksum does match.
- #define DDSC_LOG_CHECKSUM_INCONSISTENT_COMPUTE_EC (DDSC_LOG_BASE + 1304)

  The computed_crc_kind is inconsistent with what is allowed or supported.
- #define DDSC_LOG_CHECKSUM_INCONSISTENT_ALLOW_EC (DDSC_LOG_BASE + 1305)

  The allowed_crc_mask is inconsistent with what is supported.

5.86.1 Detailed Description

DDS C. ModuleID = 7.

5.86.2 Macro Definition Documentation

5.86.2.1 #define DDSC_LOG_INVALID_DURATION_EC (DDSC_LOG_BASE + 1)

The specified duration is not valid.

5.86.2.2 #define DDSC_LOG_INVALID_PARTICIPANT_NAME_EC (DDSC_LOG_BASE + 2)

An invalid participant name was specified with an unknown GUID.

5.86.2.3 #define DDSC_LOG_INVALID_PARTICIPANT_GUID_PREFIX_EC (DDSC_LOG_BASE + 3)

An invalid participant GUID prefix was specified, typically the GUID prefix does not match an already detected participant.

5.86.2.4 #define DDSC_LOG_SYS_GETTIME_EC (DDSC_LOG_BASE + 4)

A call to OSAPI_System_get_time failed.

5.86.2.5 #define DDSC_LOG_GET_NEXT_OBJECT_ID_EC (DDSC_LOG_BASE + 5)

Failed to get the next automatically generated object ID for an entity's GUID. This typically means the object id pool has been exhausted.

5.86.2.6 #define DDSC_LOG_SET_ENTITY_NAME_EC (DDSC_LOG_BASE + 6)

Failed to set name string for a DDS entity in the DomainParticipantQos entity_name policy.

5.86.2.7 #define DDSC_LOG_SYS_GET_HOSTNAME_EC (DDSC_LOG_BASE + 7)

A call to OSAPI_System_get_hostname failed.

5.86.2.8 #define DDSC_LOG_IO_SNPRINTF_FAILED_EC (DDSC_LOG_BASE + 8)

A call to OSAPI_Stdio_snprintf failed. Typically this means the destination buffer was too small.

5.86.2.9 #define DDSC_LOG_TOPIC_FIND_EC (DDSC_LOG_BASE + 9)

Failed to find a topic created by a DomainParticipant.

5.86.2.10 #define DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_NAME_EC (DDSC_LOG_BASE + 10)

Endpoint discovery failed because the name of the remote participant parent for an endpoint was not found.

5.86.2.11 #define DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_KEY_EC (DDSC_LOG_BASE + 11)

Endpoint discovery failed because the key of the remote participant parent for an endpoint was not found. Note that the key is logged as 4 integers in host endianess format.

5.86.2.12 #define DDSC_LOG_REMOTE_PARTICIPANT_KEY_NOT_EQUAL_EC (DDSC_LOG_BASE + 12)

Failed endpoint discovery when key does not match the remote participant.

5.86.2.13 #define DDSC_LOG_INVALID_ENDPOINT_GUID_EC (DDSC_LOG_BASE + 13)

Failed endpoint discovery due to an invalid or unknown endpoint GUID.

5.86.2.14 #define DDSC_LOG_ENDPOINT_NOT_CHILD_OF_PARTICIPANT_EC (DDSC_LOG_BASE + 14)

Failed endpoint discovery when an endpoint is determined to belong to a different remote participant.

5.86.2.15 #define DDSC_LOG_PARTICIPANT_DOES_NOT_EXIST_EC (DDSC_LOG_BASE + 15)

Failed participant discovery because a remote participant that should have been already asserted locally was not found.

5.86.2.16 #define DDSC_LOG_REFRESH_REM_PARTICIPANT_EC (DDSC_LOG_BASE + 16)

Did not find a remote participant when asserting participant liveliness for it. Note that the key is logged as 4 integers in host endianess format.

5.86.2.17 #define DDSC_LOG_REFRESH_REM_PARTICIPANT_TIMEOUT_EC (DDSC_LOG_BASE + 17)

Failed to assert participant liveliness to a remote participant.

5.86.2.18 #define DDSC_LOG_PARTICIPANT_LOOKUP_EC (DDSC_LOG_BASE + 18)

Failed to find a remote participant as previously discovered.

5.86.2.19 #define DDSC_LOG_REMOVE_PUBLICATION_EC (DDSC_LOG_BASE + 19)

Failed to remove resources for a remote publication.

5.86.2.20 #define DDSC_LOG_REMOVE_SUBSCRIPTION_EC (DDSC_LOG_BASE + 20)

Failed to remove resources for a remote subscription.

5.86.2.21 #define DDSC_LOG_FIND_PUBLICATION_PARENT_EC (DDSC_LOG_BASE + 21)

Cannot determine the participant of a remote publication.

5.86.2.22 #define DDSC_LOG_FIND_SUBSCRIPTION_PARENT_EC (DDSC_LOG_BASE + 22)

Cannot determine the participant of a remote subscription.

5.86.2.23 #define DDSC_LOG_MAX_PARTICIPANT_ID_REACHED_EC (DDSC_LOG_BASE + 23)

Failed to create DomainParticipant due to running out of participant IDs. This is typically caused by all UDP ports being used.

5.86.2.24 #define DDSC_LOG_RESERVE_LOCATORS_EC (DDSC_LOG_BASE + 24)

Failed to reserve endpoint locators.

5.86.2.25 #define DDSC_LOG_TIMER_CREATE_TIMEOUT_EC (DDSC_LOG_BASE + 25)

Failed to create a timeout.

5.86.2.26 #define DDSC_LOG_ILLEGAL_OBJECTID_EC (DDSC_LOG_BASE + 26)

Illegal object id specified.

Manually specifed object id must be in the range [0,0xffffff]

5.86.2.27 #define DDSC_LOG_TOPIC_NARROW_EC (DDSC_LOG_BASE + 27)

Failed to narrow a TopicDescription to the named Topic.

5.86.2.28 #define DDSC_LOG_FAILED_UPDATE_STATUS_CONDITION_EC (DDSC_LOG_BASE + 28)

Failed to update state of StatusCondition.

An error occurred when trying to update the state (i.e. the trigger value) of a StatusCondition.

5.86.2.29 #define DDSC_LOG_WS_REMOVE_COND_REFERENCE_EC (DDSC_LOG_BASE + 29)

Failed to remove a condition reference from a waitset.

5.86.2.30 #define DDSC_LOG_WS_ADD_COND_REFERENCE_EC (DDSC_LOG_BASE + 30)

Failed to add a condition reference to a waitset.

5.86.2.31 #define DDSC_LOG_RELEASE_META_MC_EC (DDSC_LOG_BASE + 31)

Failed to release resources for multicast discovery locators.

5.86.2.32 #define DDSC_LOG_RELEASE_META_UC_EC (DDSC_LOG_BASE + 32)

Failed to release resources for unicast discovery locators.

5.86.2.33 #define DDSC_LOG_RELEASE_USER_MC_EC (DDSC_LOG_BASE + 33)

Failed to release resources for multicast user locators.

5.86.2.34 #define DDSC_LOG_RELEASE_USER_UC_EC (DDSC_LOG_BASE + 34)

Failed to release resources for unicast user locators.

5.86.2.35 #define DDSC_LOG_INVALID_DOMAINID_EC (DDSC_LOG_BASE + 35)

The domain ID specified exceeds what is allowed based on the parameters specified in dp_qos.protocol.rtps_well-_known_ports.

5.86.2.36 #define DDSC_LOG_RESOLVE_LOCATORS_PRIORITY_EC (DDSC_LOG_BASE + 36)

Failed to resolve locators based on priority.

5.86.2.37 #define DDSC_LOG_DATABASE_CREATE_EC (DDSC_LOG_BASE + 100)

Failed to create database.

5.86.2.38 #define DDSC_LOG_DATABASE_DELETE_EC (DDSC_LOG_BASE + 101)

Failed to delete database.

5.86.2.39 #define DDSC_LOG_TABLE_CREATE_EC (DDSC_LOG_BASE + 102)

Failed to create database table of the specified name.

5.86.2.40 #define DDSC_LOG_TABLE_INUSE_EC (DDSC_LOG_BASE + 103)

Failed to delete a database table because it is not empty.

5.86.2.41 #define DDSC_LOG_TABLE_DELETE_EC (DDSC_LOG_BASE + 104)

Failed to delete a database table.

5.86.2.42 #define DDSC_LOG_TABLE_SELECT_EC (DDSC_LOG_BASE + 105)

A selection operation failed on the specified database table.

5.86.2.43 #define DDSC_LOG_CREATE_INDEX_EC (DDSC_LOG_BASE + 106)

Failed to create an index on a database table.

5.86.2.44 #define DDSC_LOG_DELETE_INDEX_EC (DDSC_LOG_BASE + 107)

Failed to delete an index on a database table.

5.86.2.45 #define DDSC_LOG_DB_CURSOR_INVALIDATED_EC (DDSC_LOG_BASE + 108)

A database table cursor was invalidated while in use.

5.86.2.46 #define DDSC_LOG_SUBSCRIPTION_RECORD 1

Database record for a Subscription.

5.86.2.47 #define DDSC_LOG_PUBLICATION_RECORD 2

Database record for a Publication.

5.86.2.48 #define DDSC_LOG_PARTICIPANT_RECORD 3

Database record for a Remote Participant.

5.86.2.49 #define DDSC_LOG_TOPIC_RECORD 4

Database record for a Topic.

5.86.2.50 #define DDSC_LOG_DATAWRITER_RECORD 5

Database record for a DataWriter.

5.86.2.51 #define DDSC_LOG_DATAREADER_RECORD 6

Database record for a DataReader.

5.86.2.52 #define DDSC_LOG_PUBLISHER_RECORD 7

Database record for a Publisher.

5.86.2.53 #define DDSC_LOG_SUBSCRIBER_RECORD 8

Database record for a Subscriber.

5.86.2.54 #define DDSC_LOG_ROUTE_RECORD 9

Database record for a route record.

5.86.2.55 #define DDSC_LOG_BIND_RECORD 10

Database record for a bind record.

5.86.2.56 #define DDSC_LOG_TYPE_RECORD 11

Database record for a type plugin.

5.86.2.57 #define DDSC_LOG_STRING_RECORD 14

String record.

5.86.2.58 #define DDSC_LOG_RECORD_CREATE_EC (DDSC_LOG_BASE + 109)

Failed to create a database record of the specified kind.

The creation of an internal database record failed The failure may have been caused by insufficient resources based on the record kind. The following resource-limits may apply:

DomainParticipantQos.resource_limit.local_publisher_allocation limits the number of DDS Publishers.

DomainParticipantQos.resource_limit.local_subscriber_allocation limits the number of DDS Subscribers

DomainParticipantQos.resource_limit.local_topic_allocation limits the number of DDS Topics.

DomainParticipantQos.resource_limits.local_reader_allocation limits the number of DDS DataReader records.

DomainParticipantQos.resource_limits.local_writer_allocation limits the number of DDS DataWriter records.

DomainParticipantQos.resource_limits.remote_writer_allocation limits the number of remote publication records

DomainParticipantQos.resource_limits.remote_reader_allocation limits the number of remote subscription records.

DomainParticipantQos.resource_limits.remote_participant_allocation limits the number of remote participant records.

DataWriterQos.writer_resource_limits.max_remote_readers limits of the number of DDS DataReaders a DDS DataWriter can communicate with.

DataReaderQos.reader_resource_limits.max_remote_writers limits of the number of DDS DataWriter a DDS DataReader can communicate with.

5.86.2.59 #define DDSC_LOG_RECORD_DELETE_EC (DDSC_LOG_BASE + 110)

Failed to delete a database record of the specified kind.

5.86.2.60 #define DDSC_LOG_RECORD_INSERT_EC (DDSC_LOG_BASE + 111)

Failed to insert a database record of the specified kind.

5.86.2.61 #define DDSC_LOG_RECORD_ERROR_EC (DDSC_LOG_BASE + 112)

Unknown error for database record of the specified kind.

5.86.2.62 #define DDSC_LOG_RECORD_EXISTS_EC (DDSC_LOG_BASE + 113)

A database record of the specified kind already exists.

5.86.2.63 #define DDSC_LOG_RECORD_LOOKUP_EC (DDSC_LOG_BASE + 114)

A lookup of a database record of the specified kind failed.

5.86.2.64 #define DDSC_LOG_RECORD_NOT_EXISTS_EC (DDSC_LOG_BASE + 115)

A database record of the specified kind does not exist.

5.86.2.65 #define DDSC_LOG_RECORD_SELECT_EC (DDSC_LOG_BASE + 116)

A database select on the specified record kind failed.

5.86.2.66 #define DDSC_LOG_RECORD_REMOVE_EC (DDSC_LOG_BASE + 117)

Removal a database record of the specified kind failed.

5.86.2.67 #define DDSC_LOG_RECORD_INITIALIZE_EC (DDSC_LOG_BASE + 118)

A database record of the specified kind could not be initialized.

5.86.2.68 #define DDSC_LOG_RECORD_FINALIZE_EC (DDSC_LOG_BASE + 119)

A database record of the specified kind could not be finalized.

5.86.2.69 #define DDSC_LOG_PUBLICATIONDATA_OBJECT 1

DDS PublicationBuiltinTopicData object.

5.86.2.70 #define DDSC_LOG_SUBSCRIPTIONDATA_OBJECT 2

DDS SubscriptionBuiltinTopicData object.

5.86.2.71 #define DDSC_LOG_PARTICIPANTDATA_OBJECT 3

DDS ParticipantBuiltinTopicData object.

5.86.2.72 #define DDSC_LOG_DATAREADERQOS_OBJECT 4

DDS DataReaderQos object.

5.86.2.73 #define DDSC_LOG_DATAWRITERQOS_OBJECT 5

DDS DataWriterQos object.

5.86.2.74 #define DDSC_LOG_TOPICQOS_OBJECT 6

DDS TopicQos object.

5.86.2.75 #define DDSC_LOG_PUBLISHERQOS_OBJECT 7

DDS PublisherQos object.

5.86.2.76 #define DDSC_LOG_SUBSCRIBERQOS_OBJECT 8

DDS SubscriberQos object.

5.86.2.77 #define DDSC_LOG_PARTICIPANTQOS_OBJECT 9

DDS DomainParticipantQos object.

5.86.2.78 #define DDSC_LOG_ENTITY_OBJECT 10

DDS Entity object.

5.86.2.79 #define DDSC_LOG_PARTICIPANTFACTORYQOS_OBJECT 11

DDS DomainParticipantFactoryQos object.

5.86.2.80 #define DDSC_LOG_TYPE_OBJECT 12

A DDS Type object.

5.86.2.81 #define DDSC_LOG_BINDRESOLVER_OBJECT 13

A NETIO BindResolver object.

5.86.2.82 #define DDSC_LOG_ROUTERESOLVER_OBJECT 14

A NETIO RouteResolver object.

5.86.2.83 #define DDSC_LOG_ADDRESSRESOLVER_OBJECT 15

A NETIO AddressResolver object.

5.86.2.84 #define DDSC_LOG_DATAWRITERIO_OBJECT 16

A NETIO DataWriterInterface object.

5.86.2.85 #define DDSC_LOG_DATAREADERIO_OBJECT 17

A NETIO DataReaderInterface object.

5.86.2.86 #define DDSC_LOG_LOG_OBJECT 18

A OSAPI Log object.

5.86.2.87 #define DDSC_LOG_SYSTEM_OBJECT 19

A OSAPI system object.

5.86.2.88 #define DDSC_LOG_MUTEX_OBJECT 20

A OSAPI mutex object.

5.86.2.89 #define DDSC_LOG_CONDREF_OBJECT 21

A DDS Waitset condition reference object.

5.86.2.90 #define DDSC_LOG_WSREF_OBJECT 22

A DDS Condition waitset reference object.

5.86.2.91 #define DDSC_LOG_MD5STREAM_OBJECT 23

A MD5 stream object.

5.86.2.92 #define DDSC_LOG_CONDITION_OBJECT 24

A DDS Condition object.

5.86.2.93 #define DDSC_LOG_RT_OBJECT 25

A RT Condition object.

5.86.2.94 #define DDSC_LOG_PARTICIPANT_POOL_OBJECT 26

A Participant pool object.

5.86.2.95 #define DDSC_LOG_TIMER_OBJECT 27

A OSAPI_Timer object.

5.86.2.96 #define DDSC_LOG_TIMEROUT_OBJECT 28

A OSAPI_Timer timeout object.

5.86.2.97 #define DDSC_LOG_TOPIC_OBJECT 29

A DDS Topic object.

5.86.2.98 #define DDSC_LOG_WAITSET_OBJECT 30

A DDS WaitSet object.

5.86.2.99 #define DDSC_LOG_UUID_OBJECT 31

A UUID object.

5.86.2.100 #define DDSC_LOG_STRING_OBJECT 40

A String object.

5.86.2.101 #define DDSC_LOG_STRINGMANAGER_OBJECT 43

A String Manager object.

5.86.2.102 #define DDSC_LOG_OBJECT_INITIALIZE_EC (DDSC_LOG_BASE + 200)

Out of resources to initialize object of the specified kind.

5.86.2.103 #define DDSC_LOG_OBJECT_ALLOCATE_EC (DDSC_LOG_BASE + 201)

Out of resources to allocate an object of the specified kind.

5.86.2.104 #define DDSC_LOG_OBJECT_FINALIZE_EC (DDSC_LOG_BASE + 202)

Failed to finalize object of specified kind.

5.86.2.105 #define DDSC_LOG_OBJECT_DELETE_EC (DDSC_LOG_BASE + 203)

Failed to delete object of specified kind.

5.86.2.106 #define DDSC_LOG_OBJECT_COPY_EC (DDSC_LOG_BASE + 204)

Failed to copy object of specified kind.

5.86.2.107 #define DDSC_LOG_OBJECT_REFCOUNT_EC (DDSC_LOG_BASE + 205)

Failed to delete/finalize an object because other objects are referencing it.

5.86.2.108 #define DDSC_LOG_OBJECT_GET_PROPERTY_EC (DDSC_LOG_BASE + 206)

Failed to get the object properties.

5.86.2.109 #define DDSC_LOG_OBJECT_SET_PROPERTY_EC (DDSC_LOG_BASE + 207)

Failed to set the object properties.

5.86.2.110 #define DDSC_LOG_OBJECT_EMPTY_EC (DDSC_LOG_BASE + 208)

An object is empty, typically applies only to buffer-pool objects.

5.86.2.111 #define DDSC_LOG_OBJECT_INUSECOUNT_EC (DDSC_LOG_BASE + 209)

Failed to delete/finalize an object because other objects are using it.

5.86.2.112 #define DDSC_LOG_OBJECT_CREATE_EC (DDSC_LOG_BASE + 210)

Failed to create an object.

5.86.2.113 #define DDSC_LOG_NETMASK_SEQUENCE 1

A NETIO Netmask sequence.

5.86.2.114 #define DDSC_LOG_ROUTE_SEQUENCE 2

A NETIO Route sequence.

5.86.2.115 #define DDSC_LOG_RESERVED_SEQUENCE 3

A NETIO Reserved address sequence.

5.86.2.116 #define DDSC_LOG_ENABLED_USER_TRANSPORT_SEQUENCE 4

A DDS sequence of enabled user NETIO transports.

5.86.2.117 #define DDSC_LOG_ENABLED_TRANSPORT_SEQUENCE 5

A DDS sequence of enabled transports.

5.86.2.118 #define DDSC_LOG_ENABLED_DISCVOERY_TRANSPORT_SEQUENCE 6

A DDS sequence of enabled discovery transports.

5.86.2.119 #define DDSC_LOG_INITIAL_PEER_SEQUENCE 7

A DDS sequence of initial peers.

5.86.2.120 #define DDSC_LOG_DESTINATION_SEQUENCE 8

A NETIO sequence of destinations to send to.

5.86.2.121 #define DDSC_LOG_METAUNICAST_SEQUENCE 9

A DDS sequence of meta-traffic unicast locators.

5.86.2.122 #define DDSC_LOG_METAMULTICAST_SEQUENCE 10

A DDS sequence of meta-traffic multicast locators.

5.86.2.123 #define DDSC_LOG_USERUNICAST_SEQUENCE 11

A DDS sequence of user-traffic unicast locators.

5.86.2.124 #define DDSC_LOG_USERMULTICAST_SEQUENCE 12

A DDS sequence of user-traffic multicast locators.

5.86.2.125 #define DDSC_LOG_READTAKE_SEQUENCE 13

A DDS sequence used in a read/take call.

5.86.2.126 #define DDSC_LOG_SEQUENCE_SETMAX_EC (DDSC_LOG_BASE + 300)

Failed to set the maximum length of a sequence of the specified kind.

5.86.2.127 #define DDSC_LOG_SEQUENCE_SETLENGTH_EC (DDSC_LOG_BASE + 301)

Failed to set the length of a sequence of the specified kind.

5.86.2.128 #define DDSC_LOG_SEQUENCE_GETREF_EC (DDSC_LOG_BASE + 302)

Failed to get a reference at the specified index for a sequence of the specified kind.

5.86.2.129 #define DDSC_LOG_SEQUENCE_INITIALIZE_EC (DDSC_LOG_BASE + 303)

Failed to initialize a sequence of the specified kind.

5.86.2.130 #define DDSC_LOG_SEQUENCE_FINALIZE_EC (DDSC_LOG_BASE + 304)

Failed to finalize a sequence of the specified kind.

5.86.2.131 #define DDSC_LOG_SEQUENCE_COPY_EC (DDSC_LOG_BASE + 305)

Failed to copy a sequence of the specified kind.

5.86.2.132 #define DDSC_LOG_SEQUENCE_INVALID_EC (DDSC_LOG_BASE + 306)

The sequence of the specified kind was invalid in the context it is used.

5.86.2.133 #define DDSC_LOG_DISCOVERY_COMPONENT 1

A component of discovery kind.

5.86.2.134 #define DDSC_LOG_RTPS_COMPONENT 2

A component of RTPS kind.

5.86.2.135 #define DDSC_LOG_READERHISTORY_COMPONENT 3

A component of reader-history kind.

5.86.2.136 #define DDSC_LOG_WRITERHISTORY_COMPONENT 4

A component of writer-history kind.

5.86.2.137 #define DDSC_LOG_DATAREADERIO_COMPONENT 5

A component of DataReaderInterface kind.

5.86.2.138 #define DDSC_LOG_DATAWRITERIO_COMPONENT 6

A component of DataWriterInterface kind.

5.86.2.139 #define DDSC_LOG_NETIO_COMPONENT 7

A component of NETIO kind.

5.86.2.140 #define DDSC_LOG_TRANSPORT_COMPONENT 8

A component of transport kind.

5.86.2.141 #define DDSC_LOG_COMPONENT_LOOKUP_EC (DDSC_LOG_BASE + 400)

Did not find a component factory with the given name in the registry.

5.86.2.142 #define DDSC_LOG_COMPONENT_CREATE_EC (DDSC_LOG_BASE + 401)

Could not create a component of the specified kind using the specified factory.

5.86.2.143 #define DDSC_LOG_COMPONENT_DELETE_EC (DDSC_LOG_BASE + 402)

Could not delete a component of the specified kind using the specified factory.

5.86.2.144 #define DDSC_LOG_HISTORY_RESOURCE 1

Exceeded resource limits for writer history queue.

A DataWriter failed to get a queue entry for a new sample it is attempting to send, due to exceeding a limit:

- When the sample is for new instance: DataWriterQos.resource_limits.max_instances may have been exceeded
- When the sample is for an existing instance, and History kind is KEEP_ALL: DataWriterQos.resource_limits.max_samples_per_instance may have been exceeded.
- The limit on the total number of samples in the queue: DataWriterQos.resource_limits.max_samples, may have been exceeded.

5.86.2.145 #define DDSC_LOG_SAMPLE_RESOURCES 2

Failed to get resource for new sample due to resource limit.

A DataWriter failed to get a buffer for a new sample being written because the limit DataWriterQos.resource_limits.max_samples was exceeded.

A DataReader failed in getting a buffer for a newly received sample because DataReaderQos.resource_limits.max_samples was exceeded.

5.86.2.146 #define DDSC_LOG_PARTICIPANT_RESOURCES 3

Failed to allocate a new participant.

The DomainParticipantFactoryfailed to get a buffer for a new DomainParticipant because DomainParticipantFactoryQos.system_resource.max_participants because was exceeded.

5.86.2.147 #define DDSC_LOG_RESOURCE_EXCEEDED_EC (DDSC_LOG_BASE + 500)

Could not allocate a resource of the specified kind.

5.86.2.148 #define DDSC_LOG_TOPIC_QOS 1

The TopicQos kind.

5.86.2.149 #define DDSC_LOG_DATAREADER_QOS 2

The DataReaderQos kind.

5.86.2.150 #define DDSC_LOG_DATAWRITER_QOS 3

The DataWriterQos kind.

5.86.2.151 #define DDSC_LOG_SUBSCRIBER_QOS 4

The SubscriberQos kind.

5.86.2.152 #define DDSC_LOG_PUBLISHER_QOS 5

The PublisherQos kind.

5.86.2.153 #define DDSC_LOG_PARTICIPANT_QOS 6

The DomainParticipantQos kind.

5.86.2.154 #define DDSC_LOG_PARTICIPANTFACTORY_QOS 7

The DomainParticipantFactoryQos kind.

5.86.2.155 #define DDSC_LOG_DEFAULTTOPIC_QOS 8

The default TopicQos kind.

5.86.2.156 #define DDSC_LOG_DEFAULTDATAREADER_QOS 9

The default DataReaderQos kind.

5.86.2.157 #define DDSC_LOG_DEFAULTDATAWRITER_QOS 10

The default DataWriterQos kind.

5.86.2.158 #define DDSC_LOG_DEFAULTSUBSCRIBER_QOS 11

The default SubscriberQos kind.

5.86.2.159 #define DDSC_LOG_DEFAULTPUBLISHER_QOS 12

The default PublisherQos kind.

5.86.2.160 #define DDSC_LOG_DEFAULTPARTICIPANT_QOS 13

The default PublisherQos kind.

5.86.2.161 #define DDSC_LOG_DEADLINE_QOS_POLICY 14

The deadline qos policy kind.

5.86.2.162 #define DDSC_LOG_LIVELINESS_QOS_POLICY 15

The liveliness qos policy kind.

5.86.2.163 #define DDSC_LOG_HISTORY_QOS_POLICY 16

The history qos policy kind.

5.86.2.164 #define DDSC_LOG_RESOURCE_LIMIT_QOS_POLICY 17

The resource limits qos policy kind.

5.86.2.165 #define DDSC_LOG_PROTOCOL_QOS_POLICY 18

The protocol qos policy kind.

5.86.2.166 #define DDSC_LOG_TYPE_SUPPORT_QOS_POLICY 19

The type-support qos policy kind.

5.86.2.167 #define DDSC_LOG_RELIABILITY_QOS_POLICY 20

The reliability qos policy kind.

5.86.2.168 #define DDSC_LOG_DURABILITY_QOS_POLICY 21

The durability qos policy kind.

5.86.2.169 #define DDSC_LOG_OWNERSHIP_QOS_POLICY 22

The ownership qos policy kind.

5.86.2.170 #define DDSC_LOG_OWNERSHIP_STRENGTH_QOS_POLICY 23

The ownership-strength qos policy kind.

5.86.2.171 #define DDSC_LOG_TRANSPORT_QOS_POLICY 24

The transport qos policy kind.

5.86.2.172 #define DDSC_LOG_PARTICIPANT_ID_QOS_POLICY 25

The participant id qos policy kind.

5.86.2.173 #define DDSC_LOG_HEARTBEATS_QOS_POLICY 26

The heartbeat qos policy kind.

When configured for reliable communication, heartbeats_per_max_samples must be fit within max_samples

5.86.2.174 #define DDSC_LOG_DATAWRITER_RESOURCE_QOS_POLICY 27

The DataWriterQos writer_resource_limits policy.

An invalid value has been set for a limit of DataWriterQos.writer_resource_limits. Each value must be positive and finite. May be logged by Discovery writers with no initial_peers set.

5.86.2.175 #define DDSC_LOG_DATAREADER_RESOURCE_QOS_POLICY 28

The DataReaderQos reader_resource_limits policy.

5.86.2.176 #define DDSC_LOG_DESTINATION_ORDER_POLICY 29

The DestinationOrderQos destination_order policy.

5.86.2.177 #define DDSC_LOG_USER_TRAFFIC_QOS_POLICY 30

The UserTraffic policy.

5.86.2.178 #define DDSC_LOG_DISCOVERY_TRAFFIC_QOS_POLICY 31

The Discovery.enabled_transport policy.

5.86.2.179 #define DDSC_LOG_QOS_INCONSISTENT_POLICY_EC (DDSC_LOG_BASE + 501)

An inconsistent Qos policy for the specified Qos kind was found.

5.86.2.180 #define DDSC_LOG_QOS_INCONSISTENT_POLICIES_EC (DDSC_LOG_BASE + 502)

Inconsistency between two Qos policies for the specified Qos kind was found.

5.86.2.181 #define DDSC_LOG_QOS_INCONSISTENT_EC (DDSC_LOG_BASE + 503)

Failed to create an entity or set a qos due to inconsistent policy.

5.86.2.182 #define DDSC_LOG_QOS_COPY_EC (DDSC_LOG_BASE + 504)

Failed to copy a Qos of the specified kind.

5.86.2.183 #define DDSC_LOG_QOS_INITIALIZE_EC (DDSC_LOG_BASE + 505)

Failed to initialize a Qos of the specified kind.

5.86.2.184 #define DDSC_LOG_QOS_FINALIZE_EC (DDSC_LOG_BASE + 506)

Failed to finalize a Qos of the specified kind.

5.86.2.185 #define DDSC_LOG_QOS_SET_EC (DDSC_LOG_BASE + 507)

Failed to set a Qos of the specified kind.

5.86.2.186 #define DDSC_LOG_QOS_SET_ON_ENABLED_EC (DDSC_LOG_BASE + 508)

Failed to set a Qos of the specified kind because the entity is already enabled.

5.86.2.187 #define DDSC_LOG_QOS_IMMUTABLE_EC (DDSC_LOG_BASE + 509)

Failed to set a Qos of the specified kind the immutable Qos policies have been changed.

5.86.2.188 #define DDSC_LOG_QOS_CHANGED_EC (DDSC_LOG_BASE + 510)

A discovered Qos changed (the entity already existed)

5.86.2.189 #define DDSC_LOG_QOS_GET_EC (DDSC_LOG_BASE + 511)

Failed to get a Qos of the specified kind.

5.86.2.190 #define DDSC_LOG_TOPIC_LISTENER 1

The topic listener kind.

5.86.2.191 #define DDSC_LOG_DATAREADER_LISTENER 2

The datareader listener kind.

5.86.2.192 #define DDSC_LOG_DATAWRITER_LISTENER 3

The datawriter listener kind.

5.86.2.193 #define DDSC_LOG_SUBSCRIBER_LISTENER 4

The subscriber listener kind.

5.86.2.194 #define DDSC_LOG_PUBLISHER_LISTENER 5

The publisher listener kind.

5.86.2.195 #define DDSC_LOG_PARTICIPANT_LISTENER 6

The participant listener kind.

5.86.2.196 #define DDSC_LOG_PARTICIPANTFACTORY_LISTENER 7

The participant-factory listener kind.

5.86.2.197 #define DDSC_LOG_LISTENER_INCONSISTENT_EC (DDSC_LOG_BASE + 512)

Failed to create an entity due to inconsistent listener and status mask.

5.86.2.198 #define DDSC_LOG_LISTENER_SET_EC (DDSC_LOG_BASE + 513)

Failed to set the listener of the specified kind.

5.86.2.199 #define DDSC_LOG_LISTENER_GET_EC (DDSC_LOG_BASE + 514)

Failed to get the listener of the specified kind.

5.86.2.200 #define DDSC_LOG_LISTENER_SET_ILLEGAL_NULL_EC (DDSC_LOG_BASE + 515)

Illegal combination of NULL listener and non-NONE status mask when setting a listener for an Entity.

5.86.2.201 #define DDSC_LOG_TOPIC_ENTITY 1

The topic entity kind.

5.86.2.202 #define DDSC_LOG_DATAWRITER_ENTITY 2

The datawriter entity kind.

5.86.2.203 #define DDSC_LOG_DATAREADER_ENTITY 3

The datareader entity kind.

5.86.2.204 #define DDSC_LOG_PUBLISHER_ENTITY 4

The publisher entity kind.

5.86.2.205 #define DDSC_LOG_SUBSCRIBER_ENTITY 5

The subscriber entity kind.

5.86.2.206 #define DDSC_LOG_PARTICIPANT_ENTITY 6

The participant entity kind.

5.86.2.207 #define DDSC_LOG_PUBLICATION_ENTITY 7

The publication entity kind.

5.86.2.208 #define DDSC_LOG_SUBSCRIPTION_ENTITY 8

The subscription entity kind.

5.86.2.209 #define DDSC_LOG_ENTITY_ENABLE_EC (DDSC_LOG_BASE + 600)

Failed to enable an entity of the specified kind.

5.86.2.210 #define DDSC_LOG_ENTITY_NOT_EMPTY_EC (DDSC_LOG_BASE + 601)

Failed to an delete/finalize an entity of the specified kind because it is not empty.

5.86.2.211 #define DDSC_LOG_ENTITY_FINALIZE_EC (DDSC_LOG_BASE + 602)

Failed to finalize an entity of the specified kind.

5.86.2.212 #define DDSC_LOG_ENTITY_INITIALIZE_EC (DDSC_LOG_BASE + 603)

Failed to initialize an entity of the specified kind.

5.86.2.213 #define DDSC_LOG_ENTITY_NOT_ENABLED_EC (DDSC_LOG_BASE + 604)

An operation was attempted on an entity that is not enabled.

5.86.2.214 #define DDSC_LOG_ENTITY_DIFFERENT_FACTORY_EC (DDSC_LOG_BASE + 605)

Entities are in different factories.

A entity tried to use an entity created by a different factory. For example, it is illegal to create a datawriter using a topic from a different participant.

5.86.2.215 #define DDSC_LOG_DATAWRITER_CDR 1

The datawriter CDR kind.

5.86.2.216 #define DDSC_LOG_DATAREADER_CDR 2

The datareader CDR kind.

5.86.2.217 #define DDSC_LOG_CDR_POOL_ALLOC_EC (DDSC_LOG_BASE + 700)

Failed to allocate a pool of the specified kind.

5.86.2.218 #define DDSC_LOG_CDR_BUFFER_SET_EC (DDSC_LOG_BASE + 701)

Failed to set the CDR buffer for a packet.

5.86.2.219 #define DDSC_LOG_CDR_POOL_DELETE_EC (DDSC_LOG_BASE + 702)

Failed to delete the CDR pool.

5.86.2.220 #define DDSC_LOG_CDR_SERIALIZE_PID_EC (DDSC_LOG_BASE + 703)

Failed to serialize a parameter ID.

5.86.2.221 #define DDSC_LOG_CDR_SERIALIZE_PID_LENGTH_EC (DDSC_LOG_BASE + 704)

Failed to serialize a parameter length.

5.86.2.222 #define DDSC_LOG_CDR_SERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 705)

Failed to serialize a key-hash.

5.86.2.223 #define DDSC_LOG_CDR_SERIALIZE_DATA_EC (DDSC_LOG_BASE + 706)

Failed to serialize payload data.

5.86.2.224 #define DDSC_LOG_DESERIALIZE_BAD_PID_LENGTH_EC (DDSC_LOG_BASE + 707)

Deserialized an invalid parameter length for a specific parameter ID.

5.86.2.225 #define DDSC_LOG_CDR_DESERIALIZE_PID_EC (DDSC_LOG_BASE + 708)

Failed to deserialize the ID of an inline parameter.

5.86.2.226 #define DDSC_LOG_CDR_DESERIALIZE_PID_LENGTH_EC (DDSC_LOG_BASE + 709)

Failed to deserialize the length of an inline parameter.

5.86.2.227 #define DDSC_LOG_CDR_INCREMENT_POS_EC (DDSC_LOG_BASE + 710)

Failed to increment to the position of the next inline parameter.

5.86.2.228 #define DDSC_LOG_CDR_SET_POS_EC (DDSC_LOG_BASE + 711)

Failed to set the reception stream position.

5.86.2.229 #define DDSC_LOG_CDR_DESERIALIZE_HEADER_EC (DDSC_LOG_BASE + 712)

Failed to deserialize the encapsulation header.

5.86.2.230 #define DDSC_LOG_CDR_DESERIALIZE_DATA_EC (DDSC_LOG_BASE + 713)

Failed to deserialize CDR payload data.

5.86.2.231 #define DDSC_LOG_CDR_INITIALIZE_SAMPLE_EC (DDSC_LOG_BASE + 714)

Failed to initialize CDR sample.

5.86.2.232 #define DDSC_LOG_CDR_FINALIZE_SAMPLE_EC (DDSC_LOG_BASE + 715)

Failed to finalize sample.

5.86.2.233 #define DDSC_LOG_CDR_SERIALIZE_STATUS_INFO_EC (DDSC_LOG_BASE + 716)

Failed to serialize the status info parameter.

5.86.2.234 #define DDSC_LOG_CDR_DESERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 717)

Failed to deserialize a key-hash.

5.86.2.235 #define DDSC_LOG_CDR_DESERIALIZE_KEY_EC (DDSC_LOG_BASE + 718)

Failed to deserialize CDR payload key.

5.86.2.236 #define DDSC_LOG_NETIO_ADD_ANON_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 800)

Failed to add a route to an anonymous participant discovery datawriter.

5.86.2.237 #define DDSC_LOG_NETIO_ADD_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 801)

Failed to add a route to a topic from a datawriter.

5.86.2.238 #define DDSC_LOG_NETIO_DELETE_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 802)

Failed to delete a route to a topic.

5.86.2.239 #define DDSC_LOG_NETIO_FORWARD_TOPIC_EC (DDSC_LOG_BASE + 803)

Failed to forward a topic.

5.86.2.240 #define DDSC_LOG_NETIO_NETIO_KIND 1

Any NETIO interface kind, typically UDP.

5.86.2.241 #define DDSC_LOG_INTRA_NETIO_KIND 2

The intra interface kind.

5.86.2.242 #define DDSC_LOG_RTPS_NETIO_KIND 3

The RTPS interface kind.

5.86.2.243 #define DDSC_LOG_DATAREADER_NETIO_KIND 4

The DataReader interface kind.

5.86.2.244 #define DDSC_LOG_DATAWRITER_NETIO_KIND 5

The DataWriter interface kind.

5.86.2.245 #define DDSC_LOG_NETIO_BIND_EXTERNAL_EC (DDSC_LOG_BASE + 804)

Failed to bind two external interface of the specified kind.

5.86.2.246 #define DDSC_LOG_NETIO_UNBIND_EXTERNAL_EC (DDSC_LOG_BASE + 805)

Failed to unbind two external interfaces of the specified kind.

5.86.2.247 #define DDSC_LOG_NETIO_BIND_EC (DDSC_LOG_BASE + 806)

Failed to bind an interface to a peer interface.

5.86.2.248 #define DDSC_LOG_NETIO_UNBIND_EC (DDSC_LOG_BASE + 807)

Failed to unbind an interface from a peer interface.

5.86.2.249 #define DDSC_LOG_NETIO_ADD_ROUTE_EC (DDSC_LOG_BASE + 808)

Failed to add a route from an interface to a peer interface.

5.86.2.250 #define DDSC_LOG_NETIO_DELETE_ROUTE_EC (DDSC_LOG_BASE + 809)

Failed to delete a route from an interface to a peer interface.

5.86.2.251 #define DDSC_LOG_NETIO_GET_EXTERNAL_INTF_EC (DDSC_LOG_BASE + 810)

Failed to get an external interface for the specified interface kind.

5.86.2.252 #define DDSC_LOG_NETIO_NO_ROUTE_EC (DDSC_LOG_BASE + 811)

A DataReader failed a bind due to no existing route.

5.86.2.253 #define DDSC_LOG_NETIO_ROUTE_LOOKUP_FAILED_EC (DDSC_LOG_BASE + 812)

Lookup a route to a destination failed.

A failure was encountered when trying to lookup a route to a destination, this log message is preceded by a more specific message. Failure to lookup a route does not mean the route does not exist, it means it failed to determine if a route did exist.

5.86.2.254 #define DDSC_LOG_NETIO_GET_ROUTE_TABLE_FAILED_EC (DDSC_LOG_BASE + 813)

Failed to get the route table for an interface.

5.86.2.255 #define DDSC_LOG_NETIO_SEND_FAILED_EC (DDSC_LOG_BASE + 814)

Failure when sending on an interface.

5.86.2.256 #define DDSC_LOG_NETIO_SET_STATE_EC (DDSC_LOG_BASE + 815)

Failed to set an interface state.

5.86.2.257 #define DDSC_LOG_NETIO_PEER_LOOKUP_EC (DDSC_LOG_BASE + 816)

Datawriter did not find a peer.

5.86.2.258 #define DDSC_LOG_NETIO_FORCED_REMOVE_EC (DDSC_LOG_BASE + 817)

Forced removal of sample downstream failed.

The datawriter failed to force a sample removal downstream

5.86.2.259 #define DDSC_LOG_PACKET_INIT_EC (DDSC_LOG_BASE + 818)

Failed to initialize a packet.

5.86.2.260 #define DDSC_LOG_PACKET_SET_HEAD_EC (DDSC_LOG_BASE + 819)

Failed to set the head of a packet.

5.86.2.261 #define DDSC_LOG_PACKET_SET_TAIL_EC (DDSC_LOG_BASE + 820)

Failed to set the tail of a packet.

5.86.2.262 #define DDSC_LOG_NETIO_DISCOVERY_ENABLED_TRANSPORT_EC (DDSC_LOG_BASE + 821)

Configured discovery enabled transport is not valid.

5.86.2.263 #define DDSC_LOG_DW_ACKNACK_FAILED_EC (DDSC_LOG_BASE + 900)

Failed to ACKNACK sample in the writer history.

5.86.2.264 #define DDSC_LOG_DW_COMMIT_EC (DDSC_LOG_BASE + 901)

Failed to commit a sample to the writer queue.

5.86.2.265 #define DDSC_LOG_DW_KEYHASH_CREATE_EC (DDSC_LOG_BASE + 902)

Failed to create keyhash of instance handle.

5.86.2.266 #define DDSC_LOG_DW_ILLEGAL_KEY_KIND_EC (DDSC_LOG_BASE + 903)

Failed a write due to an invalid key kind for the type being written.

5.86.2.267 #define DDSC_LOG_DW_CREATE_TYPED_WRITER_EC (DDSC_LOG_BASE + 904)

Failed to create a typed writer.

5.86.2.268 #define DDSC_LOG_DW_HISTORY_REGISTER_KEY_EC (DDSC_LOG_BASE + 905)

Failed to register the key of an instance.

A DataWriter failed to register the key of a new instance because DataWriterQos.resource_limits.max_instances was exceeded.

5.86.2.269 #define DDSC_LOG_DR_CREATE_TYPED_READER_EC (DDSC_LOG_BASE + 1000)

Failed to create a typed datareader.

5.86.2.270 #define DDSC_LOG_DR_COPY_DATA_SAMPLE_EC (DDSC_LOG_BASE + 1001)

Failed to copy a sample upon reception, read, or take.

5.86.2.271 #define DDSC_LOG_DR_COMMIT_SAMPLE_EC (DDSC_LOG_BASE + 1002)

Failed to commit a sample to be made available to be read or taken.

5.86.2.272 #define DDSC_LOG_DR_FILTER_ERROR_EC (DDSC_LOG_BASE + 1003)

A datareader filter function failed.

5.86.2.273 #define DDSC_LOG_DR_DESERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 1004)

Failed to deserialize a key-hash parameter.

5.86.2.274 #define DDSC_LOG_DR_GET_ENTRY_FAILED_EC (DDSC_LOG_BASE + 1005)

Failed to get a Reader History entry for a received sample.

5.86.2.275 #define DDSC_LOG_DR_COMMIT_ENTRY_EC (DDSC_LOG_BASE + 1006)

Failed to commit a receive sample to Reader History to be read or taken.

5.86.2.276 #define DDSC_LOG_DR_UNREGISTER_KEY_EC (DDSC_LOG_BASE + 1007)

A DataReader failed to unregister an instance.

5.86.2.277 #define DDSC_LOG_DR_DISPOSE_KEY_EC (DDSC_LOG_BASE + 1008)

A DataReader failed to dispose an instance.

5.86.2.278 #define DDSC_LOG_DR_READ_TAKE_FAILURE_EC (DDSC_LOG_BASE + 1009)

A call to a reader/take function failed.

5.86.2.279 #define DDSC_LOG_DR_INSTANCE_TO_KEYHASH_EC (DDSC_LOG_BASE + 1010)

Failed to create keyhash of instance handle.

5.86.2.280 #define DDSC_LOG_TYPE_NAME_CMP_EC (DDSC_LOG_BASE + 1100)

Two type names are incompatible.

5.86.2.281 #define DDSC_LOG_TOPIC_NAME_CMP_EC (DDSC_LOG_BASE + 1101)

Two topic names are incompatible.

5.86.2.282 #define DDSC_LOG_TYPE_FUNCTION_GET_SERIALIZED_SAMPLE_MAX_SIZE 1

The get_serialized_sample_max_size function kind.

5.86.2.283 #define DDSC_LOG_TYPE_FUNCTION_SERIALIZE_DATA 2

The serialize_data function kind.

5.86.2.284 #define DDSC_LOG_TYPE_FUNCTION_DESERIALIZE_DATA 3

The deserialize_data function kind.

5.86.2.285 #define DDSC_LOG_TYPE_FUNCTION_CREATE_SAMPLE 4

The create_sample function kind.

5.86.2.286 #define DDSC_LOG_TYPE_FUNCTION_COPY_SAMPLE 5

The copy_sample function kind.

5.86.2.287 #define DDSC_LOG_TYPE_FUNCTION_DELETE_SAMPLE 6

The delete_sample function kind.

5.86.2.288 #define DDSC_LOG_TYPE_FUNCTION_GET_KEY_KIND 7

The get_key_kind function kind.

5.86.2.289 #define DDSC_LOG_TYPE_FUNCTION_INSTANCE_TO_KEYHASH 8

The instance_to_keyhash function kind.

5.86.2.290 #define DDSC_LOG_TYPE_FUNCTION_NULL_EC (DDSC_LOG_BASE + 1102)

Invalid type plugin, The specified function pointer is NULL.

5.86.2.291 #define DDSC_LOG_TOPIC_TOO_LONG_EC (DDSC_LOG_BASE + 1103)

Failed to create a topic because the name exceeded the maximum length of 255 octets (excluding the terminating NUL)

5.86.2.292 #define DDSC_LOG_TYPE_TOO_LONG_EC (DDSC_LOG_BASE + 1104)

Failed to create a type because the name exceeded the maximum length of 255 octets (excluding the terminating NUL)

5.86.2.293 #define DDSC_LOG_LOOKUP_TYPE_PLUGIN_EC (DDSC_LOG_BASE + 1105)

A type-plugin for the given type could not be found.

5.86.2.294 #define DDSC_LOG_TYPE_KEY_TYPE_EC (DDSC_LOG_BASE + 1106)

Two types have incompatible keys.

5.86.2.295 #define DDSC_LOG_TYPE_MANAGED_POOL_FAILURE_EC (DDSC_LOG_BASE + 1107)

Managed pool function failed.

5.86.2.296 #define DDSC_LOG_TYPE_GET_SAMPLE_ID_EC (DDSC_LOG_BASE + 1108)

Managed pool failed to get sample id.

5.86.2.297 #define DDSC_LOG_DISC_LOCAL_PARTICIPANT_ENABLED_EC (DDSC_LOG_BASE + 1200)

Discovery plugin failed its update after a local DomainParticipant was enabled.

5.86.2.298 #define DDSC_LOG_DISC_BEFORE_LOCAL_PARTICIPANT_CREATED_EC (DDSC_LOG_BASE + 1201)

Discovery plugin failed its update before a local DomainParticipant was created.

5.86.2.299 #define DDSC_LOG_DISC_AFTER_LOCAL_PARTICIPANT_CREATED_EC (DDSC_LOG_BASE + 1202)

Discovery plugin failed its update after a local DomainParticipant was created.

5.86.2.300 #define DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_ENABLED_EC (DDSC_LOG_BASE + 1203)

Discovery plugin failed its update after a local DataReader was enabled.

5.86.2.301 #define DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_DELETED_EC (DDSC_LOG_BASE + 1204)

Discovery plugin failed its update after a local DataReader was deleted.

5.86.2.302 #define DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_ENABLED_EC (DDSC_LOG_BASE + 1205)

Discovery plugin failed its update after a local DataWriter was enabled.

5.86.2.303 #define DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_DELETED_EC (DDSC_LOG_BASE + 1206)

Discovery plugin failed its update after a local DataWriter was deleted.

5.86.2.304 #define DDSC_LOG_DISC_REMOTE_PARTICIPANT_EXPIRED_EC (DDSC_LOG_BASE + 1207)

Discovery plugin failed its update after a remote DomainParticipant's liveliness expired.

5.86.2.305 #define DDSC_LOG_DISC_ADD_PEER_EC (DDSC_LOG_BASE + 1208)

Failed to add a peer with discovery plugin.

5.86.2.306 #define DDSC_LOG_CHECKSUM_REQUIRED_EC (DDSC_LOG_BASE + 1301)

A participant requires checksum, but the discovered participant does not send a checksum.

5.86.2.307 #define DDSC_LOG_CHECKSUM_INCOMPATIBLE_EC (DDSC_LOG_BASE + 1302)

A participant does not support checksum's sent by a discovered participant.

5.86.2.308 #define DDSC_LOG_CHECKSUM_INCONSISTENT_CLASS_EC (DDSC_LOG_BASE + 1303)

The checksum class id for a custom checksum does match.

5.86.2.309 #define DDSC_LOG_CHECKSUM_INCONSISTENT_COMPUTE_EC (DDSC_LOG_BASE + 1304)

The computed_crc_kind is inconsistent with what is allowed or supported.

5.86.2.310 #define DDSC_LOG_CHECKSUM_INCONSISTENT_ALLOW_EC (DDSC_LOG_BASE + 1305)

The allowed_crc_mask is inconsistent with what is supported.

5.87 Sequence Support

Defines sequence interface and primitive data types sequences.

Data Structures

- struct FooSeq

  <<interface>> <<generic>> <<cert>> A type-safe, ordered collection of elements. The type of these elements is referred to in this documentation as "`Foo`".

Macros

- #define DDS_SEQUENCE_INITIALIZER

  An initializer for new sequence instances.

Functions

- DDS_Boolean FooSeq_initialize (struct FooSeq *self)

  <<cert>> Initialize sequence instances.
- DDS_Long FooSeq_get_maximum (const struct FooSeq *self)

  <<cert>> Get the current maximum number of elements that can be stored in this sequence.
- DDS_Boolean FooSeq_set_maximum (struct FooSeq *self, DDS_Long new_max)

  <<cert>> Resize this sequence to a new desired maximum.
- DDS_Boolean FooSeq_set_maximum_w_max (struct FooSeq *self, DDS_Long new_max, DDS_UnsignedLong max_size)

  <<cert>> Resize this sequence to a new desired maximum and allocate `max_size` + 1 bytes for each pointer element.
- DDS_Long FooSeq_get_length (const struct FooSeq *self)

  <<cert>> Get the logical length of this sequence.
- DDS_Boolean FooSeq_set_length (struct FooSeq *self, DDS_Long new_length)

  <<cert>> Change the length of this sequence.
- Foo * FooSeq_get_reference (const struct FooSeq *self, DDS_Long i)

  <<cert>> Get the pointer to the `i-th` element of this sequence.
- struct FooSeq * FooSeq_copy (struct FooSeq *self, const struct FooSeq *src_seq)

  Copy elements from another sequence, resizing the sequence if necessary.
- struct FooSeq * FooSeq_copy_w_max (struct FooSeq *self, const struct FooSeq *src_seq, DDS_UnsignedLong max_size)

  Copy elements from another sequence, resizing the sequence if necessary.

5.87.1 Detailed Description

Defines sequence interface and primitive data types sequences.

5.87.2 Macro Definition Documentation

5.87.2.1 #define DDS_SEQUENCE_INITIALIZER

An initializer for new sequence instances.

This constant will initialize a new sequence to a valid empty state. C language users should assign it to uninitialized sequence instances before using them, at the time they are declared, or use FooSeq_initialize as an alternative function to initialize new sequences.

5.87.3 Function Documentation

5.87.3.1 DDS_Boolean FooSeq_initialize (struct FooSeq * *self*)

<<*cert*>> Initialize sequence instances.

Use this function to initialize new sequences to a valid emptu state. C users should initialize sequences before using them.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

DDS_BOOLEAN_TRUE if the sequence was successfully initialized; DDS_BOOLEAN_FALSE otherwise.

MT Safety:

This operation is not thread safe.

See also

DDS_SEQUENCE_INITIALIZER

5.87.3.2 DDS_Long FooSeq_get_maximum (const struct FooSeq * *self*)

<<*cert*>> Get the current maximum number of elements that can be stored in this sequence.

The `maximum` of the sequence represents the maximum number of elements that the underlying buffer can hold. It does not represent the current number of elements.

The `maximum` is a non-negative number. It is initialized when the sequence is first created and can only be set by means of the FooSeq_set_maximum operation.

Note that the `maximum` can only be set once, and cannot be changed after FooSeq_set_maximum has been called for the first time.

Parameters

self	<<*in*>> Cannot be NULL.

Returns

the current maximum of the sequence.

MT Safety:

This operation is not thread safe.

See also

FooSeq_get_length

5.87.3.3 DDS_Boolean FooSeq_set_maximum (struct FooSeq * *self*, DDS_Long *new_max*)

<<*cert*>> Resize this sequence to a new desired maximum.

This operation does nothing if the new desired maximum matches the current `maximum`.

This operation can only allocate memory for a buffer once, and will fail if the existing `maximum` is not zero.

Precondition

maximum == 0 or maximum == new_max
owned == DDS_BOOLEAN_TRUE

Postcondition

owned == DDS_BOOLEAN_TRUE
length == MINIMUM(original length, new_max)

Parameters

self	*<<in>>* Cannot be NULL.
new_max	Must be >= 0.

Returns

DDS_BOOLEAN_TRUE on success, DDS_BOOLEAN_FALSE if the preconditions are not met. In that case the sequence is not modified.

MT Safety:

This operation is not thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

Note

This operation is not available for sequences of type DDS_WString. The operation named `DDS_WString_Seq_set_maximum` is an implementation of FooSeq_set_maximum_w_max.

5.87.3.4 DDS_Boolean FooSeq_set_maximum_w_max (struct FooSeq * *self*, DDS_Long *new_max*, DDS_UnsignedLong *max_size*)

<<cert>> Resize this sequence to a new desired maximum and allocate `max_size` + 1 bytes for each pointer element.

This operation does nothing if the new desired maximum matches the current `maximum`.

This operation can only allocate memory for a buffer once, and will fail if the existing `maximum` is not zero.

Precondition

maximum == 0 or maximum == new_max
owned == DDS_BOOLEAN_TRUE

Postcondition

owned == DDS_BOOLEAN_TRUE
length == MINIMUM(original length, new_max)

Parameters

self	*<<in>>* Cannot be NULL.
new_max	Must be >= 0.
max_size	*<<in>>* Each pointer element is allocated to to `max_size` + 1 bytes.

Returns

DDS_BOOLEAN_TRUE on success, DDS_BOOLEAN_FALSE if the preconditions are not met. In that case the sequence is not modified.

MT Safety:

This operation is not thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

Note

This operation is only available for sequences of types DDS_String and DDS_Wstring. Note that for wide strings, this operation is named `DDS_WStringSeq_set_maximum`.

5.87.3.5 DDS_Long FooSeq_get_length (const struct FooSeq * *self*)

<<cert>> Get the logical length of this sequence.

Get the length that was last set, or zero if the length has never been set.

Parameters

self	*<<in>>* Cannot be NULL.

Returns

the length of the sequence

MT Safety:

This operation is not thread safe.

5.87.3.6 DDS_Boolean FooSeq_set_length (struct FooSeq * *self*, DDS_Long *new_length*)

<<cert>> Change the length of this sequence.

This function does not allocate/deallocate memory.

The new length must not exceed the maximum of this sequence as returned by the FooSeq_get_maximum operation. (Note that, if necessary, the maximum of this sequence can be increased manually by using the FooSeq_set_maximum operation.)

The elements of the sequence are not modified by this operation. If the new length is larger than the original length, the new elements will be uninitialized; if the length is decreased, the old elements that are beyond the new length will physically remain in the sequence but will not be accessible.

Postcondition

length = new_length.

Parameters

self	*<<in>>* Cannot be NULL.
new_length	the new desired length. This value must be non-negative and cannot exceed maximum of the sequence. In other words 0 <= new_length <= maximum

Returns

DDS_BOOLEAN_TRUE on sucess or DDS_BOOLEAN_FALSE on failure

MT Safety:

This operation is not thread safe.

5.87.3.7 Foo* FooSeq_get_reference (const struct FooSeq * *self*, DDS_Long *i*)

<<*cert*>> Get the pointer to the `i-th` element of this sequence.

Parameters

self	<<*in*>> Cannot be NULL.
i	index of element to access, must be >= 0 and less than FooSeq_get_length

Returns

the `i-th` element

MT Safety:

This operation is not thread safe.

5.87.3.8 struct FooSeq* FooSeq_copy (struct FooSeq * *self*, const struct FooSeq * *src_seq*) [read]

Copy elements from another sequence, resizing the sequence if necessary.

This function invokes copies the content of the specified sequence, after ensuring that this sequence has enough capacity to hold the elements to be copied.

It is important to know that no check for overlapping memory regions is performed, and copying sequences with overlapping memory regions has undefined behavior. For example, copying two string sequences where the strings are overlapping has undefined behavior.

RTI Connext Micro does not create overlapping memory regions.

NOTE: If this function is called on a sequence that has been created with set_maxmium_w_max, it will automatically call copy_w_max().

Parameters

self	<<*in*>> Cannot be NULL.
src_seq	<<*in*>> the sequence from which to copy

Returns

`self`, this sequence

MT Safety:

This operation is not thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

Note

This operation is not available for sequences of type DDS_Wstring. The operation named `DDS_WString-Seq_copy` is an implementation of FooSeq_copy_w_max.

5.87.3.9 struct FooSeq* FooSeq_copy_w_max (struct FooSeq * *self*, const struct FooSeq * *src_seq*, DDS_UnsignedLong *max_size*) [read]

Copy elements from another sequence, resizing the sequence if necessary.

This function invokes copies the content of the specified sequence, after ensuring that this sequence has enough capacity to hold the elements to be copied.

Parameters

self	<<*in*>> Cannot be NULL.
src_seq	<<*in*>> the sequence from which to copy
max_size	<<*in*>> Each element is a pointer to `max_size` + 1 bytes.

Returns

`self`, this sequence

MT Safety:

This operation is not thread safe.

Note

This operation is only available for sequences of types DDS_String and DDS_Wstring. Note that for wide strings, this operation is named `DDS_WStringSeq_copy`.

5.88 String Support

String creation, cloning, assignment, and deletion.

Functions

- char * DDS_String_alloc (DDS_UnsignedLong length)

  Create a new empty string that can hold up to `length` characters.
- char * DDS_String_dup (const char *str)

  Clone a string. Creates a new string that duplicates the value of `string`.
- int DDS_String_cmp (const char *s1, const char *s2)

  Compare two strings.
- int DDS_String_ncmp (const char *s1, const char *s2, DDS_UnsignedLong num)

  Compare two strings.
- int DDS_String_length (const char *string)

  Return the length of a string.
- int DDS_Wstring_cmp (const CDR_Wstring *s1, const CDR_Wstring *s2)

  Return the length of a string.

5.88.1 Detailed Description

String creation, cloning, assignment, and deletion. The functions in this class ensure consistent cross-platform implementations for string creation (DDS_String_alloc()), deletion (DDS_String_free()), and cloning (DDS_String_dup()) that preserve the mutable value type semantics. These are to be viewed as functions that define a `string` class whose data is represented by a `'char*'`.

The following conventions govern the memory management of strings in RTI Connext Micro:

- The DDS implementation ensures that when value types containing strings are passed back and forth to the DDS APIs, the strings are created/deleted/assigned/cloned using the `string` class functions.
- Value types containing strings have ownership of the contained string. Thus, when a value type is deleted, the contained string field is also deleted.
- The user must ensure that when value types containing strings are passed back and forth to the DDS APIs, the strings are created/deleted/assigned/cloned using the String class functions.

The representation of a string in C/C++ unfortunately does not allow programs to detect how much memory has been allocated for a string. RTI Connext Micro must therefore make some assumptions when a user requests that a string be copied into. The following rules apply when RTI Connext Micro is copying into a string or string sequence:

- If the 'char*' is NULL, RTI Connext Micro will log a warning and allocate a new string on behalf of the user. To avoid leaking memory, you must ensure that the string will be freed (see below).
- If the 'char*' is not NULL, RTI Connext Micro will assume that you are managing the string's memory yourself and have allocated enough memory to store the string to be copied. RTI Connext Micro will copy into your memory; to avoid memory corruption, be sure to allocate enough of it. Also, do not pass structures containing junk pointers into RTI Connext Micro; you are likely to crash.

This requirement can generally be assured by adhering to the following *idiom* for manipulating strings:

- Always use DDS_String_alloc() to create a string
- Always use DDS_String_dup() to clone a string

5.88.2 Function Documentation

5.88.2.1 char* DDS_String_alloc (DDS_UnsignedLong *length*)

Create a new empty string that can hold up to `length` characters.

This function will allocate enough memory to hold a string of `length` characters, **plus** one additional byte to hold the NULL terminating character.

Parameters

length	<<*in*>> Capacity of the string.

Returns

A newly created non-NULL string upon success or NULL upon failure.

MT Safety:

This operation is thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.88.2.2 char* DDS_String_dup (const char * *str*)

Clone a string. Creates a new string that duplicates the value of `string`.

Parameters

str	<<*in*>> The string to duplicate.

Returns

If `string` == NULL, this function always returns NULL. Otherwise, upon success it returns a newly created string whose value is `string`; upon failure it returns NULL.

MT Safety:

This operation is not thread safe.

API Restriction:

This function *must* only be called after DDS_DomainParticipantFactory_get_instance.

5.88.2.3 int DDS_String_cmp (const char * *s1*, const char * *s2*)

Compare two strings.

Precondition

`s1` and s2 can be NULL or non-NULL.

Parameters

s1	<<*in*>> String 1 to compare.
s2	<<*in*>> String 2 to compare.

Returns

0 if s1 equals s2, positive integer if s1 is greater than s2, negative integer if s1 is less than s2

MT Safety:

This operation is not thread safe.

5.88.2.4 int DDS_String_ncmp (const char * *s1*, const char * *s2*, DDS_UnsignedLong *num*)

Compare two strings.

This function compares two strings for lexicographic equivalence, but only up to `num` bytes.

Parameters

`in`	*s1*	<<*in*>> Left side of comparison
`in`	*s2*	<<*in*>> Right side of comparison
`in`	*num*	Maximum number of bytes to compare

Returns

0 if s1 equals s2, positive integer if s1 is greater than s2, negative integer if s1 is less than s2

MT Safety:

This operation is not thread safe.

5.88.2.5 int DDS_String_length (const char * *string*)

Return the length of a string.

Precondition

`string` must be non-NULL

Parameters

string	<<*in*>> String to return length of

Returns

length of string. Does not include the NUL-termination character. If a NULL value is passed as argument, RTI_SIZE_INVALID will be returned.

MT Safety:

This operation is not thread safe.

5.88.2.6 int DDS_Wstring_cmp (const CDR_Wstring * *s1*, const CDR_Wstring * *s2*)

Return the length of a string.

Precondition

`s1` and s2 can be NULL or non-NULL.

Parameters

s1	<<*in*>> String 1 to compare.
s2	<<*in*>> String 2 to compare.

Returns

0 if s1 equals s2, positive integer if s1 is greater than s2, negative integer if s1 is less than s2

MT Safety:

This operation is not thread safe.

5.89 RH

Reader History. ModuleID = 8.

Macros

- #define RHSM_LOG_OUTSTANDING_SAMPLE_EC (RHSM_LOG_BASE + 1)

  Failed to delete Reader History due to outstanding, unremoved sample.
- #define RHSM_LOG_NO_PROPERTY_QOS_EC (RHSM_LOG_BASE + 2)

  Failed to create Reader History due to unspecified property Qos.
- #define RHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 3)

  KEEP_ALL History kind is not supported.
- #define RHSM_LOG_MAX_SAMPLE_TOO_SMALL_EC (RHSM_LOG_BASE + 4)

  DataReaderQos.resource_limits.max_samples set too small.
- #define RHSM_LOG_TBF_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 5)

  Time-based filtering is not supported.
- #define RHSM_LOG_SAMPLE_INFO_POOL_EC (RHSM_LOG_BASE + 6)

  Failed to allocate a buffer pool for sample info.
- #define RHSM_LOG_SAMPLE_POOL_EC (RHSM_LOG_BASE + 7)

  Failed to allocate a buffer pool for sample pointers.
- #define RHSM_LOG_SAMPLE_PTR_ARRAY_EC (RHSM_LOG_BASE + 8)

  Failed to get sample buffer.
- #define RHSM_LOG_INFO_ARRAY_EC (RHSM_LOG_BASE + 9)

  Failed to get sample info buffer.
- #define RHSM_LOG_GET_RECEPTION_TIMESTAMP_EC (RHSM_LOG_BASE + 10)

  Failed to get time for reception timestamp.
- #define RHSM_LOG_INVALID_INSTANCE_REPLACEMENT_EC (RHSM_LOG_BASE + 11)

  An invalid instance replacement policy was specified.
- #define RHSM_LOG_FAILED_TO_REMOVE_OLDEST_EC (RHSM_LOG_BASE + 12)

  Failed to remove the oldest sample.
- #define RHSM_LOG_SAMPLE_POOL_EMPTY_EC (RHSM_LOG_BASE + 13)

  Sample pool empty, may have exceeded.
- #define RHSM_LOG_RW_PRUNE_FAILED_EC (RHSM_LOG_BASE + 14)

  Failed to prune remote writer entry.
- #define RHSM_LOG_ENTRY_RESERVATION_FAILED_EC (RHSM_LOG_BASE + 15)

  Failed to reserve an entry.
- #define RHSM_LOG_RETURN_SAMPLE_EC (RHSM_LOG_BASE + 16)

  Failed to return a sample.
- #define RHSM_LOG_HISTORY_UPDATE_EC (RHSM_LOG_BASE + 17)

  Failed to update the history queue.
- #define RHSM_LOG_GET_TIME_EC (RHSM_LOG_BASE + 18)

  Failed to get the system time.
- #define RHSM_LOG_OBJECT_ALLOCATE_EC (RHSM_LOG_BASE + 19)

  Failed to allocate object of the specified kind.
- #define RHSM_LOG_OBJECT_DELETE_EC (RHSM_LOG_BASE + 20)

  Failed to delete object of the specified kind.
- #define RHSM_LOG_OBJECT_INDEX_EC (RHSM_LOG_BASE + 21)

  Failed to index an object of the specified kind.
- #define RHSM_LOG_OBJECT_INVALID_EC (RHSM_LOG_BASE + 22)

  Invalid object specified.
- #define RHSM_LOG_NO_PROPERTY_EC (RHSM_LOG_BASE + 23)

No property when creating a reader history instance.

- #define RHSM_LOG_DESTINATION_BY_SOURCE_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 24)

  Destination ordering by SOURCE_TIMESTAMP ordering is not supported.

- #define RHSM_LOG_RECOMMIT_FAILURE_EC (RHSM_LOG_BASE + 25)

  Failed to recommit sample.

5.89.1 Detailed Description

Reader History. ModuleID = 8.

5.89.2 Macro Definition Documentation

5.89.2.1 #define RHSM_LOG_OUTSTANDING_SAMPLE_EC (RHSM_LOG_BASE + 1)

Failed to delete Reader History due to outstanding, unremoved sample.

5.89.2.2 #define RHSM_LOG_NO_PROPERTY_QOS_EC (RHSM_LOG_BASE + 2)

Failed to create Reader History due to unspecified property Qos.

5.89.2.3 #define RHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 3)

KEEP_ALL History kind is not supported.

5.89.2.4 #define RHSM_LOG_MAX_SAMPLE_TOO_SMALL_EC (RHSM_LOG_BASE + 4)

DataReaderQos.resource_limits.max_samples set too small.

DataReaderQos.resource_limits.max_samples too small, must be at least DataReaderQos.resource_limits.max_instances ∗ DataReaderQos.resource_limits.max_samples_per_instance

5.89.2.5 #define RHSM_LOG_TBF_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 5)

Time-based filtering is not supported.

5.89.2.6 #define RHSM_LOG_SAMPLE_INFO_POOL_EC (RHSM_LOG_BASE + 6)

Failed to allocate a buffer pool for sample info.

5.89.2.7 #define RHSM_LOG_SAMPLE_POOL_EC (RHSM_LOG_BASE + 7)

Failed to allocate a buffer pool for sample pointers.

5.89.2.8 #define RHSM_LOG_SAMPLE_PTR_ARRAY_EC (RHSM_LOG_BASE + 8)

Failed to get sample buffer.

May have exceeded DataReaderQos.reader_resource_limits.max_outstanding_reads

5.89.2.9 #define RHSM_LOG_INFO_ARRAY_EC (RHSM_LOG_BASE + 9)

Failed to get sample info buffer.

May have exceeded DataReaderQos.reader_resource_limits.max_outstanding_reads

5.89.2.10 #define RHSM_LOG_GET_RECEPTION_TIMESTAMP_EC (RHSM_LOG_BASE + 10)

Failed to get time for reception timestamp.

5.89.2.11 #define RHSM_LOG_INVALID_INSTANCE_REPLACEMENT_EC (RHSM_LOG_BASE + 11)

An invalid instance replacement policy was specified.

5.89.2.12 #define RHSM_LOG_FAILED_TO_REMOVE_OLDEST_EC (RHSM_LOG_BASE + 12)

Failed to remove the oldest sample.

5.89.2.13 #define RHSM_LOG_SAMPLE_POOL_EMPTY_EC (RHSM_LOG_BASE + 13)

Sample pool empty, may have exceeded.

The sample pool is empty, may have exceeded DataReaderQos.resource_limits.max_samples

5.89.2.14 #define RHSM_LOG_RW_PRUNE_FAILED_EC (RHSM_LOG_BASE + 14)

Failed to prune remote writer entry.

5.89.2.15 #define RHSM_LOG_ENTRY_RESERVATION_FAILED_EC (RHSM_LOG_BASE + 15)

Failed to reserve an entry.

An entry reservation failed beyond normal resource settings

5.89.2.16 #define RHSM_LOG_RETURN_SAMPLE_EC (RHSM_LOG_BASE + 16)

Failed to return a sample.

5.89.2.17 #define RHSM_LOG_HISTORY_UPDATE_EC (RHSM_LOG_BASE + 17)

Failed to update the history queue.

5.89.2.18 #define RHSM_LOG_GET_TIME_EC (RHSM_LOG_BASE + 18)

Failed to get the system time.

5.89.2.19 #define RHSM_LOG_OBJECT_ALLOCATE_EC (RHSM_LOG_BASE + 19)

Failed to allocate object of the specified kind.

5.89.2.20 #define RHSM_LOG_OBJECT_DELETE_EC (RHSM_LOG_BASE + 20)

Failed to delete object of the specified kind.

5.89.2.21 #define RHSM_LOG_OBJECT_INDEX_EC (RHSM_LOG_BASE + 21)

Failed to index an object of the specified kind.

5.89.2.22 #define RHSM_LOG_OBJECT_INVALID_EC (RHSM_LOG_BASE + 22)

Invalid object specified.

5.89.2.23 #define RHSM_LOG_NO_PROPERTY_EC (RHSM_LOG_BASE + 23)

No property when creating a reader history instance.

5.89.2.24 #define RHSM_LOG_DESTINATION_BY_SOURCE_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 24)

Destination ordering by SOURCE_TIMESTAMP ordering is not supported.

5.89.2.25 #define RHSM_LOG_RECOMMIT_FAILURE_EC (RHSM_LOG_BASE + 25)

Failed to recommit sample.

5.90 WH

Writer History. ModuleID = 9.

Macros

- #define WHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (WHSM_LOG_BASE + 1)
  KEEP_ALL History kind is not supported.
- #define WHSM_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC (WHSM_LOG_BASE + 2)
  Unlimited length resource limits unsupported.
- #define WHSM_LOG_MAX_SAMPLES_TOO_SMALL_EC (WHSM_LOG_BASE + 3)
  DataWriterQos.resource_limits.max_samples set too small.
- #define WHSM_LOG_OBJECT_ALLOCATE_EC (WHSM_LOG_BASE + 4)
  Failed to allocate object of the specified kind.
- #define WHSM_LOG_OBJECT_DELETE_EC (WHSM_LOG_BASE + 5)
  Failed to delete object of the specified kind.
- #define WHSM_LOG_OBJECT_INDEX_EC (WHSM_LOG_BASE + 6)
  Failed to index an object of the specified kind.
- #define WHSM_LOG_NO_PROPERTY_EC (WHSM_LOG_BASE + 7)
  No property when creating a writer history instance.
- #define WHSM_LOG_INVALID_INDEX_OBJECT_EC (WHSM_LOG_BASE + 8)
  The sample removed from an index did not match sample's key.

5.90.1 Detailed Description

Writer History. ModuleID = 9.

5.90.2 Macro Definition Documentation

5.90.2.1 #define WHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (WHSM_LOG_BASE + 1)

KEEP_ALL History kind is not supported.

5.90.2.2 #define WHSM_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC (WHSM_LOG_BASE + 2)

Unlimited length resource limits unsupported.

Ensure that max_samples, max_samples_per_instance, and max_instances resource limits are all finite values

5.90.2.3 #define WHSM_LOG_MAX_SAMPLES_TOO_SMALL_EC (WHSM_LOG_BASE + 3)

DataWriterQos.resource_limits.max_samples set too small.

DataWriterQos.resource_limits.max_samples must be no less than max_instances $*$ max_samples_per_instance

5.90.2.4 #define WHSM_LOG_OBJECT_ALLOCATE_EC (WHSM_LOG_BASE + 4)

Failed to allocate object of the specified kind.

5.90.2.5 #define WHSM_LOG_OBJECT_DELETE_EC (WHSM_LOG_BASE + 5)

Failed to delete object of the specified kind.

5.90.2.6 #define WHSM_LOG_OBJECT_INDEX_EC (WHSM_LOG_BASE + 6)

Failed to index an object of the specified kind.

5.90.2.7 #define WHSM_LOG_NO_PROPERTY_EC (WHSM_LOG_BASE + 7)

No property when creating a writer history instance.

5.90.2.8 #define WHSM_LOG_INVALID_INDEX_OBJECT_EC (WHSM_LOG_BASE + 8)

The sample removed from an index did not match sample's key.

5.91 DPSE

DPSE. ModuleID = 10.

Macros

- #define DPSE_LOG_INVALID_PEER_ADDRESS_EC (DPSE_LOG_BASE + 1)

  A peer address string specifies an invalid address.
- #define DPSE_LOG_CREATE_DISCOVERY_PUBLISHER_EC (DPSE_LOG_BASE + 2)

  Failed to create the publisher for a participant discovery writer.
- #define DPSE_LOG_CREATE_DISCOVERY_SUBSCRIBER_EC (DPSE_LOG_BASE + 3)

  Failed to create the subscriber for a participant discovery datareader.
- #define DPSE_LOG_REGISTER_TYPE_EC (DPSE_LOG_BASE + 4)

  Failed to register a built-in participant discovery type.
- #define DPSE_LOG_CREATE_TOPIC_EC (DPSE_LOG_BASE + 5)

  Failed to create a topic for the built-in participant discovery topic.
- #define DPSE_LOG_CREATE_WRITER_EC (DPSE_LOG_BASE + 6)

  Failed to create a DataWriter for participant discovery.
- #define DPSE_LOG_CREATE_READER_EC (DPSE_LOG_BASE + 7)

  Failed to create a DataReader for participant discovery.
- #define DPSE_LOG_DISPOSE_EC (DPSE_LOG_BASE + 8)

  Failed to dispose a participant discovery instance.
- #define DPSE_LOG_ANNOUNCE_LOCAL_PARTICIPANT_DELETION_EC (DPSE_LOG_BASE + 9)

  Failed to dispose a participant discovery instance upon deletion.
- #define DPSE_LOG_DELETE_WRITER_EC (DPSE_LOG_BASE + 10)

  Failed to delete a participant discovery DataWriter.
- #define DPSE_LOG_DELETE_PUBLISHER_EC (DPSE_LOG_BASE + 11)

  Failed to delete a participant discovery Publisher.
- #define DPSE_LOG_DELETE_READER_EC (DPSE_LOG_BASE + 12)

  Failed to delete a participant discovery DataReader.
- #define DPSE_LOG_DELETE_TOPIC_EC (DPSE_LOG_BASE + 13)

  Failed to delete a participant discovery Topic.
- #define DPSE_LOG_DELETE_SUBSCRIBER_EC (DPSE_LOG_BASE + 14)

  Failed to delete a participant discovery Subscriber.
- #define DPSE_LOG_OBJECT_ALLOCATE_EC (DPSE_LOG_BASE + 15)

  Failed to allocate an object of the specified kind.
- #define DPSE_LOG_OBJECT_INITIALIZE_EC (DPSE_LOG_BASE + 16)

  Failed to initialize an object of the specified kind.
- #define DPSE_LOG_OBJECT_FINALIZE_EC (DPSE_LOG_BASE + 17)

  Failed to finalize an object of the specified kind.
- #define DPSE_LOG_OBJECT_DELETE_EC (DPSE_LOG_BASE + 18)

  Failed to delete an object of the specified kind.
- #define DPSE_LOG_OBJECT_INVALID_EC (DPSE_LOG_BASE + 19)

  The object was invalid in the context it was used.
- #define DPSE_LOG_CDR_SET_POSITION_EC (DPSE_LOG_BASE + 20)

  Failed to set the CDR stream position.
- #define DPSE_LOG_CDR_SERIALIZE_EC (DPSE_LOG_BASE + 21)

  Failed to serialize the specified kind.
- #define DPSE_LOG_CDR_DESERIALIZE_EC (DPSE_LOG_BASE + 22)

  Failed to deserialize the specified kind.
- #define DPSE_LOG_SERIALIZE_GUID_EC (DPSE_LOG_BASE + 23)

Failed to serialize a GUID key parameter.

- #define DPSE_LOG_SERIALIZE_BUILTIN_ENDPOINTS_EC (DPSE_LOG_BASE + 24)

  Failed to serialize a Builtin Endpoint Mask parameter.
- #define DPSE_LOG_SERIALIZE_PROTOCOL_VERSION_EC (DPSE_LOG_BASE + 25)

  Failed to serialize a Protocol Version parameter.
- #define DPSE_LOG_SERIALIZE_VENDOR_ID_EC (DPSE_LOG_BASE + 26)

  Failed to serialize a Vendor ID parameter.
- #define DPSE_LOG_SERIALIZE_DEFAULT_UNICAST_EC (DPSE_LOG_BASE + 27)

  Failed to serialize a Default Unicast Locator parameter.
- #define DPSE_LOG_SERIALIZE_META_UNICAST_EC (DPSE_LOG_BASE + 28)

  Failed to serialize a Meta Unicast Locator parameter.
- #define DPSE_LOG_SERIALIZE_META_MULTICAST_EC (DPSE_LOG_BASE + 29)

  Failed to serialize a Meta Multicast Locator parameter.
- #define DPSE_LOG_SERIALIZE_LEASE_DURATION_EC (DPSE_LOG_BASE + 30)

  Failed to serialize a Lease Duration parameter.
- #define DPSE_LOG_SERIALIZE_PRODUCT_VERSION_EC (DPSE_LOG_BASE + 31)

  Failed to serialize a Product Version parameter.
- #define DPSE_LOG_DESERIALIZE_PRODUCT_VERSION_EC (DPSE_LOG_BASE + 32)

  Failed to serialize a Product Version parameter.
- #define DPSE_LOG_SERIALIZE_PARTICIPANT_NAME_EC (DPSE_LOG_BASE + 33)

  Failed to serialize a Participant Name parameter.
- #define DPSE_LOG_DESERIALIZE_GUID_EC (DPSE_LOG_BASE + 34)

  Failed to deserialize a GUID key parameter.
- #define DPSE_LOG_DESERIALIZE_BUILTIN_ENDPOINTS_EC (DPSE_LOG_BASE + 35)

  Failed to deserialize a Builtin Endpoint Mask parameter.
- #define DPSE_LOG_DESERIALIZE_PROTOCOL_VERSION_EC (DPSE_LOG_BASE + 36)

  Failed to deserialize a Protocol Version parameter.
- #define DPSE_LOG_DESERIALIZE_VENDOR_ID_EC (DPSE_LOG_BASE + 37)

  Failed to deserialize a Vendor ID parameter.
- #define DPSE_LOG_DESERIALIZE_TOO_MANY_LOCATORS_EC (DPSE_LOG_BASE + 38)

  Cannot deserialize another locator parameter, having reached maximum of 4 unicast or 4 multicast locators.
- #define DPSE_LOG_DESERIALIZE_PARTICIPANT_NAME_EC (DPSE_LOG_BASE + 39)

  Failed to deserialize a Participant Name parameter.
- #define DPSE_LOG_DESERIALIZE_UNKNOWN_PID_EC (DPSE_LOG_BASE + 40)

  Failed to deserialize a parameter with an unknown ID.
- #define DPSE_LOG_ANNOUNCEMENT_EC (DPSE_LOG_BASE + 41)

  Failed to send a participant discovery announcement.
- #define DPSE_LOG_UPDATE_PARTICIPANT_ASSERT_PERIOD_EC (DPSE_LOG_BASE + 42)

  Failed to schedule an event to send the next participant discovery announcement.
- #define DPSE_LOG_ADVANCE_SN_EC (DPSE_LOG_BASE + 43)

  Failed to advance the sequence number of a participant discovery announcement.
- #define DPSE_LOG_ANNOUNCE_WRITE_EC (DPSE_LOG_BASE + 44)

  Failed to write a participant discovery announcement message.
- #define DPSE_LOG_SCHEDULE_FAST_ASSERTION_EC (DPSE_LOG_BASE + 45)

  Failed to schedule an event to send the next participant discovery announcment.
- #define DPSE_LOG_PARTICIPANT_TAKE_EC (DPSE_LOG_BASE + 46)

  Failed to take a sample from a participant discovery DataReader.
- #define DPSE_LOG_ASSERT_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 47)

  Failed to assert remote participant.
- #define DPSE_LOG_ON_ASSERT_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 48)

  Failed to assert and complete discovery of a remote participant.

- #define DPSE_LOG_ADD_ANONYMOUS_ROUTE_EC (DPSE_LOG_BASE + 49)

  Failed to add an anonymous route.
- #define DPSE_LOG_DELETE_ANONYMOUS_ROUTE_EC (DPSE_LOG_BASE + 50)

  Failed to delete an anonymous route.
- #define DPSE_LOG_MAX_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 51)

  Exceeded resource limit, remote_participant_allocation.
- #define DPSE_LOG_REFRESH_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 52)

  Failed to refresh liveliness for a remote participant.
- #define DPSE_LOG_RESET_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 53)

  Failed to reset liveliness for a remote participant.
- #define DPSE_LOG_INVALID_DISCOVERY_SAMPLE_EC (DPSE_LOG_BASE + 54)

  Received a participant discovery announcement with invalid state.
- #define DPSE_LOG_RETURN_DISCOVERY_SAMPLE_EC (DPSE_LOG_BASE + 55)

  Failed to return a loan on a participant discovery announcement.
- #define DPSE_LOG_SERIALIZE_MULTICAST_EC (DPSE_LOG_BASE + 56)

  Failed to serialize a Multicast Locator parameter.
- #define DPSE_LOG_GET_DDS_PROPERTIES_EC (DPSE_LOG_BASE + 57)

  Failed to serialize a Multicast Locator parameter.
- #define DPSE_LOG_ENTITY_ENABLE_EC (DPSE_LOG_BASE + 58)

  Failed to enable the specified entity kind.
- #define DPSE_LOG_SEQUENCE_SETMAX_EC (DPSE_LOG_BASE + 59)

  Failed to set the maximum length of a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_SETLENGTH_EC (DPSE_LOG_BASE + 60)

  Failed to set the length of a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_GETREF_EC (DPSE_LOG_BASE + 61)

  Failed to get a reference at the specified index for a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_INITIALIZE_EC (DPSE_LOG_BASE + 62)

  Failed to initialize a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_FINALIZE_EC (DPSE_LOG_BASE + 63)

  Failed to finalize a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_COPY_EC (DPSE_LOG_BASE + 64)

  Failed to copy a sequence of the specified kind.
- #define DPSE_LOG_GET_TIMER_EC (DPSE_LOG_BASE + 65)

  Failed to get timer.
- #define DPSE_LOG_UNSUPPORTED_LOCATOR_EC (DPSE_LOG_BASE + 66)

  The locator kind is not supported by any transport registered with the domain participant and is dropped.
- #define DPSE_LOG_CHECKSUM_INCOMPATIBLE_PARTICIPANT_IGNORED_EC (DPDE_LOG_BASE + 67)

  A participant with incompatible checksum was discovered and ignored.

5.91.1 Detailed Description

DPSE. ModuleID = 10.

5.91.2 Macro Definition Documentation

5.91.2.1 #define DPSE_LOG_INVALID_PEER_ADDRESS_EC (DPSE_LOG_BASE + 1)

A peer address string specifies an invalid address.

5.91.2.2 #define DPSE_LOG_CREATE_DISCOVERY_PUBLISHER_EC (DPSE_LOG_BASE + 2)

Failed to create the publisher for a participant discovery writer.

5.91.2.3 #define DPSE_LOG_CREATE_DISCOVERY_SUBSCRIBER_EC (DPSE_LOG_BASE + 3)

Failed to create the subscriber for a participant discovery datareader.

5.91.2.4 #define DPSE_LOG_REGISTER_TYPE_EC (DPSE_LOG_BASE + 4)

Failed to register a built-in participant discovery type.

5.91.2.5 #define DPSE_LOG_CREATE_TOPIC_EC (DPSE_LOG_BASE + 5)

Failed to create a topic for the built-in participant discovery topic.

5.91.2.6 #define DPSE_LOG_CREATE_WRITER_EC (DPSE_LOG_BASE + 6)

Failed to create a DataWriter for participant discovery.

5.91.2.7 #define DPSE_LOG_CREATE_READER_EC (DPSE_LOG_BASE + 7)

Failed to create a DataReader for participant discovery.

5.91.2.8 #define DPSE_LOG_DISPOSE_EC (DPSE_LOG_BASE + 8)

Failed to dispose a participant discovery instance.

5.91.2.9 #define DPSE_LOG_ANNOUNCE_LOCAL_PARTICIPANT_DELETION_EC (DPSE_LOG_BASE + 9)

Failed to dispose a participant discovery instance upon deletion.

5.91.2.10 #define DPSE_LOG_DELETE_WRITER_EC (DPSE_LOG_BASE + 10)

Failed to delete a participant discovery DataWriter.

5.91.2.11 #define DPSE_LOG_DELETE_PUBLISHER_EC (DPSE_LOG_BASE + 11)

Failed to delete a participant discovery Publisher.

5.91.2.12 #define DPSE_LOG_DELETE_READER_EC (DPSE_LOG_BASE + 12)

Failed to delete a participant discovery DataReader.

5.91.2.13 #define DPSE_LOG_DELETE_TOPIC_EC (DPSE_LOG_BASE + 13)

Failed to delete a participant discovery Topic.

5.91.2.14 #define DPSE_LOG_DELETE_SUBSCRIBER_EC (DPSE_LOG_BASE + 14)

Failed to delete a participant discovery Subscriber.

5.91.2.15 #define DPSE_LOG_OBJECT_ALLOCATE_EC (DPSE_LOG_BASE + 15)

Failed to allocate an object of the specified kind.

5.91.2.16 #define DPSE_LOG_OBJECT_INITIALIZE_EC (DPSE_LOG_BASE + 16)

Failed to initialize an object of the specified kind.

5.91.2.17 #define DPSE_LOG_OBJECT_FINALIZE_EC (DPSE_LOG_BASE + 17)

Failed to finalize an object of the specified kind.

5.91.2.18 #define DPSE_LOG_OBJECT_DELETE_EC (DPSE_LOG_BASE + 18)

Failed to delete an object of the specified kind.

5.91.2.19 #define DPSE_LOG_OBJECT_INVALID_EC (DPSE_LOG_BASE + 19)

The object was invalid in the context it was used.

5.91.2.20 #define DPSE_LOG_CDR_SET_POSITION_EC (DPSE_LOG_BASE + 20)

Failed to set the CDR stream position.

5.91.2.21 #define DPSE_LOG_CDR_SERIALIZE_EC (DPSE_LOG_BASE + 21)

Failed to serialize the specified kind.

5.91.2.22 #define DPSE_LOG_CDR_DESERIALIZE_EC (DPSE_LOG_BASE + 22)

Failed to deserialize the specified kind.

5.91.2.23 #define DPSE_LOG_SERIALIZE_GUID_EC (DPSE_LOG_BASE + 23)

Failed to serialize a GUID key parameter.

5.91.2.24 #define DPSE_LOG_SERIALIZE_BUILTIN_ENDPOINTS_EC (DPSE_LOG_BASE + 24)

Failed to serialize a Builtin Endpoint Mask parameter.

5.91.2.25 #define DPSE_LOG_SERIALIZE_PROTOCOL_VERSION_EC (DPSE_LOG_BASE + 25)

Failed to serialize a Protocol Version parameter.

5.91.2.26 #define DPSE_LOG_SERIALIZE_VENDOR_ID_EC (DPSE_LOG_BASE + 26)

Failed to serialize a Vendor ID parameter.

5.91.2.27 #define DPSE_LOG_SERIALIZE_DEFAULT_UNICAST_EC (DPSE_LOG_BASE + 27)

Failed to serialize a Default Unicast Locator parameter.

5.91.2.28 #define DPSE_LOG_SERIALIZE_META_UNICAST_EC (DPSE_LOG_BASE + 28)

Failed to serialize a Meta Unicast Locator parameter.

5.91.2.29 #define DPSE_LOG_SERIALIZE_META_MULTICAST_EC (DPSE_LOG_BASE + 29)

Failed to serialize a Meta Multicast Locator parameter.

5.91.2.30 #define DPSE_LOG_SERIALIZE_LEASE_DURATION_EC (DPSE_LOG_BASE + 30)

Failed to serialize a Lease Duration parameter.

5.91.2.31 #define DPSE_LOG_SERIALIZE_PRODUCT_VERSION_EC (DPSE_LOG_BASE + 31)

Failed to serialize a Product Version parameter.

5.91.2.32 #define DPSE_LOG_DESERIALIZE_PRODUCT_VERSION_EC (DPSE_LOG_BASE + 32)

Failed to serialize a Product Version parameter.

5.91.2.33 #define DPSE_LOG_SERIALIZE_PARTICIPANT_NAME_EC (DPSE_LOG_BASE + 33)

Failed to serialize a Participant Name parameter.

5.91.2.34 #define DPSE_LOG_DESERIALIZE_GUID_EC (DPSE_LOG_BASE + 34)

Failed to deserialize a GUID key parameter.

5.91.2.35 #define DPSE_LOG_DESERIALIZE_BUILTIN_ENDPOINTS_EC (DPSE_LOG_BASE + 35)

Failed to deserialize a Builtin Endpoint Mask parameter.

5.91.2.36 #define DPSE_LOG_DESERIALIZE_PROTOCOL_VERSION_EC (DPSE_LOG_BASE + 36)

Failed to deserialize a Protocol Version parameter.

5.91.2.37 #define DPSE_LOG_DESERIALIZE_VENDOR_ID_EC (DPSE_LOG_BASE + 37)

Failed to deserialize a Vendor ID parameter.

5.91.2.38 #define DPSE_LOG_DESERIALIZE_TOO_MANY_LOCATORS_EC (DPSE_LOG_BASE + 38)

Cannot deserialize another locator parameter, having reached maximum of 4 unicast or 4 multicast locators.

5.91.2.39 #define DPSE_LOG_DESERIALIZE_PARTICIPANT_NAME_EC (DPSE_LOG_BASE + 39)

Failed to deserialize a Participant Name parameter.

5.91.2.40 #define DPSE_LOG_DESERIALIZE_UNKNOWN_PID_EC (DPSE_LOG_BASE + 40)

Failed to deserialize a parameter with an unknown ID.

5.91.2.41 #define DPSE_LOG_ANNOUNCEMENT_EC (DPSE_LOG_BASE + 41)

Failed to send a participant discovery announcement.

5.91.2.42 #define DPSE_LOG_UPDATE_PARTICIPANT_ASSERT_PERIOD_EC (DPSE_LOG_BASE + 42)

Failed to schedule an event to send the next participant discovery announcement.

5.91.2.43 #define DPSE_LOG_ADVANCE_SN_EC (DPSE_LOG_BASE + 43)

Failed to advance the sequence number of a participant discovery announcement.

5.91.2.44 #define DPSE_LOG_ANNOUNCE_WRITE_EC (DPSE_LOG_BASE + 44)

Failed to write a participant discovery announcement message.

5.91.2.45 #define DPSE_LOG_SCHEDULE_FAST_ASSERTION_EC (DPSE_LOG_BASE + 45)

Failed to schedule an event to send the next participant discovery announcment.

5.91.2.46 #define DPSE_LOG_PARTICIPANT_TAKE_EC (DPSE_LOG_BASE + 46)

Failed to take a sample from a participant discovery DataReader.

5.91.2.47 #define DPSE_LOG_ASSERT_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 47)

Failed to assert remote participant.

5.91.2.48 #define DPSE_LOG_ON_ASSERT_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 48)

Failed to assert and complete discovery of a remote participant.

5.91.2.49 #define DPSE_LOG_ADD_ANONYMOUS_ROUTE_EC (DPSE_LOG_BASE + 49)

Failed to add an anonymous route.

Results in a DomainParticipant not having a destination to which it should have sent participant discovery announcements

5.91.2.50 #define DPSE_LOG_DELETE_ANONYMOUS_ROUTE_EC (DPSE_LOG_BASE + 50)

Failed to delete an anonymous route.

5.91.2.51 #define DPSE_LOG_MAX_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 51)

Exceeded resource limit, remote_participant_allocation.

5.91.2.52 #define DPSE_LOG_REFRESH_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 52)

Failed to refresh liveliness for a remote participant.

5.91.2.53 #define DPSE_LOG_RESET_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 53)

Failed to reset liveliness for a remote participant.

5.91.2.54 #define DPSE_LOG_INVALID_DISCOVERY_SAMPLE_EC (DPSE_LOG_BASE + 54)

Received a participant discovery announcement with invalid state.

5.91.2.55 #define DPSE_LOG_RETURN_DISCOVERY_SAMPLE_EC (DPSE_LOG_BASE + 55)

Failed to return a loan on a participant discovery announcement.

5.91.2.56 #define DPSE_LOG_SERIALIZE_MULTICAST_EC (DPSE_LOG_BASE + 56)

Failed to serialize a Multicast Locator parameter.

5.91.2.57 #define DPSE_LOG_GET_DDS_PROPERTIES_EC (DPSE_LOG_BASE + 57)

Failed to serialize a Multicast Locator parameter.

5.91.2.58 #define DPSE_LOG_ENTITY_ENABLE_EC (DPSE_LOG_BASE + 58)

Failed to enable the specified entity kind.

5.91.2.59 #define DPSE_LOG_SEQUENCE_SETMAX_EC (DPSE_LOG_BASE + 59)

Failed to set the maximum length of a sequence of the specified kind.

5.91.2.60 #define DPSE_LOG_SEQUENCE_SETLENGTH_EC (DPSE_LOG_BASE + 60)

Failed to set the length of a sequence of the specified kind.

5.91.2.61 #define DPSE_LOG_SEQUENCE_GETREF_EC (DPSE_LOG_BASE + 61)

Failed to get a reference at the specified index for a sequence of the specified kind.

5.91.2.62 #define DPSE_LOG_SEQUENCE_INITIALIZE_EC (DPSE_LOG_BASE + 62)

Failed to initialize a sequence of the specified kind.

5.91.2.63 #define DPSE_LOG_SEQUENCE_FINALIZE_EC (DPSE_LOG_BASE + 63)

Failed to finalize a sequence of the specified kind.

5.91.2.64 #define DPSE_LOG_SEQUENCE_COPY_EC (DPSE_LOG_BASE + 64)

Failed to copy a sequence of the specified kind.

5.91.2.65 #define DPSE_LOG_GET_TIMER_EC (DPSE_LOG_BASE + 65)

Failed to get timer.

5.91.2.66 #define DPSE_LOG_UNSUPPORTED_LOCATOR_EC (DPSE_LOG_BASE + 66)

The locator kind is not supported by any transport registered with the domain participant and is dropped.

5.91.2.67 #define DPSE_LOG_CHECKSUM_INCOMPATIBLE_PARTICIPANT_IGNORED_EC (DPDE_LOG_BASE + 67)

A participant with incompatible checksum was discovered and ignored.

5.92 Notification Interface

ZCOPY Notification Interface.

Data Structures

- struct ZCOPY_NotifInterfaceFactoryProperty

  Notification interface factory specific property.

Macros

- #define ZCOPY_NotifInterfaceFactoryProperty_INITIALIZER

  Constant to initialize a ZCOPY_NotifInterfaceFactoryProperty.

Functions

- RTI_BOOL ZCOPY_NotifInterface_receive (NETIO_Interface_T *netio_intf, const struct NETIO_Address *port, RTI_BOOL *has_more_data_out)

  Definition of the NETIO_InterfaceI receive method.
- RTI_BOOL ZCOPY_NotifInterfaceFactory_register (RT_Registry_T *registry, const char *name, struct ZCOPY_NotifInterfaceFactoryProperty *property)

  Register the a Notification interface factory with a factory property.
- RTI_BOOL ZCOPY_NotifInterfaceFactory_unregister (RT_Registry_T *registry, const char *name)

  Unregister a Notification interface factory.

5.92.1 Detailed Description

ZCOPY Notification Interface. The Notification Interface is implemented as a NETIO interface and NETIO interface factory.

5.92.2 Macro Definition Documentation

5.92.2.1 #define ZCOPY_NotifInterfaceFactoryProperty_INITIALIZER

Value:

```
{                   \
    NETIO_InterfaceFactoryProperty_INITIALIZER, /* _parent */               \
    1,                                          /* max_samples_per_notif */ \
    NULL,                                       /* user_intf */             \
    NULL,                                       /* user_property */         \
}
```

Constant to initialize a ZCOPY_NotifInterfaceFactoryProperty.

5.92.3 Function Documentation

5.92.3.1 RTI_BOOL ZCOPY_NotifInterface_receive (NETIO_Interface_T * *netio_intf,* const struct NETIO_Address * *port,* RTI_BOOL * *has_more_data_out*)

Definition of the NETIO_InterfaceI receive method.

Parameters

in	*netio_intf*	The Notification interface to receive on
in	*port*	The port to receive on
out	*has_more_data-_out*	Whether there is more data to receive

Returns

RTI_TRUE on success, RTI_FALSE on failure

See Also NETIO_Interface_receive, NETIO_InterfaceI

5.92.3.2 RTI_BOOL ZCOPY_NotifInterfaceFactory_register (RT_Registry_T * *registry,* const char * *name,* struct ZCOPY_NotifInterfaceFactoryProperty * *property*)

Register the a Notification interface factory with a factory property.

Parameters

in	*registry*	The registry to register the factory with
in	*name*	The name to register the factory with
in	*property*	The property to register the factory with

Returns

RTI_TRUE if the factory was registered successfully, RTI_FALSE otherwise

5.92.3.3 RTI_BOOL ZCOPY_NotifInterfaceFactory_unregister (RT_Registry_T * *registry,* const char * *name*)

Unregister a Notification interface factory.

Parameters

in	*registry*	The registry to unregister the factory from
in	*name*	The name to unregister the factory from

Returns

RTI_TRUE if the factory was unregistered successfully, RTI_FALSE otherwise

5.93 User notification mechanism interface

Notification mechanism definitions.

Data Structures

- struct ZCOPY_NotifUserInterfaceI

  Notification mechanism user interface.

Typedefs

- typedef NETIO_Interface_T *(* ZCOPY_NotifUserInterface_createFunc)(NETIO_Interface_T *upstream, void *user_property)

  Create an instance of a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_reserve_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, void **port_entry_out)

  Reserve an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_release_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, void *port_entry)

  Release an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_add_routeFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source, struct NETIO_Address *destination, void **route_entry_out)

  Add a route on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_delete_routeFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source, struct NETIO_Address *destination, void *route_entry)

  Delete a route on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_sendFunc)(NETIO_Interface_T *self, struct NETIO_Address *source, struct NETIO_Address *destination, void *route_entry)

  Send a notification on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_bindFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, struct NETIO_Address *dst_addr, void *port_entry, void **bind_entry_out)

  Bind an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_unbindFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, struct NETIO_Address *dst_addr, void *port_entry, void *bind_entry)

  Unbind an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_notify_portFunc)(NETIO_Interface_T *user_intf, void *port_entry)

  Notify a specific receive port.

5.93.1 Detailed Description

Notification mechanism definitions.

5.93.2 Typedef Documentation

5.93.2.1 typedef NETIO_Interface_T*(* ZCOPY_NotifUserInterface_createFunc)(NETIO_Interface_T *upstream, void *user_property)

Create an instance of a user interface.

Parameters

in	*upstream*	The upstream notification interface
in	*user_property*	The user property

Returns

The user interface instance or NULL if an error occurred

5.93.2.2 typedef RTI_BOOL(* ZCOPY_NotifUserInterface_reserve_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, void **port_entry_out)

Reserve an address on a user interface.

Parameters

in	*user_intf*	The user interface instance
in	*src_addr*	The address to reserve
out	*port_entry_out*	The port entry associated with the address

5.93.2.3 typedef RTI_BOOL(* ZCOPY_NotifUserInterface_release_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, void *port_entry)

Release an address on a user interface.

Parameters

in	*user_intf*	The user interface instance
in	*src_addr*	The address to release
in	*port_entry*	The port entry associated with the address

Returns

RTI_TRUE on success, RTI_FALSE otherwise

5.93.2.4 typedef RTI_BOOL(* ZCOPY_NotifUserInterface_add_routeFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source, struct NETIO_Address *destination, void **route_entry_out)

Add a route on a user interface.

Parameters

in	*user_intf*	The user interface instance
in	*source*	The source address
in	*destination*	The destination address
out	*route_entry_out*	The route entry associated with the route

Returns

RTI_TRUE on success, RTI_FALSE otherwise

5.93.2.5 typedef RTI_BOOL(* ZCOPY_NotifUserInterface_delete_routeFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source, struct NETIO_Address *destination, void *route_entry)

Delete a route on a user interface.

Parameters

in	*user_intf*	The user interface instance
in	*source*	The source address
in	*destination*	The destination address
in	*route_entry*	The route entry associated with the route

Returns

RTI_TRUE on success, RTI_FALSE otherwise

5.93.2.6 typedef RTI_BOOL(∗ ZCOPY_NotifUserInterface_sendFunc)(NETIO_Interface_T ∗self, struct NETIO_Address ∗source, struct NETIO_Address ∗destination, void ∗route_entry)

Send a notification on a user interface.

Parameters

in	*user_intf*	The user interface instance
in	*source*	The source address
in	*destination*	The destination address
in	*route_entry*	The route entry associated with the route

Returns

RTI_TRUE on success, RTI_FALSE otherwise

5.93.2.7 typedef RTI_BOOL(∗ ZCOPY_NotifUserInterface_bindFunc)(NETIO_Interface_T ∗user_intf, struct NETIO_Address ∗src_addr,struct NETIO_Address ∗dst_addr,void ∗port_entry, void ∗∗bind_entry_out)

Bind an address on a user interface.

Parameters

in	*user_intf*	The user interface instance
in	*src_addr*	The remote address
in	*dst_addr*	The local address
in	*port_entry*	The port entry associated with the local address
out	*bind_entry_out*	The bind entry associated with the bind

Returns

RTI_TRUE on success, RTI_FALSE otherwise

5.93.2.8 typedef RTI_BOOL(∗ ZCOPY_NotifUserInterface_unbindFunc)(NETIO_Interface_T ∗user_intf, struct NETIO_Address ∗src_addr,struct NETIO_Address ∗dst_addr,void ∗port_entry, void ∗bind_entry)

Unbind an address on a user interface.

Parameters

in	*user_intf*	The user interface instance
in	*src_addr*	The remote address
in	*dst_addr*	The local address
in	*port_entry*	The port entry associated with the local address
in	*bind_entry*	The bind entry associated with the bind

Returns

RTI_TRUE on success, RTI_FALSE otherwise

5.93.2.9 typedef RTI_BOOL(∗ ZCOPY_NotifUserInterface_notify_portFunc)(NETIO_Interface_T ∗user_intf, void ∗port_entry)

Notify a specific receive port.

Used by the reader to send notification on a specific receive port.

Parameters

in	*user_intf*	The user interface instance
in	*port_entry*	The port entry to notify

Returns

RTI_TRUE on success, RTI_FALSE otherwise

5.94 WHSQ

Writer History Shared Queue. ModuleID = 22.

Macros

- #define WHSQ_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (WHSQ_LOG_BASE + 1)
  KEEP_ALL History kind is not supported.
- #define WHSQ_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC (WHSQ_LOG_BASE + 2)
  Unlimited length resource limits unsupported.
- #define WHSQ_LOG_MAX_SAMPLES_TOO_SMALL_EC (WHSQ_LOG_BASE + 3)
  DataWriterQos.resource_limits.max_samples set too small.
- #define WHSQ_LOG_OBJECT_ALLOCATE_EC (WHSQ_LOG_BASE + 4)
  Failed to allocate object of the specified kind.
- #define WHSQ_LOG_PURGE_FAILED_EC (WHSQ_LOG_BASE + 5)
  Failed to purge a sample from the shared queue.
- #define WHSQ_LOG_WH_CREATE_FAILED_EC (WHSQ_LOG_BASE + 6)
  Failed to create the underlying writer history.
- #define WHSQ_LOG_OBJECT_DELETE_EC (WHSQ_LOG_BASE + 7)
  Failed to delete object of the specified kind.
- #define WHSQ_LOG_CONFIGURATION_EC (WHSQ_LOG_BASE + 8)
  Listener or Property not configured when creating the component.
- #define WHSQ_LOG_NO_DDS_FACTORY_EC (WHSQ_LOG_BASE + 9)
  No factory for forwarding when creating a writer history instance.
- #define WHSQ_LOG_COMMIT_FAILED_EC (WHSQ_LOG_BASE + 10)
  Committing a Sample failed.

5.94.1 Detailed Description

Writer History Shared Queue. ModuleID = 22.

5.94.2 Macro Definition Documentation

5.94.2.1 #define WHSQ_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (WHSQ_LOG_BASE + 1)

KEEP_ALL History kind is not supported.

5.94.2.2 #define WHSQ_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC (WHSQ_LOG_BASE + 2)

Unlimited length resource limits unsupported.

Ensure that max_samples, max_samples_per_instance, and max_instances resource limits are all finite values

5.94.2.3 #define WHSQ_LOG_MAX_SAMPLES_TOO_SMALL_EC (WHSQ_LOG_BASE + 3)

DataWriterQos.resource_limits.max_samples set too small.

DataWriterQos.resource_limits.max_samples must be no less than max_instances $*$ max_samples_per_instance

5.94.2.4 #define WHSQ_LOG_OBJECT_ALLOCATE_EC (WHSQ_LOG_BASE + 4)

Failed to allocate object of the specified kind.

5.94.2.5 #define WHSQ_LOG_PURGE_FAILED_EC (WHSQ_LOG_BASE + 5)

Failed to purge a sample from the shared queue.

5.94.2.6 #define WHSQ_LOG_WH_CREATE_FAILED_EC (WHSQ_LOG_BASE + 6)

Failed to create the underlying writer history.

5.94.2.7 #define WHSQ_LOG_OBJECT_DELETE_EC (WHSQ_LOG_BASE + 7)

Failed to delete object of the specified kind.

5.94.2.8 #define WHSQ_LOG_CONFIGURATION_EC (WHSQ_LOG_BASE + 8)

Listener or Property not configured when creating the component.

5.94.2.9 #define WHSQ_LOG_NO_DDS_FACTORY_EC (WHSQ_LOG_BASE + 9)

No factory for forwarding when creating a writer history instance.

5.94.2.10 #define WHSQ_LOG_COMMIT_FAILED_EC (WHSQ_LOG_BASE + 10)

Committing a Sample failed.

5.95 Log Codes

Modules

- REDA

  Real-time Efficient Data Structures and Algorithms. ModuleID = 1.
- DB

  Database. ModuleID = 2.
- CDR

  CDR. ModuleID = 5.
- RT

  RT. ModuleID = 3.
- NETIO

  Network I/O. ModuleID = 4.
- RTPS

  Real-Time Publish-Subscribe. ModuleID = 6.
- DDS_C

  DDS C. ModuleID = 7.
- RH

  Reader History. ModuleID = 8.
- WH

  Writer History. ModuleID = 9.
- DPSE

  DPSE. ModuleID = 10.
- WHSQ

  Writer History Shared Queue. ModuleID = 22.

5.95.1 Detailed Description

This page contains references to all log codes reported by RTI Connext Micro. Please refer to the Debugging section in the User's Manual for how to interpret log messages and Log API for the Log API. Please note that logging is enabled enabled in the Debug build of RTI Connext Micro Cert.

5.96 NETIO Address API

Data Structures

- struct NETIO_AddressUInt32

  <<eXtension>> <<cert>> NETIO_AddressUInt32
- union NETIO_AddressValue

  <<eXtension>> <<cert>> NETIO_AddresValue
- struct NETIO_Address
- struct NETIO_Netmask

  <<eXtension>> <<cert>> NETIO_Netmask
- struct NETIO_AddressSeq

  <<eXtension>> <<cert>> The NETIO_AddressSeq sequence type.
- struct NETIO_NetmaskSeq

  <<eXtension>> <<cert>> The NETIO_NetmaskSeq sequence type.

Macros

- #define NETIO_ADDRESS_KIND_UDPv4 1

  <<eXtension>> <<cert>> The UDPv4 NETIO Address.
- #define NETIO_ADDRESS_KIND_UDPv6 2

  <<eXtension>> <<cert>> The UDPv6 NETIO Address.
- #define NETIO_ADDRESS_RTPS_RESERVED_LOW 0x02000000

  <<eXtension>> <<cert>> Addresses in the range [NETIO_ADDRESS_RTPS_RESERVED_LOW, NETIO_ADDRESS_RTPS_RESERVED_HIGH] are reserved by the RTPS standard.
- #define NETIO_ADDRESS_RTPS_RESERVED_HIGH 0x02ffffff

  <<eXtension>> <<cert>> Addresses in the range [NETIO_ADDRESS_RTPS_RESERVED_LOW, NETIO_ADDRESS_RTPS_RESERVED_HIGH] are reserved by the RTPS standard.
- #define NETIO_Netmask_INITIALIZER

  <<eXtension>> <<cert>> Initialize a NETIO_Netmask.
- #define NETIO_Address_INITIALIZER

  <<eXtension>> <<cert>> Initialize a NETIO_Address.

Functions

- RTI_BOOL NETIO_Address_set_multicast (struct NETIO_Address *const addr)

  Mark a NETIO_Address as a multicast address.
- void NETIO_Address_set_ipv4 (struct NETIO_Address *addr, RTI_UINT32 port, RTI_UINT32 address)

  Set the IPv4 address.

5.96.1 Detailed Description

5.96.2 Macro Definition Documentation

5.96.2.1 #define NETIO_ADDRESS_KIND_UDPv4 1

<<eXtension>> <<cert>> The UDPv4 NETIO Address.

5.96.2.2 #define NETIO_ADDRESS_KIND_UDPv6 2

<<eXtension>> <<cert>> The UDPv6 NETIO Address.

5.96.2.3 #define NETIO_ADDRESS_RTPS_RESERVED_LOW 0x02000000

<<eXtension>> <<cert>> Addresses in the range [NETIO_ADDRESS_RTPS_RESERVED_LOW, NETIO_ADDRESS_RTPS_RESERVED_HIGH] are reserved by the RTPS standard.

5.96.2.4 #define NETIO_ADDRESS_RTPS_RESERVED_HIGH 0x02ffffff

<<eXtension>> <<cert>> Addresses in the range [NETIO_ADDRESS_RTPS_RESERVED_LOW, NETIO_ADDRESS_RTPS_RESERVED_HIGH] are reserved by the RTPS standard.

5.96.2.5 #define NETIO_Netmask_INITIALIZER

Value:

```
{\
    0,\
    {0,0,0,0}\
}
```

<<eXtension>> <<cert>> Initialize a NETIO_Netmask.

5.96.2.6 #define NETIO_Address_INITIALIZER

Value:

```
{\
    NETIO_ADDRESS_KIND_RESERVED, /* kind */ \
    0, /* port */ \
    {{0,0,0,0}} /* value */ \
}
```

<<eXtension>> <<cert>> Initialize a NETIO_Address.

5.96.3 Function Documentation

5.96.3.1 RTI_BOOL NETIO_Address_set_multicast (struct NETIO_Address *const *addr*)

Mark a NETIO_Address as a multicast address.

Parameters

in	*addr*	Address to set as multicast. Must be non-NULL.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

MT Safety:

This operation is not thread safe.

5.96.3.2 void NETIO_Address_set_ipv4 (struct NETIO_Address * *addr*, RTI_UINT32 *port*, RTI_UINT32 *address*)

Set the IPv4 address.

<<eXtension>> <<cert>>

MT Safety:

This operation is not thread safe. Set the IPv4 address part of a NETIO_Address

Parameters

in	*addr*	Address to initialize
in	*port*	Address port
in	*address*	IPv4 address in host order

6 Data Structure Documentation

6.1 DDS_BuiltinTopicKey_t Struct Reference

The key type of the built-in topic types.

Data Fields

- DDS_BUILTIN_TOPIC_KEY_TYPE_NATIVE value [DDS_BUILTIN_TOPIC_KEY_TYPE_NATIVE_LENGTH]

  An array of four integers that uniquely represents a remote DDS_Entity.

6.1.1 Detailed Description

The key type of the built-in topic types.

Each remote DDS_Entity to be discovered can be uniquely identified by this key. This is the key of all the built-in topic data types.

The value of element DDS_BUILTIN_TOPIC_KEY_OBJECT_ID is the object ID of the remote entity. In Dynamic Participant/Static Endpoint discovery, this must be set correctly for each remote DDS_DataReader and DDS_DataWriter that an application intends to discover.

For example, to configure a remote DDS_DataWriter with object ID 100 by setting the key in the DDS_PublicationBuiltinTopicData:

```
remote_pub_data.key.value[DDS_BUILTIN_TOPIC_KEY_OBJECT_ID]  = 100;
```

See also

DDS_ParticipantBuiltinTopicData
DDS_PublicationBuiltinTopicData
DDS_SubscriptionBuiltinTopicData

6.1.2 Field Documentation

6.1.2.1 DDS_BUILTIN_TOPIC_KEY_TYPE_NATIVE DDS_BuiltinTopicKey_t::value[DDS_BUILTIN_TOPIC_KEY_TYPE_NATIVE_LENGTH]

An array of four integers that uniquely represents a remote DDS_Entity.

6.2 DDS_ChecksumProperty Struct Reference

<<eXtension>> Type to define the checksum properties of the participant RTI Connext Micro.

Data Fields

- DDS_ChecksumKind_t computed_crc_kind

  The checksum kind for the computed and sent checksum.
- DDS_ChecksumKindMask_t allowed_crc_mask

  The allowed checksum for checked checksum.
- DDS_Boolean require_crc

  Whether the discovered participant requires checksum or not.

6.2.1 Detailed Description

<<*eXtension*>> Type to define the checksum properties of the participant RTI Connext Micro.

6.2.2 Field Documentation

6.2.2.1 DDS_ChecksumKind_t DDS_ChecksumProperty::computed_crc_kind

The checksum kind for the computed and sent checksum.

6.2.2.2 DDS_ChecksumKindMask_t DDS_ChecksumProperty::allowed_crc_mask

The allowed checksum for checked checksum.

6.2.2.3 DDS_Boolean DDS_ChecksumProperty::require_crc

Whether the discovered participant requires checksum or not.

6.3 DDS_ChecksumProperty_t Struct Reference

<<*eXtension*>> Type to define the checksum properties of the participant RTI Connext Micro.

Data Fields

- DDS_ChecksumKind_t computed_crc_kind

  The checksum kind for the computed and sent checksum.
- DDS_ChecksumKindMask_t allowed_crc_mask

  The allowed checksum for checked checksum.
- DDS_Boolean require_crc

  Whether the discovered participant requires checksum or not.

6.3.1 Detailed Description

<<*eXtension*>> Type to define the checksum properties of the participant RTI Connext Micro.

6.3.2 Field Documentation

6.3.2.1 DDS_ChecksumKind_t DDS_ChecksumProperty_t::computed_crc_kind

The checksum kind for the computed and sent checksum.

6.3.2.2 DDS_ChecksumKindMask_t DDS_ChecksumProperty_t::allowed_crc_mask

The allowed checksum for checked checksum.

6.3.2.3 DDS_Boolean DDS_ChecksumProperty_t::require_crc

Whether the discovered participant requires checksum or not.

6.4 DDS_ConditionSeq Struct Reference

A sequence object containing references to one or more DDS_Condition objects.

6.4.1 Detailed Description

A sequence object containing references to one or more DDS_Condition objects.

6.5 DDS_DataReaderInstanceReplacedStatus Struct Reference

<<eXtension>> *<<cert>>* DDS_INSTANCE_REPLACED_STATUS.

Data Fields

- DDS_Long total_count

  The total number of instances that have been replaced during the life-span of the DDS_DataReader.
- DDS_Long total_count_change

  The incremental number of instances replaced since the last time the listener was called or the status was read.
- DDS_InstanceHandle_t last_instance_handle

  The instance that is being replaced.
- DDS_InstanceHandle_t last_replacement_instance

  Handle to the instance which last replaced an instance.
- DDS_InstanceHandle_t publication_handle

  Handle to the publisher which last replaced an instance.
- DDS_Long lost_samples

  The number of samples which where lost due to the instance being replaced. Note that this number is an approximation because the middle-ware does not.

6.5.1 Detailed Description

<<eXtension>> *<<cert>>* DDS_INSTANCE_REPLACED_STATUS.

A status structure describing the current instance replacement status for the reader.

6.5.2 Field Documentation

6.5.2.1 DDS_Long DDS_DataReaderInstanceReplacedStatus::total_count

The total number of instances that have been replaced during the life-span of the DDS_DataReader.

6.5.2.2 DDS_Long DDS_DataReaderInstanceReplacedStatus::total_count_change

The incremental number of instances replaced since the last time the listener was called or the status was read.

6.5.2.3 DDS_InstanceHandle_t DDS_DataReaderInstanceReplacedStatus::last_instance_handle

The instance that is being replaced.

6.5.2.4 DDS_InstanceHandle_t DDS_DataReaderInstanceReplacedStatus::last_replacement_instance

Handle to the instance which last replaced an instance.

6.5.2.5 DDS_InstanceHandle_t DDS_DataReaderInstanceReplacedStatus::publication_handle

Handle to the publisher which last replaced an instance.

6.5.2.6 DDS_Long DDS_DataReaderInstanceReplacedStatus::lost_samples

The number of samples which where lost due to the instance being replaced. Note that this number is an approximation because the middle-ware does not.

6.6 DDS_DataReaderListener Struct Reference

<<interface>> *<<cert>>* DDS_Listener for reader status.

Data Fields

- struct DDS_Listener as_listener

  <<cert>> *The superclass instance of this DDS_DataReaderListener.*
- DDS_DataReaderListener_RequestedDeadlineMissedCallback on_requested_deadline_missed

  <<cert>> *Handles the DDS_REQUESTED_DEADLINE_MISSED_STATUS communication status.*
- DDS_DataReaderListener_RequestedIncompatibleQosCallback on_requested_incompatible_qos

  <<cert>> *Handles the DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS communication status.*
- DDS_DataReaderListener_SampleRejectedCallback on_sample_rejected

  <<cert>> *Handles the DDS_SAMPLE_REJECTED_STATUS communication status.*
- DDS_DataReaderListener_LivelinessChangedCallback on_liveliness_changed

  <<cert>> *Handles the DDS_LIVELINESS_CHANGED_STATUS communication status.*
- DDS_DataReaderListener_DataAvailableCallback on_data_available

  <<cert>> *Handle the DDS_DATA_AVAILABLE_STATUS communication status.*
- DDS_DataReaderListener_SubscriptionMatchedCallback on_subscription_matched

  <<cert>> *Handles the DDS_SUBSCRIPTION_MATCHED_STATUS communication status.*
- DDS_DataReaderListener_SampleLostCallback on_sample_lost

  <<cert>> *Handles the DDS_SAMPLE_LOST_STATUS communication status.*
- DDS_DataReaderListener_BeforeSampleDeserializeCallback on_before_sample_deserialize

  <<cert>> *Callback to filter a received sample based on serialized data*
- DDS_DataReaderListener_BeforeSampleCommitCallback on_before_sample_commit

  <<cert>> *Callback to filter a received sample based on deserialized data*
- DDS_DataReaderListener_InstanceReplacedCallback on_instance_replaced

  <<cert>> *Handles the DDS_INSTANCE_REPLACED_STATUS communication status.*

6.6.1 Detailed Description

<<interface>> *<<cert>>* DDS_Listener for reader status.

Entity:

DDS_DataReader

Status:

DDS_DATA_AVAILABLE_STATUS;
DDS_LIVELINESS_CHANGED_STATUS, DDS_LivelinessChangedStatus;
DDS_REQUESTED_DEADLINE_MISSED_STATUS, DDS_RequestedDeadlineMissedStatus;
DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS, DDS_RequestedIncompatibleQosStatus;
DDS_SAMPLE_LOST_STATUS, DDS_SampleLostStatus;
DDS_SAMPLE_REJECTED_STATUS, DDS_SampleRejectedStatus;
DDS_SUBSCRIPTION_MATCHED_STATUS, DDS_SubscriptionMatchedStatus;
DDS_INSTANCE_REPLACED_STATUS, DDS_DataReaderInstanceReplacedStatus;

See also

Status Kinds

6.6.2 Field Documentation

6.6.2.1 struct DDS_Listener DDS_DataReaderListener::as_listener

<<*cert*>> The superclass instance of this DDS_DataReaderListener.

6.6.2.2 DDS_DataReaderListener_RequestedDeadlineMissedCallback DDS_DataReaderListener::on_requested_deadline_missed

<<*cert*>> Handles the DDS_REQUESTED_DEADLINE_MISSED_STATUS communication status.

6.6.2.3 DDS_DataReaderListener_RequestedIncompatibleQosCallback DDS_DataReaderListener::on_requested_incompatible_qos

<<*cert*>> Handles the DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS communication status.

6.6.2.4 DDS_DataReaderListener_SampleRejectedCallback DDS_DataReaderListener::on_sample_rejected

<<*cert*>> Handles the DDS_SAMPLE_REJECTED_STATUS communication status.

6.6.2.5 DDS_DataReaderListener_LivelinessChangedCallback DDS_DataReaderListener::on_liveliness_changed

<<*cert*>> Handles the DDS_LIVELINESS_CHANGED_STATUS communication status.

6.6.2.6 DDS_DataReaderListener_DataAvailableCallback DDS_DataReaderListener::on_data_available

<<*cert*>> Handle the DDS_DATA_AVAILABLE_STATUS communication status.

6.6.2.7 DDS_DataReaderListener_SubscriptionMatchedCallback DDS_DataReaderListener::on_subscription_matched

<<*cert*>> Handles the DDS_SUBSCRIPTION_MATCHED_STATUS communication status.

6.6.2.8 DDS_DataReaderListener_SampleLostCallback DDS_DataReaderListener::on_sample_lost

<<*cert*>> Handles the DDS_SAMPLE_LOST_STATUS communication status.

6.6.2.9 DDS_DataReaderListener_BeforeSampleDeserializeCallback DDS_DataReaderListener::on_before_sample_deserialize

<<*cert*>> Callback to filter a received sample based on serialized data

When a DataReader receives a (serialized) sample, it will deserialize it before storing it in its queue. On_before_sample_deserialize() is called before deserialization. It allows the application to determine whether or not to drop the sample, before it is read or taken.

The callback controls whether the sample is filter out of the DataReader's queue, with the sample_dropped parameter. If user code within the callback determines that the sample should be dropped, the user should set sample_dropped to true; consequently, when the callback returns, the DataReader will drop the sample before it is committed to the queue. Otherwise, returning the callback with sample_dropped as false will allow the sample to be read or taken.

Compared to on_before_sample_commit(), the sample would be dropped with less processing (without deserialization) by the subscribing application, but with the added complexity of filtering on serialized data.

The callback is not associated with a DDS Status. It is enabled when the callback in the listener is assigned, to a non-NULL callback.

6.6.2.10 DDS_DataReaderListener_BeforeSampleCommitCallback DDS_DataReaderListener::on_before_sample_commit

<<*cert*>> Callback to filter a received sample based on deserialized data

When a DataReader receives a (serialized) sample, it will deserialize it before storing it in its queue. On_before_sample_commit() is called after deserialization, but before the sample is stored into the DataReader's queue. It allows the application to determine whether or not to drop the sample, before it is read or taken.

The callback controls whether the sample is filter out of the DataReader's queue, with the sample_dropped parameter. If user code within the callback determines that the sample should be dropped, the user should set sample_dropped to true; consequently, when the callback returns, the DataReader will drop the sample before it is committed to the queue. Otherwise, returning the callback with sample_dropped as false will allow the sample to be read or taken.

Compared to on_before_sample_deserialize(), the sample would be dropped with more processing (with deserialization) by the subscribing application, but with the lesser complexity of filtering on deserialized data.

The callback is not associated with a DDS Status. It is enabled when the callback in the listener is assigned, to a non-NULL callback.

The following fields are valid in the DDS_SampleInfo structure:

- source_timestamp
- instance_handle
- publication_handle
- valid_data
- encapsulation_id

6.6.2.11 DDS_DataReaderListener_InstanceReplacedCallback DDS_DataReaderListener::on_instance_replaced

<<*cert*>> Handles the DDS_INSTANCE_REPLACED_STATUS communication status.

This listener may only be called when the instance replacement policy is set to DDS_REPLACE_OLDEST_INSTANCE_REPLACEMENT_QOS. When a new instance is detected and resources are exhausted the oldest instance will be removed even if samples with a not read state exists for the instance. Thus, this is a forceful removal of samples and the instance. The freed instance resources will be re-used for the newly detected instance. If there are outstanding loans for the replaced instance the samples can not be freed until returned. However, when the loan is returned the samples are automatically freed. Note that the instance will not be rejected.

6.7 DDS_DataReaderProtocolQosPolicy Struct Reference

<<*eXtension*>> Protocol that applies only to DDS_DataReader instances.

Data Fields

- DDS_UnsignedLong rtps_object_id

  The RTPS Object ID.
- struct DDS_RtpsReliableReaderProtocol_t rtps_reliable_reader

  RTPS protocol-related configuration settings for the RTPS reliable reader associated to a DDS_DataReader. This parameter only has effect if the reader is configured with DDS_RELIABLE_RELIABILITY_QOS DDS_ReliabilityQosPolicyKind.

6.7.1 Detailed Description

<<*eXtension*>> Protocol that applies only to DDS_DataReader instances.

This QoS policy is an extension to the DDS standard.

Entity:

DDS_DataReader

Properties:

RxO = N/A
Changeable = NO

6.7.2 Field Documentation

6.7.2.1 DDS_UnsignedLong DDS_DataReaderProtocolQosPolicy::rtps_object_id

The RTPS Object ID.

This value is used to determine the RTPS object ID of a data reader according to the DDS-RTPS Interoperability Wire Protocol.

Only the last 3 bytes are used; the most significant byte is ignored.

[default] DDS_RTPS_AUTO_ID

[range] [0,0x00ffffff]

6.7.2.2 struct DDS_RtpsReliableReaderProtocol_t DDS_DataReaderProtocolQosPolicy::rtps_reliable_reader

RTPS protocol-related configuration settings for the RTPS reliable reader associated to a DDS_DataReader. This parameter only has effect if the reader is configured with DDS_RELIABLE_RELIABILITY_QOS DDS_ReliabilityQosPolicyKind.

For details, refer to the DDS_RtpsReliableReaderProtocol_t

[default]

nack_period 50ms seconds;

6.8 DDS_DataReaderQos Struct Reference

<<cert>> QoS policies supported by a DDS_DataReader entity.

Data Fields

- struct DDS_DeadlineQosPolicy deadline
  Deadline policy, DEADLINE.
- struct DDS_LivelinessQosPolicy liveliness
  Liveliness policy, LIVELINESS.
- struct DDS_HistoryQosPolicy history
  History policy, HISTORY.
- struct DDS_ResourceLimitsQosPolicy resource_limits
  Resource limits policy, RESOURCE_LIMITS.
- struct DDS_OwnershipQosPolicy ownership
  Ownership policy, OWNERSHIP.
- struct DDS_ReliabilityQosPolicy reliability
  Reliability policy, RELIABILITY.
- struct DDS_DurabilityQosPolicy durability
  <<eXtension>> *Durability policy, DURABILITY.*
- struct DDS_DestinationOrderQosPolicy destination_order
  <<eXtension>> *Destination Order policy, DESTINATION_ORDER.*

- struct DDS_DataReaderProtocolQosPolicy protocol

  <<eXtension>> DDS_DataReader protocol policy, DATA_READER_PROTOCOL
- struct DDS_TransportQosPolicy transport

  <<eXtension>> Transport policy, TRANSPORT.
- struct DDS_DataReaderResourceLimitsQosPolicy reader_resource_limits

  <<eXtension>> DDS_DataReader resource limits policy, DataReader Resource Limits. This policy is an extension to the DDS standard.

6.8.1 Detailed Description

<<cert>> QoS policies supported by a DDS_DataReader entity.

The DDS_DataReaderQos controls the resources used by the DDS_DataReader. Each DDS_DataReader allocates its own resources, even DataReaders of the same DDS_Topic. For this reason is it advised to limit the number of DataReaders per DDS_Topic.

6.8.2 Field Documentation

6.8.2.1 struct DDS_DeadlineQosPolicy DDS_DataReaderQos::deadline

Deadline policy, DEADLINE.

6.8.2.2 struct DDS_LivelinessQosPolicy DDS_DataReaderQos::liveliness

Liveliness policy, LIVELINESS.

6.8.2.3 struct DDS_HistoryQosPolicy DDS_DataReaderQos::history

History policy, HISTORY.

6.8.2.4 struct DDS_ResourceLimitsQosPolicy DDS_DataReaderQos::resource_limits

Resource limits policy, RESOURCE_LIMITS.

6.8.2.5 struct DDS_OwnershipQosPolicy DDS_DataReaderQos::ownership

Ownership policy, OWNERSHIP.

6.8.2.6 struct DDS_ReliabilityQosPolicy DDS_DataReaderQos::reliability

Reliability policy, RELIABILITY.

6.8.2.7 struct DDS_DurabilityQosPolicy DDS_DataReaderQos::durability

<<eXtension>> Durability policy, DURABILITY.

6.8.2.8 struct DDS_DestinationOrderQosPolicy DDS_DataReaderQos::destination_order

<<eXtension>> Destination Order policy, DESTINATION_ORDER.

6.8.2.9 struct DDS_DataReaderProtocolQosPolicy DDS_DataReaderQos::protocol

<<eXtension>> DDS_DataReader protocol policy, DATA_READER_PROTOCOL

6.8.2.10 struct DDS_TransportQosPolicy DDS_DataReaderQos::transport

<<eXtension>> Transport policy, TRANSPORT.

6.8.2.11 struct DDS_DataReaderResourceLimitsQosPolicy DDS_DataReaderQos::reader_resource_limits

<<*eXtension*>> DDS_DataReader resource limits policy, DataReader Resource Limits. This policy is an extension to the DDS standard.

6.9 DDS_DataReaderResourceLimitsQosPolicy Struct Reference

<<*cert*>> Resource limits that apply only to DDS_DataReader instances.

Data Fields

- DDS_Long max_remote_writers

  The maximum number of remote writers for which the reader will maintain state and communication.
- DDS_Long max_remote_writers_per_instance

  The maximum number of remote writers for a unique instance for which the reader will maintain state and communication.
- DDS_Long max_samples_per_remote_writer

  The maximum number of untaken samples from a given remote DataWriter that a DataReader may store.
- DDS_Long max_outstanding_reads

  The maximum number of outstanding read/take calls (or one of the variants) on the same DataReader for which the memory has not been returned by calling FooDataReader_return_loan.
- DDS_DataReaderResourceLimitsInstanceReplacementKind instance_replacement
- DDS_Long max_routes_per_writer

  The maximum number of routes the reader will send to per matched writer.

6.9.1 Detailed Description

<<*cert*>> Resource limits that apply only to DDS_DataReader instances.

This QoS policy is an extension to the DDS standard.

Entity:

DDS_DataReader

Properties:

RxO = N/A
Changeable = NO

6.9.2 Field Documentation

6.9.2.1 DDS_Long DDS_DataReaderResourceLimitsQosPolicy::max_remote_writers

The maximum number of remote writers for which the reader will maintain state and communication.

The DDS_DataReaderResourceLimitsQosPolicy::max_remote_writers resource-limit limits the maximum number of remote DataWriters the DDS_DataReader can receive data from regardless of which instance it is.

Resources controlled by this limit can only be reused when a remote DDS_DataWriter is either deleted or looses liveliness.

[default] 1

[range] [1, 1000000], >= max_remote_writers_per_instance

6.9.2.2 DDS_Long DDS_DataReaderResourceLimitsQosPolicy::max_remote_writers_per_instance

The maximum number of remote writers for a unique instance for which the reader will maintain state and communication.

The DDS_DataReaderResourceLimitsQosPolicy::max_remote_writers_per_instance limits the maximum number of remote DDS_DataWriter entities a DDS_DataReader maintains state for per instance. If this resource-limit is lower than the actual number of remote DataWriters, updating an instance only state for up to DDS_DataReaderResourceLimitsQosPolicy::max_remote_writers_per_instance is maintained. It is not guaranteed to be the last DDS_DataReaderResourceLimitsQosPolicy::max_remote_writers_per_instance though. However, the last remote DDS_DataWriter to update an instance is *always* saved. When an instance is no longer updated by a known, to the DDS_DataReader or DDS_DataWriter, DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE state is signaled on the DDS_DataReader. Thus, if this resource-limit is lower than the actual number of remote DataWriters updating an instance, DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE may be signaled even though there may be one or more DataWriters still updating an instance. In this case the instance will go back to the DDS_ALIVE_INSTANCE_STATE.

[default] 1

[range] [1, 2147483647], <= max_remote_writers

6.9.2.3 DDS_Long DDS_DataReaderResourceLimitsQosPolicy::max_samples_per_remote_writer

The maximum number of untaken samples from a given remote DataWriter that a DataReader may store.

This resource-limit limits the maximum number of samples that can be cached per remote DDS_DataWriter the DDS_DataReader is receiving data from. This resource-limit is useful in preventing one DDS_DataWriter from starving another DDS_DataWriter. NOTE: The samples cached from a DDS_DataWriter are allocated from the same sample pool controlled by DDS_ResourceLimitsQosPolicy::max_samples.

[default] 1

[range] [1, 100000000], <= DDS_ResourceLimitsQosPolicy::max_samples

6.9.2.4 DDS_Long DDS_DataReaderResourceLimitsQosPolicy::max_outstanding_reads

The maximum number of outstanding read/take calls (or one of the variants) on the same DataReader for which the memory has not been returned by calling FooDataReader_return_loan.

The DDS_DataReaderResourceLimitsQosPolicy::max_outstanding_reads resource-limit limits the maximum number of simultaneous loans an application can make at any given time using the FooDataReader_read() and FooDataReader_take() APIs.

Typically a value of 1 is sufficient. It is only necessary to increase this value if an application thread will access the same cache before another thread has returned the loan.

[default] 1

[range] [1, 2147483647]

6.9.2.5 DDS_DataReaderResourceLimitsInstanceReplacementKind DDS_DataReaderResourceLimitsQosPolicy::instance_replacement

[default] DDS_NO_INSTANCE_REPLACEMENT_QOS

6.9.2.6 DDS_Long DDS_DataReaderResourceLimitsQosPolicy::max_routes_per_writer

The maximum number of routes the reader will send to per matched writer.

NOTE: If this resource limit is exceeded when matching a DDS_DataReader with a DDS_DataWriter, the two entities are not matched.

Each DDS_DataReader maintains information about the state of its peer DDS_DataWriter (those it has matched with). Part of this state is which locators (or destination addresses) it should use to send data to a particular DDS_DataWriter. The DDS_DataReaderResourceLimitsQosPolicy::max_routes_per_writer resource-limit limits the maximum number of routes that can be saved per DDS_DataWriter.

This resource-limit is shared across all matched DDS_DataWriter per DDS_DataReader. Thus, if DDS_DataReaderResourceLimitsQosPolicy::max_remote_writers is 2 and DDS_DataReaderResourceLimitsQosPolicy::max_routes_per_writer is 4, a total of 8 routes can be saved for both DataWriters. One DDS_DataWriter may have 6 routes and the other 2.

[default] 4

[range] [1, 2000]

6.10 DDS_DataWriterListener Struct Reference

<<interface>> *<<cert>>* DDS_Listener for writer status.

Data Fields

- struct DDS_Listener as_listener

  <<cert>> *The superclass instance of this DDS_DataWriterListener.*
- DDS_DataWriterListener_OfferedDeadlineMissedCallback on_offered_deadline_missed

  <<cert>> *Handles the DDS_OFFERED_DEADLINE_MISSED_STATUS status.*
- DDS_DataWriterListener_OfferedIncompatibleQosCallback on_offered_incompatible_qos

  <<cert>> *Handles the DDS_OFFERED_INCOMPATIBLE_QOS_STATUS status.*
- DDS_DataWriterListener_LivelinessLostCallback on_liveliness_lost

  <<cert>> *Handles the DDS_LIVELINESS_LOST_STATUS status.*
- DDS_DataWriterListener_PublicationMatchedCallback on_publication_matched

  <<cert>> *Handles the DDS_PUBLICATION_MATCHED_STATUS status.*
- DDS_DataWriterListener_ReliableReaderActivityChangedCallback on_reliable_reader_activity_changed

  <<eXtension>> *<<cert>>* *A matched reliable reader has become active or become inactive.*
- DDS_DataWriterListener_ReliableSampleUnacknowledgedCallback on_reliable_sample_unacknowledged

  <<eXtension>> *<<cert>>* *A matched reliable reader has become active or become inactive.*

6.10.1 Detailed Description

<<interface>> *<<cert>>* DDS_Listener for writer status.

Entity:

DDS_DataWriter

Status:

DDS_LIVELINESS_LOST_STATUS, DDS_LivelinessLostStatus;
DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_OfferedIncompatibleQosStatus;
DDS_PUBLICATION_MATCHED_STATUS, DDS_PublicationMatchedStatus;

See also

Status Kinds

6.10.2 Field Documentation

6.10.2.1 struct DDS_Listener DDS_DataWriterListener::as_listener

<<cert>> The superclass instance of this DDS_DataWriterListener.

6.10.2.2 DDS_DataWriterListener_OfferedDeadlineMissedCallback DDS_DataWriterListener::on_offered_deadline_missed

<<*cert*>> Handles the DDS_OFFERED_DEADLINE_MISSED_STATUS status.

This callback is called when the deadline that the DDS_DataWriter has committed through its DEADLINE qos policy was not respected for a specific instance. This callback is called for each deadline period elapsed during which the DDS_DataWriter failed to provide data for an instance.

6.10.2.3 DDS_DataWriterListener_OfferedIncompatibleQosCallback DDS_DataWriterListener::on_offered_incompatible_qos

<<*cert*>> Handles the DDS_OFFERED_INCOMPATIBLE_QOS_STATUS status.

This callback is called when the DDS_DataWriterQos of the DDS_DataWriter was incompatible with what was requested by a DDS_DataReader. This callback is called when a DDS_DataWriter has discovered a DDS_DataReader for the same DDS_Topic and common partition, but with a requested QoS that is incompatible with that offered by the DDS_DataWriter.

6.10.2.4 DDS_DataWriterListener_LivelinessLostCallback DDS_DataWriterListener::on_liveliness_lost

<<*cert*>> Handles the DDS_LIVELINESS_LOST_STATUS status.

This callback is called when the liveliness that the DDS_DataWriter has committed through its LIVELINESS qos policy was not respected; this DDS_DataReader entities will consider the DDS_DataWriter as no longer "alive/active". This callback will not be called when an already not alive DDS_DataWriter simply remains not alive for another liveliness period.

6.10.2.5 DDS_DataWriterListener_PublicationMatchedCallback DDS_DataWriterListener::on_publication_matched

<<*cert*>> Handles the DDS_PUBLICATION_MATCHED_STATUS status.

This callback is called when the DDS_DataWriter has found a DDS_DataReader that matches the DDS_Topic, has a common partition and compatible QoS, or has ceased to be matched with a DDS_DataReader that was previously considered to be matched.

6.10.2.6 DDS_DataWriterListener_ReliableReaderActivityChangedCallback DDS_DataWriterListener::on_reliable_reader_activity_changed

<<*eXtension*>> <<*cert*>> A matched reliable reader has become active or become inactive.

A reliable writer considers a matched reliable reader to be active when it receives ACKNACK messages from the reader in response to its HEARTBEAT messages. The writer considers an active reader to have become inactive when it has sent a number of consecutive periodic HEARTBEATs equal to DDS_RtpsReliableWriterProtocol_t::max_heartbeat_retries without receiving an ACKNACK from the reader. An inactive reader becomes active once the writer receives an ACKNACK from it.

This callback is called when the transition from active to inactive, and vice versa, has occurred for a matched reliable reader.

6.10.2.7 DDS_DataWriterListener_ReliableSampleUnacknowledgedCallback DDS_DataWriterListener::on_reliable_sample_unacknowledged

<<*eXtension*>> <<*cert*>> A matched reliable reader has become active or become inactive.

A reliable writer considers a matched reliable reader to be active when it receives ACKNACK messages from the reader in response to its HEARTBEAT messages. The writer considers an active reader to have become inactive when it has sent a number of consecutive periodic HEARTBEATs equal to DDS_RtpsReliableWriterProtocol_t::max_heartbeat_retries without receiving an ACKNACK from the reader. An inactive reader becomes active once the writer receives an ACKNACK from it.

This callback is called when the transition from active to inactive, and vice versa, has occurred for a matched reliable reader.

6.11 DDS_DataWriterProtocolQosPolicy Struct Reference

<<eXtension>> *<<cert>>* Protocol that applies only to DDS_DataWriter instances.

Data Fields

- DDS_UnsignedLong rtps_object_id

  The RTPS Object ID.
- struct DDS_RtpsReliableWriterProtocol_t rtps_reliable_writer

  The reliable protocol defined in RTPS.

6.11.1 Detailed Description

<<eXtension>> *<<cert>>* Protocol that applies only to DDS_DataWriter instances.

This QoS policy is an extension to the DDS standard.

Entity:

DDS_DataWriter

Properties:

RxO = N/A
Changeable = NO

6.11.2 Field Documentation

6.11.2.1 DDS_UnsignedLong DDS_DataWriterProtocolQosPolicy::rtps_object_id

The RTPS Object ID.

This value is used to determine the RTPS object ID of a data writer according to the DDS-RTPS Interoperability Wire Protocol.

Only the last 3 bytes are used; the most significant byte is ignored.

[default] DDS_RTPS_AUTO_ID

[range] [0,0x00ffffff]

6.11.2.2 struct DDS_RtpsReliableWriterProtocol_t DDS_DataWriterProtocolQosPolicy::rtps_reliable_writer

The reliable protocol defined in RTPS.

[default] heartbeat_period 3.0 seconds;

heartbeats_per_max_samples: 1;

max_send_window: minimum(256, ::DDS_HistoryQosPolicy::depth * ::DDS_ResourceLimitsQosPolicy::max_instances);

max_heartbeat_retries: DDS_LENGTH_UNLIMITED;

first_write_sequence_number: {0,1};

6.12 DDS_DataWriterQos Struct Reference

<<cert>> QoS policies supported by a DDS_DataWriter entity.

Data Fields

- struct DDS_DeadlineQosPolicy deadline

  Deadline policy, DEADLINE.
- struct DDS_LivelinessQosPolicy liveliness

  Liveliness policy, LIVELINESS.
- struct DDS_HistoryQosPolicy history

  History policy, HISTORY.
- struct DDS_ResourceLimitsQosPolicy resource_limits

  Resource limits policy, RESOURCE_LIMITS.
- struct DDS_OwnershipQosPolicy ownership

  Ownership policy, OWNERSHIP.
- struct DDS_OwnershipStrengthQosPolicy ownership_strength

  Ownership strength policy, OWNERSHIP_STRENGTH.
- struct DDS_ReliabilityQosPolicy reliability

  Reliability policy, RELIABILITY.
- struct DDS_DurabilityQosPolicy durability

  <<eXtension>> Durability policy, DURABILITY.
- struct DDS_DestinationOrderQosPolicy destination_order

  <<eXtension>> Destination Order policy, DESTINATION_ORDER.
- struct DDS_DataWriterProtocolQosPolicy protocol

  <<eXtension>> DDS_DataWriter protocol policy, DATA_WRITER_PROTOCOL
- struct DDS_TransportQosPolicy transport

  Transport policy, TRANSPORT. Only unicast transports are supported.
- struct DDS_DataWriterResourceLimitsQosPolicy writer_resource_limits

  Writer resource limits policy, DataWriter Resource Limits.

6.12.1 Detailed Description

<<cert>> QoS policies supported by a DDS_DataWriter entity.

The DDS_DataWriterQos controls the resources allocated by a DDS_DataWriter. Each DDS_DataWriter allocates its own resources, even DataWriters of the same DDS_Topic. For this reason is it advised to limit the number of DataWriters per topic.

You must set certain members in a consistent manner:

- DDS_DataWriterQos::DDS_HistoryQosPolicy::depth <= DDS_DataWriterQos::DDS_ResourceLimitsQosPolicy-::max_samples_per_instance

- DDS_DataWriterQos::DDS_ResourceLimitsQosPolicy::max_samples_per_instance <= DDS_DataWriterQos::D-DS_ResourceLimitsQosPolicy::max_samples

If any of the above are not true DDS_Publisher_create_datawriter will fail and return NULL.

Entity:

DDS_DataWriter

See also

QoS Policies allowed ranges within each Qos.

6.12.2 Field Documentation

6.12.2.1 struct DDS_DeadlineQosPolicy DDS_DataWriterQos::deadline

Deadline policy, DEADLINE.

6.12.2.2 struct DDS_LivelinessQosPolicy DDS_DataWriterQos::liveliness

Liveliness policy, LIVELINESS.

6.12.2.3 struct DDS_HistoryQosPolicy DDS_DataWriterQos::history

History policy, HISTORY.

6.12.2.4 struct DDS_ResourceLimitsQosPolicy DDS_DataWriterQos::resource_limits

Resource limits policy, RESOURCE_LIMITS.

6.12.2.5 struct DDS_OwnershipQosPolicy DDS_DataWriterQos::ownership

Ownership policy, OWNERSHIP.

6.12.2.6 struct DDS_OwnershipStrengthQosPolicy DDS_DataWriterQos::ownership_strength

Ownership strength policy, OWNERSHIP_STRENGTH.

6.12.2.7 struct DDS_ReliabilityQosPolicy DDS_DataWriterQos::reliability

Reliability policy, RELIABILITY.

6.12.2.8 struct DDS_DurabilityQosPolicy DDS_DataWriterQos::durability

<<*eXtension*>> Durability policy, DURABILITY.

6.12.2.9 struct DDS_DestinationOrderQosPolicy DDS_DataWriterQos::destination_order

<<*eXtension*>> Destination Order policy, DESTINATION_ORDER.

6.12.2.10 struct DDS_DataWriterProtocolQosPolicy DDS_DataWriterQos::protocol

<<*eXtension*>> DDS_DataWriter protocol policy, DATA_WRITER_PROTOCOL

6.12.2.11 struct DDS_TransportQosPolicy DDS_DataWriterQos::transport

Transport policy, TRANSPORT. Only unicast transports are supported.

6.12.2.12 struct DDS_DataWriterResourceLimitsQosPolicy DDS_DataWriterQos::writer_resource_limits

Writer resource limits policy, DataWriter Resource Limits.

6.13 DDS_DataWriterResourceLimitsQosPolicy Struct Reference

<<*cert*>> Resource limits that apply only to DDS_DataWriter instances.

Data Fields

- DDS_Long max_remote_readers

  The maximum number of remote readers for which the writer will maintain state and communication.
- DDS_Long max_routes_per_reader

The maximum number of routes the writer will send to per matched reader.

6.13.1 Detailed Description

<<*cert*>> Resource limits that apply only to DDS_DataWriter instances.

This QoS policy is an extension to the DDS standard.

Entity:

DDS_DataWriter

Properties:

RxO = N/A
Changeable = NO

6.13.2 Field Documentation

6.13.2.1 DDS_Long DDS_DataWriterResourceLimitsQosPolicy::max_remote_readers

The maximum number of remote readers for which the writer will maintain state and communication.

The DDS_DataWriterResourceLimitsQosPolicy::max_remote_readers resource-limit limits the maximum number of remote DataReaders with which the DDS_DataWriter will maintain state and communication.

[default] 16

[range] [1, 100000000]

6.13.2.2 DDS_Long DDS_DataWriterResourceLimitsQosPolicy::max_routes_per_reader

The maximum number of routes the writer will send to per matched reader.

NOTE: If this resource limit is exceeded when matching a DDS_DataWriter with a DDS_DataReader, the two entities are not matched.

Each DDS_DataWriter maintains information about the state of its peer DataReaders (those it has matched with). Part of this state is which locators (or destination addresses) it should use to send data to a particular DDS_DataReader. The DDS_DataWriterResourceLimitsQosPolicy::max_routes_per_reader resource-limit limits the number of routes that can be saved per DDS_DataReader.

This resource-limit is shared across all matched DataReaders per DDS_DataWriter. Thus, if DDS_DataWriterResourceLimitsQosPolicy::max_remote_readers is 2 and DDS_DataWriterResourceLimitsQosPolicy::max_routes_per_reader is 4, a total of 8 routes can be saved for both DataReaders. One DDS_DataReader may have 6 routes and the other 2.

[default] 4

[range] [1, 2000]

6.14 DDS_DeadlineQosPolicy Struct Reference

<<*cert*>> Expresses the maximum duration (deadline) within which an instance is expected to be updated.

Data Fields

- struct DDS_Duration_t period

  Duration of the deadline period.

6.14.1 Detailed Description

<<*cert*>> Expresses the maximum duration (deadline) within which an instance is expected to be updated.

DDS_DataReader expects a new sample updating the value of each instance at least once every `period`. DDS_DataWriter indicates that the application commits to write a new value (using the DDS_DataWriter) for each instance managed by the DDS_DataWriter at least once every `period`.

If a DDS_DataReader stops receiving samples, it will take at most (`period` + (`period` / 8)) to detect that the deadline has been missed. The deadline period starts again when the next sample is received.

Entity:

DDS_Topic, DDS_DataReader, DDS_DataWriter

Status:

DDS_OFFERED_DEADLINE_MISSED_STATUS, DDS_REQUESTED_DEADLINE_MISSED_STATUS, DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS

Properties:

RxO = YES
Changeable = NO

6.14.2 Usage

This policy is useful for cases where a DDS_Topic is expected to have each instance updated periodically. On the publishing side this setting establishes a contract that the application must meet. On the subscribing side the setting establishes a minimum requirement for the remote publishers that are expected to supply the data values.

When RTI Connext Micro 'matches' a DDS_DataWriter and a DDS_DataReader it checks whether the settings are compatible (i.e., *offered deadline* <= *requested deadline*); if they are not, the two entities are informed (via the DDS_Listener mechanism) of the incompatibility of the QoS settings and communication will not occur.

Assuming that the reader and writer ends have compatible settings, the fulfilment of this contract is monitored by RTI Connext Micro and the application is informed of any violations by means of the proper DDS_Listener.

6.14.3 Compatibility

The value offered is considered compatible with the value requested if and only if the inequality *offered period* <= *requested period* evaluates to 'TRUE'.

6.14.4 Field Documentation

6.14.4.1 struct DDS_Duration_t DDS_DeadlineQosPolicy::period

Duration of the deadline period.

[default] DDS_DURATION_INFINITE

[range] [1 nanosec, 1 year] or DDS_DURATION_INFINITE

6.15 DDS_DestinationOrderQosPolicy Struct Reference

Destination order policies that affect the ordering of subsribed data.

Data Fields

- DDS_DestinationOrderQosPolicyKind kind

  The kind of destination ordering that applies to subscribed data.
- struct DDS_Duration_t source_timestamp_tolerance

  <<eXtension>> Allowed tolerance between source timestamps of consecutive samples.

6.15.1 Detailed Description

Destination order policies that affect the ordering of subsribed data.

6.15.2 Field Documentation

6.15.2.1 DDS_DestinationOrderQosPolicyKind DDS_DestinationOrderQosPolicy::kind

The kind of destination ordering that applies to subscribed data.

[default] DDS_BY_RECEPTION_TIMESTAMP_DESTINATIONORDER_QOS

6.15.2.2 struct DDS_Duration_t DDS_DestinationOrderQosPolicy::source_timestamp_tolerance

<<eXtension>> Allowed tolerance between source timestamps of consecutive samples.

When a DDS_DataWriter sets DDS_DestinationOrderQosPolicy to DDS_BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS, when writing a sample, its timestamp must not be less than the timestamp of the previously written sample. However, if it is less than the timestamp of the previously written sample but the difference is less than this tolerance, the sample will use the previously written sample's timestamp as its timestamp. Otherwise, if the difference is greater than this tolerance, the write will fail.

DDS_BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS is not support on the DDS_DataReader.

[default] 100 milliseconds for DDS_DataWriter

[range] [1 nanosec, 1 year]

6.16 DDS_DiscoveryComponent Struct Reference

Specifies the discovery plugin the DDS_DomainParticipant should use.

Data Fields

- RT_ComponentFactoryId_T name

  Specifies the name of a registered discovery plugin the DDS_DomainParticipant should use.

6.16.1 Detailed Description

Specifies the discovery plugin the DDS_DomainParticipant should use.

6.16.2 Field Documentation

6.16.2.1 RT_ComponentFactoryId_T DDS_DiscoveryComponent::name

Specifies the name of a registered discovery plugin the DDS_DomainParticipant should use.

```
This attribute specifies the name of the plug-in to be
used for DDS Discovery. The maximum length of the name
is RT_MAX_FACTORY_NAME excluding the NUL termination.
```

6.17 DDS_DiscoveryQosPolicy Struct Reference

<<cert>> Specifies the attributes required to discover participants in the domain.

Data Fields

- struct DDS_StringSeq initial_peers

  Determines the initial list of peers that will be contacted by the Discovery mechanism to send announcements about the presence of this participant.
- struct DDS_StringSeq enabled_transports

  The transports available for use by the Discovery mechanism.
- struct DDS_DiscoveryComponent discovery

  The name of the discovery plugin that will be used to create a discovery plugin for this DDS_DomainParticipant.
- DDS_Boolean accept_unknown_peers

  Whether to accept a new participant that is not in the initial peer list.

6.17.1 Detailed Description

<<cert>> Specifies the attributes required to discover participants in the domain.

Entity:

DDS_DomainParticipant

Properties:

RxO = N/A
Changeable = NO

6.17.2 Usage

This extended Qos is used to define how participants find each other on a network.

6.17.3 Field Documentation

6.17.3.1 struct DDS_StringSeq DDS_DiscoveryQosPolicy::initial_peers

Determines the initial list of peers that will be contacted by the Discovery mechanism to send announcements about the presence of this participant.

If there is a remote peer DDS_DomainParticipant such as is described this list, it will become aware of this participant and will engage in the Discovery protocol to exchange meta-data with this participant.

Each element of this list must be a `peer` descriptor in the proper format.

[default] Empty sequence

[range] Sequence of arbitrary length.

6.17.3.2 struct DDS_StringSeq DDS_DiscoveryQosPolicy::enabled_transports

The transports available for use by the Discovery mechanism.

Only these transports can be used by the discovery mechanism to send meta-traffic via the builtin endpoints (built-in DDS_DataReader and DDS_DataWriter).

Also determines the IP addresses on which the Discovery mechanism will listen for meta-traffic. These along with the `domain_id` and `participant_id` determine the IP locators on which the Discovery mechanism can receive meta-data.

The memory for the strings in this sequence is managed according to the conventions described in String Support. In particular, be careful to avoid a situation in which RTI Connext Micro allocates a string on your behalf and you then reuse that string in such a way that RTI Connext Micro believes it to have more memory allocated to it than it actually does.

[default] Empty sequence.

When an empty sequence is passed to DDS_DomainParticipantFactory_create_participant in the DDS_DomainParticipantQos, the domain participant will automatically add the following locators to listen for discovery traffic on: "_udp://" and "_udp://239.255.0.1"

[range] Sequence of non-null,non-empty strings.

6.17.3.3 struct DDS_DiscoveryComponent DDS_DiscoveryQosPolicy::discovery

The name of the discovery plugin that will be used to create a discovery plugin for this DDS_DomainParticipant.

The DDS_DomainParticipantFactory will search through a list of registered discovery plugins. If it finds a registered discovery factory with a name matching this factory name, it will use that discovery plugin to create a new instance of the discovery plugin and register the plugin with the DDS_DomainParticipant.

If it does not find a registered discovery factory with this name, it will return DDS_RETCODE_PRECONDITION_NOT_MET

Precondition

One or more discovery plugins must have been previously registered with the DDS_DomainParticipantFactory, each with an associated name.

6.17.3.4 DDS_Boolean DDS_DiscoveryQosPolicy::accept_unknown_peers

Whether to accept a new participant that is not in the initial peer list.

If DDS_BOOLEAN_FALSE, the participant will only communicate with those in the initial peer list

If DDS_BOOLEAN_TRUE, the participant will also communicate with all discovered remote participants.

[default] DDS_BOOLEAN_TRUE

6.18 DDS_DomainParticipantFactoryQos Struct Reference

<<cert>> QoS policies supported by a DDS_DomainParticipantFactory.

Data Fields

- struct DDS_EntityFactoryQosPolicy entity_factory

  Entity factory policy, ENTITY_FACTORY.
- struct DDS_SystemResourceLimitsQosPolicy resource_limits

  <<eXtension>> *System resource limits, SYSTEM_RESOURCE_LIMITS.*

6.18.1 Detailed Description

<<cert>> QoS policies supported by a DDS_DomainParticipantFactory.

Entity:

DDS_DomainParticipantFactory

See also

QoS Policies and allowed ranges within each Qos.

6.18.2 Field Documentation

6.18.2.1 struct DDS_EntityFactoryQosPolicy DDS_DomainParticipantFactoryQos::entity_factory

Entity factory policy, ENTITY_FACTORY.

6.18.2.2 struct DDS_SystemResourceLimitsQosPolicy DDS_DomainParticipantFactoryQos::resource_limits

<<eXtension>> System resource limits, SYSTEM_RESOURCE_LIMITS.

6.19 DDS_DomainParticipantListener Struct Reference

<<interface>> *<<cert>>* Listener for participant status.

Data Fields

- struct DDS_TopicListener as_topiclistener
  A superclass instance of this DDS_DomainParticipantListener.
- struct DDS_PublisherListener as_publisherlistener
  A superclass instance of this DDS_DomainParticipantListener.
- struct DDS_SubscriberListener as_subscriberlistener
  A superclass instance of this DDS_DomainParticipantListener.

6.19.1 Detailed Description

<<interface>> *<<cert>>* Listener for participant status.

Entity:

DDS_DomainParticipant

Status:

Status Kinds

This is the interface that can be implemented by an application-provided class and then registered with the DDS_DomainParticipant such that the application can be notified by RTI Connext Micro of relevant status changes.

The DDS_DomainParticipantListener interface extends all other Listener interfaces and has no additional operation beyond the ones defined by the more general listeners.

The purpose of the DDS_DomainParticipantListener is to be the listener of last resort that is notified of all status changes not captured by more specific listeners attached to the DDS_DomainEntity objects. When a relevant status change occurs, RTI Connext Micro will first attempt to notify the listener attached to the concerned DDS_DomainEntity if one is installed. Otherwise, RTI Connext Micro will notify the Listener attached to the DDS_DomainParticipant.

Important: Because a DDS_DomainParticipantListener may receive callbacks pertaining to many different entities, it is possible for the same listener to receive multiple callbacks simultaneously in different threads. (Such is not the case for listeners of other types.) It is therefore critical that users of this listener provide their own protection for any thread-unsafe activities undertaken in a DDS_DomainParticipantListener callback.

See also

DDS_Listener

6.19.2 Field Documentation

6.19.2.1 struct DDS_TopicListener DDS_DomainParticipantListener::as_topiclistener

A superclass instance of this DDS_DomainParticipantListener.

6.19.2.2 struct DDS_PublisherListener DDS_DomainParticipantListener::as_publisherlistener

A superclass instance of this DDS_DomainParticipantListener.

6.19.2.3 struct DDS_SubscriberListener DDS_DomainParticipantListener::as_subscriberlistener

A superclass instance of this DDS_DomainParticipantListener.

6.20 DDS_DomainParticipantQos Struct Reference

<<cert>> QoS policies supported by a DDS_DomainParticipant entity.

Data Fields

- struct DDS_EntityFactoryQosPolicy entity_factory
  Entity factory policy, ENTITY_FACTORY.
- struct DDS_DiscoveryQosPolicy discovery
  <<eXtension>> *Discovery policy, DISCOVERY.*
- struct DDS_DomainParticipantResourceLimitsQosPolicy resource_limits
  <<eXtension>> *Domain participant resource limits policy, DOMAIN_PARTICIPANT_RESOURCE_LIMITS.*
- struct DDS_EntityNameQosPolicy participant_name
  <<eXtension>> *The participant name. ENTITY_NAME*
- struct DDS_WireProtocolQosPolicy protocol
  <<eXtension>> *Wire Protocol policy, WIRE_PROTOCOL.*
- struct DDS_TransportQosPolicy transports
  <<eXtension>> *Available transports for all communication to and from this DomainParticipant.*
- struct DDS_UserTrafficQosPolicy user_traffic
  <<eXtension>> *Transports enabled for user traffic.*

6.20.1 Detailed Description

<<cert>> QoS policies supported by a DDS_DomainParticipant entity.

The Qos policy must be set in a consistent manner, DDS_DomainParticipantFactory_create_participant will fail.

Entity:

DDS_DomainParticipant

See also

QoS Policies and allowed ranges within each Qos.

6.20.2 Field Documentation

6.20.2.1 struct DDS_EntityFactoryQosPolicy DDS_DomainParticipantQos::entity_factory

Entity factory policy, ENTITY_FACTORY.

6.20.2.2 struct DDS_DiscoveryQosPolicy DDS_DomainParticipantQos::discovery

<<*eXtension*>> Discovery policy, DISCOVERY.

6.20.2.3 struct DDS_DomainParticipantResourceLimitsQosPolicy DDS_DomainParticipantQos::resource_limits

<<*eXtension*>> Domain participant resource limits policy, DOMAIN_PARTICIPANT_RESOURCE_LIMITS.

6.20.2.4 struct DDS_EntityNameQosPolicy DDS_DomainParticipantQos::participant_name

<<*eXtension*>> The participant name. ENTITY_NAME

6.20.2.5 struct DDS_WireProtocolQosPolicy DDS_DomainParticipantQos::protocol

<<*eXtension*>> Wire Protocol policy, WIRE_PROTOCOL.

The wire protocol (RTPS) attributes associated with the participant.

6.20.2.6 struct DDS_TransportQosPolicy DDS_DomainParticipantQos::transports

<<*eXtension*>> Available transports for all communication to and from this DomainParticipant.

6.20.2.7 struct DDS_UserTrafficQosPolicy DDS_DomainParticipantQos::user_traffic

<<*eXtension*>> Transports enabled for user traffic.

6.21 DDS_DomainParticipantResourceLimitsQosPolicy Struct Reference

<<*cert*>> Resource limits that apply only to DDS_DomainParticipant instances.

Data Fields

- DDS_Long local_writer_allocation

  The maximum number of local Writers that can be created.
- DDS_Long local_reader_allocation

  The maximum number of local Readers that can be created.
- DDS_Long local_publisher_allocation

  The maximum number of local Publishers that can be created.
- DDS_Long local_subscriber_allocation

  The maximum number of local Subscribers that can be created.
- DDS_Long local_topic_allocation

  The maximum number of topics that can be created.
- DDS_Long local_type_allocation

  The maximum number of local types that can be registered with this DomainParticipant.
- DDS_Long remote_participant_allocation

  The maximum number of remote participants that can be discovered by this DomainParticipant.
- DDS_Long remote_writer_allocation

  The maximum number of remote writers that can be discovered by this DomainParticipant.
- DDS_Long remote_reader_allocation

  The maximum number of remote readers that can be discovered by this DomainParticipant.
- DDS_Long matching_writer_reader_pair_allocation

  The maximum number of matching local writer and remote/local reader pairs.
- DDS_Long max_receive_ports

  Maximum number of addresses that can be listened on.
- DDS_Long max_destination_ports

  Maximum number of unique destination addresses that a participant can reach.

6.21.1 Detailed Description

<<*cert*>> Resource limits that apply only to DDS_DomainParticipant instances.

This QoS policy is an extension to the DDS standard.

Entity:

DDS_DomainParticipant

Properties:

RxO = N/A
Changeable = NO

6.21.2 Field Documentation

6.21.2.1 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::local_writer_allocation

The maximum number of local Writers that can be created.

This resource-limit limits the maximum number of DDS_DataWriter entities that can be created with the DDS_Publisher_create_datawriter() API within a single DDS_DomainParticipant across all DDS_Publisher entities.

[default] 1

6.21.2.2 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::local_reader_allocation

The maximum number of local Readers that can be created.

This resource-limit limits the maximum number of DDS_DataReader entities that can be created with the DDS_Subscriber_create_datareader() API within a single DDS_DomainParticipant across all DDS_Subscriber entities.

[default] 1

6.21.2.3 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::local_publisher_allocation

The maximum number of local Publishers that can be created.

This resource-limit limits the maximum number of DDS_Publisher entities that can be created within a DDS_DomainParticipant with the DDS_DomainParticipant_create_publisher() API.

[default] 1

6.21.2.4 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::local_subscriber_allocation

The maximum number of local Subscribers that can be created.

This resource-limit limits the maximum number of DDS_Subscriber entities that can be created within a DDS_DomainParticipant with the DDS_DomainParticipant_create_subscriber() API.

[default] 1

6.21.2.5 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::local_topic_allocation

The maximum number of topics that can be created.

This resource-limit limits the maximum number of unique DDS_Topic's that can be created within a single DDS_DomainParticipant with the DDS_DomainParticipant_create_topic() API.

[default] 1

6.21.2.6 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::local_type_allocation

The maximum number of local types that can be registered with this DomainParticipant.

This resource-limit limits the maximum number of unique NDDS_Type_Plugin's that can be registered locally within a single DDS_DomainParticipant with the FooTypeSupport_register_type() API.

[default] 1

6.21.2.7 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::remote_participant_allocation

The maximum number of remote participants that can be discovered by this DomainParticipant.

This resource-limit limits the maximum number of DDS_DomainParticipant's that can be discovered by a local DDS_DomainParticipant.

The DPSE discovery plugin limits the discovery to those that have been asserted based on the DomainParticipant name using DPSE_RemoteParticipant_assert

[default] 1

6.21.2.8 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::remote_writer_allocation

The maximum number of remote writers that can be discovered by this DomainParticipant.

This resource-limit limits the maximum number of remote DDS_DataWriter entities that can be discovered by a local DDS_DomainParticipant. Note that a remote DDS_DataWriter cannot be discovered until its parent DDS_DomainParticipant has been discovered.

[default] 1

6.21.2.9 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::remote_reader_allocation

The maximum number of remote readers that can be discovered by this DomainParticipant.

This resource-limit limits the maximum number of remote DDS_DataReader entities that can be discovered by a local DomainParticipant. Note that a remote DDS_DataReader cannot be discovered until its parent DDS_DomainParticipant has been discovered.

[default] 1

6.21.2.10 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::matching_writer_reader_pair_allocation

The maximum number of matching local writer and remote/local reader pairs.

Note: This QoS property only applies to endpoints matched over RTPS-based transports, such as UDPv4.

When RTI Connext Micro discovers DDS entities using a DDS discovery protocol, such as DPSE (Dynamic Participant Static Endpoint) or DPDE (Dynamic Participant Dynamic Endpoint), it matches the discovered entities with the locally created DataReaders and DataWriters. For each match, either a local DataWriter matching a remote DataReader or a local DataReader matching a remote DataWriter, internal state must be managed.

A match, in this context, is defined as being able to receive packets sent from a DDS_DataReader to a DDS_DataWriter, or from a DDS_DataWriter to a DDS_DataReader:

- A data-writer needs one resource for each DDS_DataReader it matches.
- A data-reader needs one resource for each DDS_DataWriter it matches.

This resource-limit limits the maximum number of matches that can be managed. When this resource limit is reached no further matching can occur, even if new DDS_DataReader's and DDS_DataWriter's are discovered.

A simple rule of thumb is to set this resource limit to:

- (local_reader_allocation + local_writer_allocation) $*$ max_remote_participants.

This assumes that each DDS_DataReader and DDS_DataWriter matches one DDS_DataWriter and DDS_DataReader respectively for each discovered participant. However, this may over-allocate resources.

[default] 32

6.21.2.11 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::max_receive_ports

Maximum number of addresses that can be listened on.

RTI Connext Micro listens for discovery data and user data on different transports as specified by the user. This resource-limit limits the maximum number of addresses that can be listened on. As a rule of thumb, this resource limit can be determined as follows: - A user-traffic unicast port counts as 1. - A meta-traffic unicast port counts as 1. - A user-traffic multicast address counts as 1. - A meta-traffic multicast address counts as 1.

Consider the typical case with the UDP transport, using unicast and multicast for discovery data and unicast for user data:

- Discovery data needs two ports.
- User data needs one port.

Thus, this requires 3 resources.

Consider a case with the UDP transport, using unicast and multicast for discovery data and user data: - Discovery data needs two ports. - User data needs two ports.

Thus, this requires 4 resources.

Consider a case with the UDP transport, using unicast and 2 multicast addresses each for discovery data and user data:

- Discovery data needs three ports.
- User data needs three ports.

Thus, this requires 6 resources.

The default value has been set to cover most cases. This resource-limit should only be changed if a lot of interfaces are supported (increase it) or if memory is extremely limited (decrease it).

[default] 8

6.21.2.12 DDS_Long DDS_DomainParticipantResourceLimitsQosPolicy::max_destination_ports

Maximum number of unique destination addresses that a participant can reach.

RTI Connext Micro maintains a table of destination addresses which it can send to based on information queried from the registered transports. This resource limit is difficult to give a precise rule for how to determine since it is transport dependent. However, as a rule of thumb the following is true: - Each network that can be reached directly by the transport requires 1 resource. - If multicast is supported it counts as 1 resource. - If the transport is the default transport (by default UDP is) it counts as one address

Consider a typical Linux computer with 2 network interfaces, eth0 and lo, with support for multicast:

- 1 for the eth0 network.
- 1 for the lo network.
- 1 for multicast.
- 1 for being the default transport.

This requires 4 resources.

Consider a typical embedded computer with 1 network interfaces, eth0, with no support for multicast:

- 1 for the eth0 network.
- 1 for multicast.
- 1 for being the default transport.

This requires 3 resources.

The default value has been set to cover most cases. This resource-limit should only be changed if a lot of interfaces are supported (increase it) or if memory is extremely limited (decrease it).

[default] 8

6.22 DDS_DurabilityQosPolicy Struct Reference

Durability properties that affect the life-cycle of published data.

Data Fields

- DDS_DurabilityQosPolicyKind kind

  The kind of durability that applies to published data.

6.22.1 Detailed Description

Durability properties that affect the life-cycle of published data.

6.22.2 Field Documentation

6.22.2.1 DDS_DurabilityQosPolicyKind DDS_DurabilityQosPolicy::kind

The kind of durability that applies to published data.

6.23 DDS_Duration_t Struct Reference

Type for *duration* representation.

Data Fields

- DDS_Long sec

  seconds
- DDS_UnsignedLong nanosec

  nanoseconds

6.23.1 Detailed Description

Type for *duration* representation.

Represents a time interval.

6.23.2 Field Documentation

6.23.2.1 DDS_Long DDS_Duration_t::sec

seconds

6.23.2.2 DDS_UnsignedLong DDS_Duration_t::nanosec

nanoseconds

6.24 DDS_EntityFactoryQosPolicy Struct Reference

<<*cert*>> A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types.

Data Fields

- DDS_Boolean autoenable_created_entities

  Specifies whether the entity acting as a factory automatically enables the instances it creates.

6.24.1 Detailed Description

<<*cert*>> A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types.

Entity:

DDS_DomainParticipantFactory, DDS_DomainParticipant, DDS_Publisher, DDS_Subscriber

Properties:

RxO = NO
Changeable = YES

6.24.2 Usage

This policy controls the behavior of the DDS_Entity as a factory for other entities.

This policy is mutable. A change in the policy affects only the entities created after the change; not the previously created entities.

The setting of `autoenable_created_entities` to DDS_BOOLEAN_TRUE indicates that the factory `create_<entity>` operation will automatically invoke the DDS_Entity_enable operation each time a new DDS_Entity is created. Therefore, the DDS_Entity returned by `create_<entity>` will already be enabled. A setting of DDS_BOOLEAN_FALSE indicates that the DDS_Entity will not be automatically enabled. The application will need to enable it explicitly by means of DDS_Entity_enable operation.

The default setting of `autoenable_created_entities` = DDS_BOOLEAN_TRUE means that, by default, it is not necessary to explicitly call DDS_Entity_enable on newly created entities.

6.24.3 Field Documentation

6.24.3.1 DDS_Boolean DDS_EntityFactoryQosPolicy::autoenable_created_entities

Specifies whether the entity acting as a factory automatically enables the instances it creates.

If DDS_BOOLEAN_TRUE, the factory will automatically enable each created entity. Otherwise, it will not.

[default] DDS_BOOLEAN_TRUE

6.25 DDS_EntityNameQosPolicy Struct Reference

<<*cert*>> Policy used to name an entity.

Data Fields

- char name [DDS_ENTITYNAME_QOS_NAME_MAX+1]

  The actual name string.

6.25.1 Detailed Description

<<*cert*>> Policy used to name an entity.

Entity:

DDS_DomainParticipant

Properties:

RxO = NO;
Changeable = NO

6.25.2 Usage

The purpose of this QoS is to allow the user to attach naming information to created DDS_Entity objects.

It can only be 255 characters in length.

It must be a NULL-terminated string allocated with DDS_String_alloc or DDS_String_dup.

If the user provides a non-NULL pointer when getting the QoS, then it should point to valid memory that can be written to, to avoid ungraceful failures.

6.25.3 Field Documentation

6.25.3.1 char DDS_EntityNameQosPolicy::name[DDS_ENTITYNAME_QOS_NAME_MAX+1]

The actual name string.

[default] "[ENTITY]"

[range] Null terminated string with length not exceeding 255. Can be NULL.

6.26 DDS_GUID_t Struct Reference

Type for *GUID* (Global Unique Identifier) representation.

Data Fields

- DDS_Octet value [16]

  A 16 byte array containing the GUID value.

6.26.1 Detailed Description

Type for *GUID* (Global Unique Identifier) representation.

Represents a 128 bit GUID.

6.26.2 Field Documentation

6.26.2.1 DDS_Octet DDS_GUID_t::value[16]

A 16 byte array containing the GUID value.

6.27 DDS_HistoryQosPolicy Struct Reference

<<cert>> Specifies the behavior of RTI Connext Micro in the case where the value of a sample changes (one or more times) before it can be successfully communicated to one or more existing subscribers.

Data Fields

- DDS_HistoryQosPolicyKind kind

  Specifies the kind of history to be kept.
- DDS_Long depth

  Specifies the number of samples to be kept, when the `kind` *is DDS_KEEP_LAST_HISTORY_QOS.*

6.27.1 Detailed Description

<<cert>> Specifies the behavior of RTI Connext Micro in the case where the value of a sample changes (one or more times) before it can be successfully communicated to one or more existing subscribers.

This `QosPolicy` controls whether RTI Connext Micro should deliver only the most recent value, attempt to deliver all intermediate values, or do something in between.

On the publishing side this policy controls the samples that should be maintained by the DDS_DataWriter on behalf of existing DDS_DataReader entities.

On the subscribing side it controls the samples that should be maintained until the application "takes" them from RTI Connext Micro.

RTI Connext Micro supports DDS_KEEP_LAST_HISTORY_QOS kind.

Entity:

DDS_Topic, DDS_DataReader, DDS_DataWriter

Properties:

RxO = NO
Changeable = NO

See also

DDS_ReliabilityQosPolicy
DDS_HistoryQosPolicy

6.27.2 Usage

This policy controls the behavior of RTI Connext Micro when the value of an instance changes before it is finally communicated to some of its existing DDS_DataReader entities.

For kind set to DDS_KEEP_LAST_HISTORY_QOS, RTI Connext Micro will only attempt to keep the latest values of the instance and discard the older ones. In this case, the value of `depth` regulates the maximum number of values (up to and including the most current one) RTI Connext Micro will maintain and deliver. The default (and most common setting) for depth is 1, indicating that only the most recent value should be delivered.

6.27.3 Consistency

The setting of HISTORY `depth` must be consistent with the RESOURCE_LIMITS `max_samples_per_instance`. For these two QoS to be consistent, they must verify that *depth* <= *max_samples_per_instance*.

See also

DDS_ResourceLimitsQosPolicy

6.27.4 Field Documentation

6.27.4.1 DDS_HistoryQosPolicyKind DDS_HistoryQosPolicy::kind

Specifies the kind of history to be kept.

[default] DDS_KEEP_LAST_HISTORY_QOS

6.27.4.2 DDS_Long DDS_HistoryQosPolicy::depth

Specifies the number of samples to be kept, when the `kind` is DDS_KEEP_LAST_HISTORY_QOS.

If a value other than 1 (the default) is specified, it should be consistent with the settings of the RESOURCE_LIMITS policy. That is:

depth <= DDS_ResourceLimitsQosPolicy::max_samples_per_instance

[default] 1

[range] [1,2147483647]

6.28 DDS_InconsistentTopicStatus Struct Reference

DDS_INCONSISTENT_TOPIC_STATUS

6.28.1 Detailed Description

DDS_INCONSISTENT_TOPIC_STATUS

Entity:

DDS_Topic

Listener:

DDS_TopicListener

A remote DDS_Topic will be inconsistent with the locally created DDS_Topic if the type name of the two topics are different.

6.29 DDS_Listener Struct Reference

<<interface>> Abstract base class for all Listener interfaces.

Data Fields

- void * listener_data

  A place for listener implementors to keep a pointer to data that may be needed by their listener.

6.29.1 Detailed Description

<<interface>> Abstract base class for all Listener interfaces.

Entity:

DDS_Entity

QoS:

QoS Policies

Status:

Status Kinds

All the supported kinds of concrete DDS_Listener interfaces (one per concrete DDS_Entity type) derive from this root and add functions whose prototype depends on the concrete Listener.

Listeners provide a way for RTI Connext Micro to asynchronously alert the application when there are relevant status changes.

Almost every application will have to implement listener interfaces.

Each dedicated listener presents a list of operations that correspond to the relevant communication status changes to which an application may respond.

The same DDS_Listener instance may be shared among multiple entities if you so desire. Consequently, the provided parameter contains a reference to the concerned DDS_Entity.

6.29.2 Access to Plain Communication Status

The general mapping between the plain communication statuses (see Status Kinds) and the listeners' operations is as follows:

- For each communication status, there is a corresponding operation whose name is `on_<communication_status>`(), which takes a parameter of type `<communication_status>` as listed in Status Kinds.

- `on_<communication_status>` is available on the relevant DDS_Entity as well as those that embed it, as expressed in the following figure:

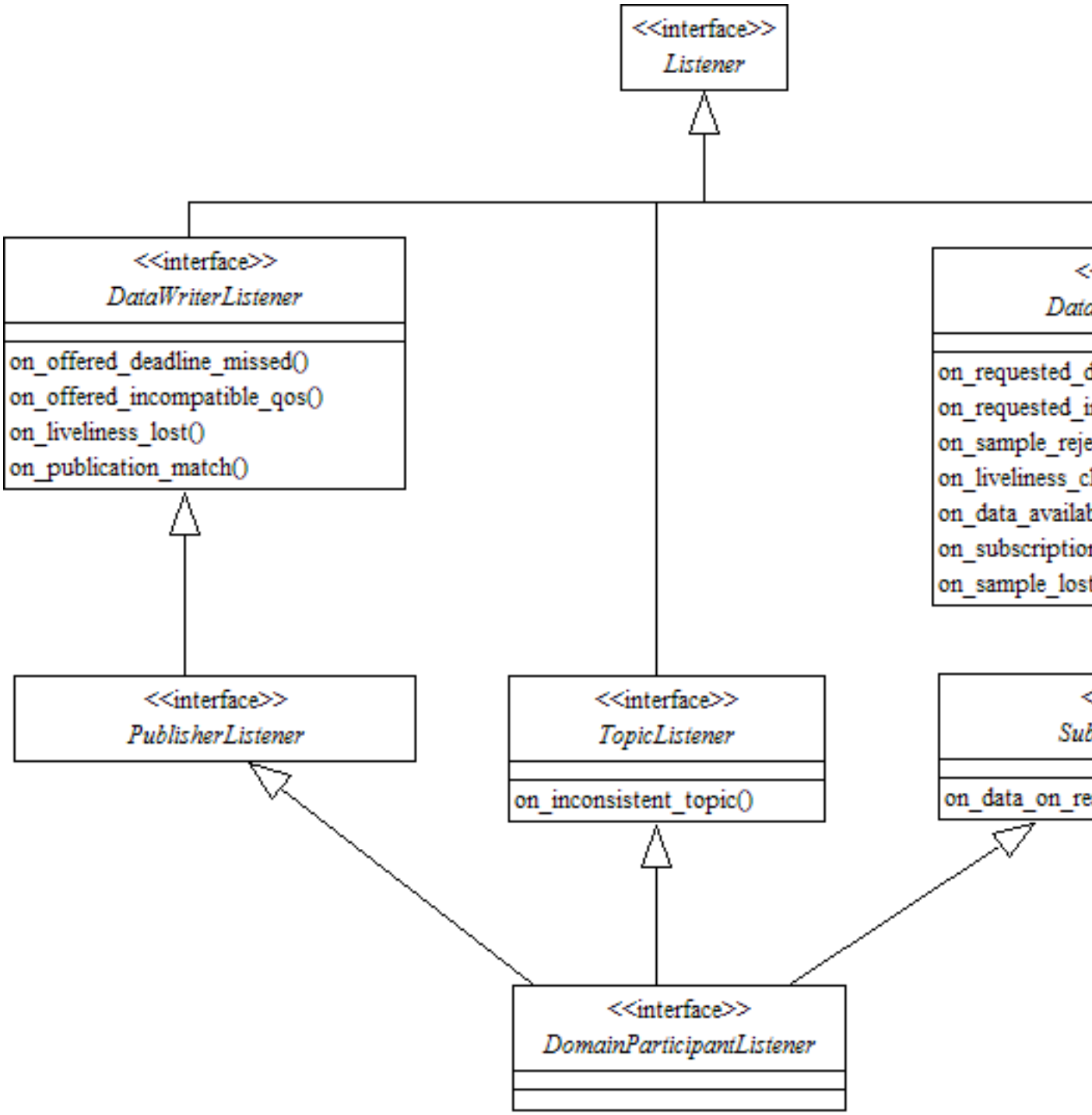

Figure 11: Hierarchical listener processing. The most *specific* relevant enabled listener is called.

- When the application attaches a listener on an entity, it must set a mask. The mask indicates to RTI Connext Micro which operations are enabled within the listener (cf. operation DDS_Entity set_listener()).
- When a plain communication status changes, RTI Connext Micro triggers the most specific relevant listener operation that is enabled. In case the most specific relevant listener operation corresponds to an application-installed 'nil' listener the operation will be considered handled by a NO-OP operation that does not reset the communication status.

This behavior allows the application to set a default behavior (e.g., in the listener associated with the DDS_DomainParticipant) and to set dedicated behaviors only where needed.

6.29.3 Access to Read Communication Status

The two statuses related to data arrival are treated slightly differently. Since they constitute the core purpose of the Data Distribution Service, there is no need to provide a default mechanism (as is done for the plain communication statuses above).

The rule is as follows. Each time the read communication status changes:

- First, RTI Connext Micro tries to trigger the DDS_SubscriberListener::on_data_on_readers with a parameter of the related DDS_Subscriber;
- If this does not succeed (there is no listener or the operation is not enabled), RTI Connext Micro tries to trigger DDS_DataReaderListener::on_data_available on all the related DDS_DataReaderListener objects, with a parameter of the related DDS_DataReader.

The rationale is that either the application is interested in relations among data arrivals and it must use the first option or it wants to treat each DDS_DataReader independently and it may choose the second option (and then get the data by calling FooDataReader_read_next_sample or FooDataReader_take_next_sample on the related DDS_DataReader).

Note that if DDS_SubscriberListener::on_data_on_readers is called, RTI Connext Micro will *not* try to call DDS_DataReaderListener::on_data_available.

The operations that are allowed in DDS_Listener callbacks depend the DDS_Entity to which the DDS_Listener is attached – or in the case of a DDS_DataWriter of DDS_DataReader listener, on the parent DDS_Publisher or DDS_Subscriber.

The following operations are *not* allowed:

- Within any listener callback, deleting the entity to which the DDS_Listener is attached
- Within a DDS_Topic listener callback, any operations on any subscribers, readers, publishers or writers

An attempt to call a disallowed function from within a callback will result in DDS_RETCODE_ILLEGAL_OPERATION.

The following are *not* allowed:

- Within any listener callback, creating any entity
- Within any listener callback, deleting any entity
- Within any listener callback, enabling any entity
- Within any listener callback, setting the QoS of any entities
- Within a DDS_DataReader or DDS_Subscriber listener callback, invoking any operation on any other DDS_Subscriber or on any DDS_DataReader belonging to another DDS_Subscriber.
- Within a DDS_DataReader or DDS_Subscriber listener callback, invoking any operation on any DDS_Publisher (or on any DDS_DataWriter belonging to such a DDS_Publisher).
- Within a DDS_DataWriter of DDS_Publisher listener callback, invoking any operation on another Publisher or on a DDS_DataWriter belonging to another DDS_Publisher.
- Within a DDS_DataWriter of DDS_Publisher listener callback, invoking any operation on any DDS_Subscriber or DDS_DataReader.

An attempt to call a disallowed function from within a callback will result in DDS_RETCODE_ILLEGAL_OPERATION.

See also

Status Kinds

6.29.4 Field Documentation

6.29.4.1 void* DDS_Listener::listener_data

A place for listener implementors to keep a pointer to data that may be needed by their listener.

6.30 DDS_LivelinessChangedStatus Struct Reference

<<*cert*>> DDS_LIVELINESS_CHANGED_STATUS

Data Fields

- DDS_Long alive_count

  The total count of currently alive DDS_DataWriter entities that write the DDS_Topic the DDS_DataReader reads.
- DDS_Long not_alive_count

  The total count of currently not_alive DDS_DataWriter entities that write the DDS_Topic the DDS_DataReader reads.
- DDS_Long alive_count_change

  The change in the alive_count since the last time the listener was called or the status was read.
- DDS_Long not_alive_count_change

  The change in the not_alive_count since the last time the listener was called or the status was read.
- DDS_InstanceHandle_t last_publication_handle

  An instance handle to the last remote writer to change its liveliness.

6.30.1 Detailed Description

<<*cert*>> DDS_LIVELINESS_CHANGED_STATUS

6.30.2 Field Documentation

6.30.2.1 DDS_Long DDS_LivelinessChangedStatus::alive_count

The total count of currently alive DDS_DataWriter entities that write the DDS_Topic the DDS_DataReader reads.

6.30.2.2 DDS_Long DDS_LivelinessChangedStatus::not_alive_count

The total count of currently not_alive DDS_DataWriter entities that write the DDS_Topic the DDS_DataReader reads.

6.30.2.3 DDS_Long DDS_LivelinessChangedStatus::alive_count_change

The change in the alive_count since the last time the listener was called or the status was read.

6.30.2.4 DDS_Long DDS_LivelinessChangedStatus::not_alive_count_change

The change in the not_alive_count since the last time the listener was called or the status was read.

6.30.2.5 DDS_InstanceHandle_t DDS_LivelinessChangedStatus::last_publication_handle

An instance handle to the last remote writer to change its liveliness.

6.31 DDS_LivelinessLostStatus Struct Reference

<<*cert*>> DDS_LIVELINESS_LOST_STATUS

Data Fields

- DDS_Long total_count

  Total cumulative number of times that a previously-alive DDS_DataWriter became not alive due to a failure to to actively signal its liveliness within the offered liveliness period.
- DDS_Long total_count_change

  The incremental changees in total_count since the last time the listener was called or the status was read.

6.31.1 Detailed Description

<<*cert*>> DDS_LIVELINESS_LOST_STATUS

Entity:

DDS_DataWriter

Listener:

DDS_DataWriterListener

The liveliness that the DDS_DataWriter has committed through its DDS_LivelinessQosPolicy was not respected; thus DDS_DataReader entities will consider the DDS_DataWriter as no longer "alive/active".

6.31.2 Field Documentation

6.31.2.1 DDS_Long DDS_LivelinessLostStatus::total_count

Total cumulative number of times that a previously-alive DDS_DataWriter became not alive due to a failure to to actively signal its liveliness within the offered liveliness period.

This count does not change when an already not alive DDS_DataWriter simply remains not alive for another liveliness period.

6.31.2.2 DDS_Long DDS_LivelinessLostStatus::total_count_change

The incremental changees in total_count since the last time the listener was called or the status was read.

6.32 DDS_LivelinessQosPolicy Struct Reference

<<*cert*>> Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive.

Data Fields

- DDS_LivelinessQosPolicyKind kind

  The kind of liveliness desired.
- struct DDS_Duration_t lease_duration

  The duration within which a DDS_Entity must be asserted, or else it is assumed to be not alive.

6.32.1 Detailed Description

<<*cert*>> Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive.

Liveliness must be asserted at least once every `lease_duration` otherwise RTI Connext Micro will assume the corresponding DDS_Entity or is no longer alive.

The liveliness status of a DDS_Entity is used to maintain instance ownership in combination with the setting of the OWNERSHIP policy. The application is also informed via DDS_Listener when an DDS_Entity is no longer alive.

A DDS_DataWriter commits to signalling its liveliness using the stated means at intervals not to exceed the `lease_duration`. That is, a failure by the DDS_DataWriter to assert liveliness within its offered `lease_duration` will result in a DDS_LIVELINESS_LOST_STATUS event.

A DDS_DataReader will match a DDS_DataWriter if the DDS_DataReader's requested `lease_duration` is equal to or larger than the `lease_duration` offered by the DDS_DataWriter and the offered `kind` is equal to or better than the requested `kind`. In addition, a DDS_DataReader uses its own `lease_duration` as the check period to check if a DDS_DataWriter was active or not. This means that a DDS_DataWriter may be inactive for up to, but not including, twice the DDS_DataReader's `lease_duration` before being detected as inactive by the DDS_DataReader. It does not matter when the DDS_DataWriter was active, or how many times the DDS_DataWriter was active within the last DDS_DataReader's `lease_duration` to be considered inactive.

For example: If a DDS_DataReader must detect that a DDS_DataWriter was inactive within at most 100ms, both the DDS_DataWriter and DDS_DataReader must specify a `lease_duration` of 50ms.

Listeners are used to notify a DDS_DataReader of loss of liveliness and DDS_DataWriter of violations to the liveliness contract. The on_liveliness_lost() callback is only called *once*, after the first time the `lease_duration` is exceeded (when the DDS_DataWriter first loses liveliness).

This QoS policy can be used during system integration to ensure that applications have been coded to meet design specifications. It can also be used during run time to detect when systems are performing outside of design specifications. Receiving applications can take appropriate actions in response to disconnected DataWriters.

Entity:

DDS_Topic, DDS_DataReader, DDS_DataWriter

Status:

DDS_LIVELINESS_LOST_STATUS, DDS_LivelinessLostStatus;
DDS_LIVELINESS_CHANGED_STATUS, DDS_LivelinessChangedStatus;
DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS, DDS_OFFERED_INCOMPATIBLE_QOS_STATUS

Properties:

RxO = YES
Changeable = NO

6.32.2 Usage

This policy controls the mechanism and parameters used by RTI Connext Micro to ensure that particular entities on the network are still alive. The liveliness can also affect the ownership of a particular instance, as determined by the OWNERSHIP policy.

This policy has several settings to support both data types that are updated periodically as well as those that are changed sporadically. It also allows customisation for different application requirements in terms of the kinds of failures that will be detected by the liveliness mechanism.

The DDS_AUTOMATIC_LIVELINESS_QOS liveliness setting is most appropriate for applications that only need to detect failures at the process-level, but not application-logic failures within a process. RTI Connext Micro takes responsibility for renewing the leases at the required rates and thus, as long as the local process where a DDS_DomainParticipant is running and the link connecting it to remote participants remains connected, the entities within the DDS_DomainParticipant will be considered alive. This requires the lowest overhead.

The manual settings (DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS, DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS) require the application on the publishing side to periodically assert the liveliness before the lease expires to indicate the corresponding DDS_Entity is still alive. The action can be explicit by calling the DDS_DataWriter_assert_liveliness operation or implicit by writing some data.

The two possible manual settings control the granularity at which the application must assert liveliness.

- The setting DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS requires only that one DDS_Entity within a participant is asserted to be alive to deduce all other DDS_Entity objects within the same DDS_DomainParticipant are also alive.

- The setting DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS requires that at least one instance within the DDS_DataWriter is asserted.

Changes in LIVELINESS must be detected by the Service with a time-granularity greater or equal to the `lease_duration`. This ensures that the value of the DDS_LivelinessChangedStatus is updated at least once during each `lease_duration` and the related Listeners are notified within a `lease_duration` from the time the LIVELINESS changed.

6.32.3 Compatibility

The value offered is considered compatible with the value requested if and only if the following conditions are met:

- the inequality *offered kind* >= *requested kind* evaluates to 'TRUE'. For the purposes of this inequality, the values of DDS_LivelinessQosPolicyKind kind are considered ordered such that: DDS_AUTOMATIC_LIVELINESS_QOS < DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS < DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS.

- the inequality *offered* `lease_duration` <= *requested* `lease_duration` evaluates to DDS_BOOLEAN_TRUE.

6.32.4 Field Documentation

6.32.4.1 DDS_LivelinessQosPolicyKind DDS_LivelinessQosPolicy::kind

The kind of liveliness desired.

[default] DDS_AUTOMATIC_LIVELINESS_QOS

6.32.4.2 struct DDS_Duration_t DDS_LivelinessQosPolicy::lease_duration

The duration within which a DDS_Entity must be asserted, or else it is assumed to be not alive.

[default] DDS_DURATION_INFINITE

[range] [0,1 year] or DDS_DURATION_INFINITE

6.33 DDS_Locator_t Struct Reference

<<eXtension>> *<<cert>>* Type used to represent the addressing information needed to send a message to an RTPS Endpoint using one of the supported transports.

Data Fields

- DDS_Long kind

  The kind of locator.

- DDS_UnsignedLong port

  the port number

- DDS_Octet address [RTPS_LOCATOR_ADDRESS_LENGTH_MAX]

  An octet field (with maximum length of RTPS_LOCATOR_ADDRESS_LENGTH_MAX to hold the IP address.

6.33.1 Detailed Description

<<*eXtension*>> <<*cert*>> Type used to represent the addressing information needed to send a message to an RTPS Endpoint using one of the supported transports.

6.33.2 Field Documentation

6.33.2.1 DDS_Long DDS_Locator_t::kind

The kind of locator.

If the Locator_t kind is DDS_LOCATOR_KIND_UDPv4, the address contains an IPv4 address. In this case, the leading 12 octets of the DDS_Locator_t::address must be zero. The last 4 octets of DDS_Locator_t::address are used to store the IPv4 address in big-endian order.

If the Locator_t kind is not DDS_LOCATOR_KIND_UDPv4, the address it not interpreted and is treated as an array of 16 octets.

6.33.2.2 DDS_UnsignedLong DDS_Locator_t::port

the port number

6.33.2.3 DDS_Octet DDS_Locator_t::address[RTPS_LOCATOR_ADDRESS_LENGTH_MAX]

An octet field (with maximum length of RTPS_LOCATOR_ADDRESS_LENGTH_MAX to hold the IP address.

6.34 DDS_LocatorSeq Struct Reference

Declares IDL `sequence` < DDS_Locator_t >

6.34.1 Detailed Description

Declares IDL `sequence` < DDS_Locator_t >

See also

DDS_Locator_t

6.35 DDS_OfferedDeadlineMissedStatus Struct Reference

<<*cert*>> DDS_OFFERED_DEADLINE_MISSED_STATUS

Data Fields

- DDS_Long total_count

  Total cumulative count of the number of times the DDS_DataWriter failed to write within its offered deadline.
- DDS_Long total_count_change

  The incremental changes in total_count since the last time the listener was called or the status was read.
- DDS_InstanceHandle_t last_instance_handle

  Handle to the last instance in the DDS_DataWriter for which an offered deadline was missed.

6.35.1 Detailed Description

<<*cert*>> DDS_OFFERED_DEADLINE_MISSED_STATUS

Entity:

DDS_DataWriter

Listener:

DDS_DataWriterListener

The deadline that the DDS_DataWriter has committed through its DDS_DeadlineQosPolicy was not respected for a specific instance.

6.35.2 Field Documentation

6.35.2.1 DDS_Long DDS_OfferedDeadlineMissedStatus::total_count

Total cumulative count of the number of times the DDS_DataWriter failed to write within its offered deadline. Missed deadlines accumulate; that is, each deadline period the `total_count` will be incremented by one.

```
total count
```

6.35.2.2 DDS_Long DDS_OfferedDeadlineMissedStatus::total_count_change

The incremental changes in total_count since the last time the listener was called or the status was read.

6.35.2.3 DDS_InstanceHandle_t DDS_OfferedDeadlineMissedStatus::last_instance_handle

Handle to the last instance in the DDS_DataWriter for which an offered deadline was missed.

6.36 DDS_OfferedIncompatibleQosStatus Struct Reference

<<*cert*>> DDS_OFFERED_INCOMPATIBLE_QOS_STATUS

Data Fields

- DDS_Long total_count

  Total cumulative number of times the concerned DDS_DataWriter discovered a DDS_DataReader for the same DDS_Topic, common partition with a requested QoS that is incompatible with that offered by the DDS_DataWriter.
- DDS_Long total_count_change

  The incremental changes in total_count since the last time the listener was called or the status was read.
- DDS_QosPolicyId_t last_policy_id

  The DDS_QosPolicyId_t of one of the policies that was found to be incompatible the last time an incompatibility was detected.
- struct DDS_QosPolicyCountSeq policies

  A list containing for each policy the total number of times that the concerned DDS_DataWriter discovered a DDS_DataReader for the same DDS_Topic and common partition with a requested QoS that is incompatible with that offered by the DDS_DataWriter.

6.36.1 Detailed Description

<<*cert*>> DDS_OFFERED_INCOMPATIBLE_QOS_STATUS

Entity:

DDS_DataWriter

Listener:

DDS_DataWriterListener

The qos policy value was incompatible with what was requested.

6.36.2 Field Documentation

6.36.2.1 DDS_Long DDS_OfferedIncompatibleQosStatus::total_count

Total cumulative number of times the concerned DDS_DataWriter discovered a DDS_DataReader for the same DDS_Topic, common partition with a requested QoS that is incompatible with that offered by the DDS_DataWriter.

6.36.2.2 DDS_Long DDS_OfferedIncompatibleQosStatus::total_count_change

The incremental changes in total_count since the last time the listener was called or the status was read.

6.36.2.3 DDS_QosPolicyId_t DDS_OfferedIncompatibleQosStatus::last_policy_id

The DDS_QosPolicyId_t of one of the policies that was found to be incompatible the last time an incompatibility was detected.

6.36.2.4 struct DDS_QosPolicyCountSeq DDS_OfferedIncompatibleQosStatus::policies

A list containing for each policy the total number of times that the concerned DDS_DataWriter discovered a DDS_DataReader for the same DDS_Topic and common partition with a requested QoS that is incompatible with that offered by the DDS_DataWriter.

6.37 DDS_OwnershipQosPolicy Struct Reference

<<cert>> Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated.

Data Fields

- DDS_OwnershipQosPolicyKind kind
  The kind of ownership.

6.37.1 Detailed Description

<<cert>> Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated.

Entity:

DDS_Topic, DDS_DataReader, DDS_DataWriter

Status:

DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS

Properties:

RxO = YES
Changeable = NO

See also

OWNERSHIP_STRENGTH

6.37.2 Usage

This policy controls whether RTI Connext Micro allows multiple DDS_DataWriter objects to update the same instance (identified by DDS_Topic + key) of a data type.

There are two kinds of owner selected by the setting of the `kind`: SHARED and EXCLUSIVE.

6.37.2.1 SHARED ownership

DDS_SHARED_OWNERSHIP_QOS indicates that RTI Connext Micro does not enforce unique ownership for each instance. In this case, multiple writers can update the same data type instance. The subscriber to the DDS_Topic will be able to access modifications from all DDS_DataWriter objects, subject to the settings of other QoS that may filter particular samples (e.g. the HISTORY policy). In any case there is no "filtering" of modifications made based on the identity of the DDS_DataWriter that causes the modification.

6.37.2.2 EXCLUSIVE ownership

DDS_EXCLUSIVE_OWNERSHIP_QOS indicates that each instance of a data type can only be modified by one DDS_DataWriter. In other words, at any point in time a single DDS_DataWriter owns each instance and is the only one whose modifications will be visible to the DDS_DataReader objects. The owner is determined by selecting the DDS_DataWriter with the highest value of the DDS_OwnershipStrengthQosPolicy::value that is currently alive as defined by the LIVELINESS policy and has not violated its DEADLINE contract with regards to the data-instance.

Ownership can therefore change as a result of:

- a DDS_DataWriter in the system with a higher value of the strength that modifies the instance,
- a change in the strength value of the DDS_DataWriter that owns the instance, and
- a change in the liveliness of the DDS_DataWriter that owns the instance.
- a deadline with regards to the instance that is missed by the DDS_DataWriter that owns the instance.

The behavior of the system is as if the determination was made independently by each DDS_DataReader. Each DDS_DataReader may detect the change of ownership at a different time. It is not a requirement that at a particular point in time all the DDS_DataReader objects for that DDS_Topic have a consistent picture of who owns each instance.

It is also not a requirement that the DDS_DataWriter objects are aware of whether they own a particular instance. There is no error or notification given to a DDS_DataWriter that modifies an instance it does not currently own.

The requirements are chosen to (a) preserve the decoupling of publishers and subscriber, and (b) allow the policy to be implemented efficiently.

It is possible that multiple DDS_DataWriter objects with the same strength modify the same instance. If this occurs RTI Connext Micro will pick one of the DDS_DataWriter objects as the owner. It is not specified how the owner is selected. However, it is required that the policy used to select the owner is such that all DDS_DataReader objects will make the same choice of the particular DDS_DataWriter that is the owner. It is required that the owner remains the same until there is a change in strength, liveliness, the owner misses a deadline on the instance, or a new DDS_DataWriter with higher same strength, or a new DDS_DataWriter with same strength that should be deemed the owner according to the policy of the Service, modifies the instance.

Exclusive ownership is on an instance-by-instance basis. That is, a subscriber can receive values written by a lower strength DDS_DataWriter as long as they affect instances whose values have not been set by the higher-strength DDS_DataWriter.

6.37.3 Compatibility

The value of the DDS_OwnershipQosPolicyKind offered must exactly match the one requested or else they are considered incompatible.

6.37.3.1 Ownership Resolution on Redundant Systems

It is expected that users may use DDS to set up redundant systems where multiple DDS_DataWriter entities are "capable" of writing the same instance. In this situation, the DDS_DataWriter entities are configured such that:

- Either both are writing the instance "constantly"
- Or else they use some mechanism to classify each other as "primary" and "secondary", such that the primary is the only one writing, and the secondary monitors the primary and only writes when it detects that the primary "writer" is no longer writing.

Both cases above use the DDS_EXCLUSIVE_OWNERSHIP_QOS and arbitrate themselves by means of the DDS_OwnershipStrengthQosPolicy. Regardless of the scheme, the desired behavior from the DDS_DataReader point of view is that DDS_DataReader normally receives data from the primary unless the "primary" writer stops writing, in which case the DDS_DataReader starts to receive data from the secondary DDS_DataWriter.

This approach requires some mechanism to detect that a DDS_DataWriter (the primary) is no longer "writing" the data as it should. There are several reasons why this may happen and all must be detected (but not necessarily distinguished):

- [crash] The writing process is no longer running (e.g. the whole application has crashed)
- [connectivity loss] Connectivity to the writing application has been lost (e.g. network disconnection)
- [application fault] The application logic that was writing the data is faulty and has stopped calling FooDataWriter_write.

Arbitrating from a DDS_DataWriter to one of a higher strength is simple and the decision can be taken autonomously by the DDS_DataReader. Switching ownership from a higher strength DDS_DataWriter to one of a lower strength DDS_DataWriter requires that the DDS_DataReader can make a determination that the stronger DDS_DataWriter is "no longer writing the instance".

6.37.3.1.1 Case where the data is periodically updated

This determination is reasonably simple when the data is being written periodically at some rate. The DDS_DataWriter simply states its offered DDS_DeadlineQosPolicy (maximum interval between updates) and the DDS_DataReader automatically monitors that the DDS_DataWriter indeed updates the instance at least once per DDS_DeadlineQosPolicy::period. If the deadline is missed, the DDS_DataReader considers the DDS_DataWriter "not alive" and automatically gives ownership to the next highest-strength DDS_DataWriter that *is* alive.

6.37.3.1.2 Case where data is not periodically updated

The case where the DDS_DataWriter is not writing data periodically is also a very important use-case. Since the instance is not being updated at any fixed period, the "deadline" mechanism cannot be used to determine ownership. The liveliness solves this situation. Ownership is maintained while the DDS_DataWriter is "alive" and for the DDS_DataWriter to be alive it must fulfill its DDS_LivelinessQosPolicy contract. The different means to renew liveliness (automatic, manual) combined by the implied renewal each time data is written handle the three conditions above [crash], [connectivity loss], and [application fault]. Note that to handle [application fault], LIVELINESS must be DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS. The DDS_DataWriter can retain ownership by periodically writing data or else calling assert_liveliness if it has no data to write. Alternatively if only protection against [crash] or [connectivity loss] is desired, it is sufficient that some task on the DDS_DataWriter process periodically writes data . However, this scenario requires that the DDS_DataReader knows what instances are being "written" by the DDS_DataWriter. That is the only way that the DDS_DataReader deduces the ownership of specific instances from the fact that the DDS_DataWriter is still "alive".

6.37.3.2 Detection of Loss in Topological Connectivity

There are applications that are designed in such a way that their correct operation requires some minimal topological connectivity, that is, the writer needs to have a minimum number of readers or alternatively the reader must have a minimum number of writers.

A common scenario is that the application does not start doing its logic until it knows that some specific writers have the minimum configured readers (e.g the alarm monitor is up).

A *more* common scenario is that the application logic will wait until some writers appear that can provide some needed source of information (e.g. the raw sensor data that must be processed).

Furthermore, once the application is running it is a requirement that this minimal connectivity (from the source of the data) is monitored and the application informed if it is ever lost. For the case where data is being written periodically, the DDS_DeadlineQosPolicy and the on_deadline_missed listener provides the notification. The case where data is not periodically updated requires the use of the DDS_LivelinessQosPolicy to detect whether the "connectivity" has been lost, and the notification is provided by means of DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.

In terms of the required mechanisms, the scenario is very similar to the case of maintaining ownership. In both cases, the reader needs to know whether a writer is still "managing the current value of an instance" even though it is not continually writing it and this knowledge requires the writer to keep its liveliness.

For a DDS_Topic with DDS_EXCLUSIVE_OWNERSHIP_QOS, if the current owner of an instance *disposes* it, the readers accessing the instance will see the instance_state as being "DISPOSED" and not see the values being written by the weaker writer (even after the stronger one has disposed the instance). This is because the DDS_DataWriter that owns the instance is saying that the instance no longer exists (e.g. the master of the database is saying that a record has been deleted) and thus the readers should see it as such.

6.37.4 Field Documentation

6.37.4.1 DDS_OwnershipQosPolicyKind DDS_OwnershipQosPolicy::kind

The kind of ownership.

[default] DDS_SHARED_OWNERSHIP_QOS

6.38 DDS_OwnershipStrengthQosPolicy Struct Reference

<<*cert*>> Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key).

Data Fields

- DDS_Long value

  The strength value used to arbitrate among multiple writers.

6.38.1 Detailed Description

<<*cert*>> Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key).

This policy only applies if the OWNERSHIP policy is of kind DDS_EXCLUSIVE_OWNERSHIP_QOS.

Entity:

DDS_DataWriter

Properties:

RxO = N/A
Changeable = NO

The value of the OWNERSHIP_STRENGTH is used to determine the ownership of a data-instance (identified by the key). The arbitration is performed by the DDS_DataReader.

See also

EXCLUSIVE ownership

6.38.2 Field Documentation

6.38.2.1 DDS_Long DDS_OwnershipStrengthQosPolicy::value

The strength value used to arbitrate among multiple writers.

[default] 0

[range] [0, 1 million]

6.39 DDS_ParticipantBuiltinTopicData Struct Reference

<<cert>> Object representing a remote DomainParticipant.

Data Fields

- struct DDS_BuiltinTopicKey_t key

  DCPS key to distinguish entries.
- struct DDS_EntityNameQosPolicy participant_name

  The participant name.
- DDS_UnsignedLong dds_builtin_endpoints

  Bitmap of builtin endpoints supported by the participant.
- DDS_ProtocolVersion_t rtps_protocol_version

  Version number of the RTPS wire protocol used.
- struct DDS_VendorId_t rtps_vendor_id

  ID of vendor implementing the RTPS wire protocol.
- struct DDS_LocatorSeq default_unicast_locators

  Unicast locators used when individual entities do not specify unicast locators.
- struct DDS_LocatorSeq default_multicast_locators

  Multicast locators used when individual entities do not specify multicast locators.
- struct DDS_LocatorSeq metatraffic_unicast_locators

  Unicast locators used for discovery metatraffic.
- struct DDS_LocatorSeq metatraffic_multicast_locators

  Multicast locators used for discovery metatraffic.
- struct DDS_Duration_t liveliness_lease_duration

  Liveliness lease duration of participant.
- struct DDS_ProductVersion_t product_version

  This is a vendor specific parameter. It gives the current version for RTI Connext Micro.
- struct DDS_ChecksumProperty checksum

  Checksum properties for the particpant.

6.39.1 Detailed Description

<<cert>> Object representing a remote DomainParticipant.

Data associated with the built-in topic DDS_PARTICIPANT_TOPIC_NAME. It contains QoS policies and additional information that apply to the remote DDS_DomainParticipant.

6.39.2 Field Documentation

6.39.2.1 struct DDS_BuiltinTopicKey_t DDS_ParticipantBuiltinTopicData::key

DCPS key to distinguish entries.

6.39.2.2 struct DDS_EntityNameQosPolicy DDS_ParticipantBuiltinTopicData::participant_name

The participant name.

This is the name of the discovered participant.

6.39.2.3 DDS_UnsignedLong DDS_ParticipantBuiltinTopicData::dds_builtin_endpoints

Bitmap of builtin endpoints supported by the participant.

Each bit indicates a builtin endpoint that may be available on the participant for use in discovery.

6.39.2.4 DDS_ProtocolVersion_t DDS_ParticipantBuiltinTopicData::rtps_protocol_version

Version number of the RTPS wire protocol used.

6.39.2.5 struct DDS_VendorId_t DDS_ParticipantBuiltinTopicData::rtps_vendor_id

ID of vendor implementing the RTPS wire protocol.

6.39.2.6 struct DDS_LocatorSeq DDS_ParticipantBuiltinTopicData::default_unicast_locators

Unicast locators used when individual entities do not specify unicast locators.

6.39.2.7 struct DDS_LocatorSeq DDS_ParticipantBuiltinTopicData::default_multicast_locators

Multicast locators used when individual entities do not specify multicast locators.

6.39.2.8 struct DDS_LocatorSeq DDS_ParticipantBuiltinTopicData::metatraffic_unicast_locators

Unicast locators used for discovery metatraffic.

6.39.2.9 struct DDS_LocatorSeq DDS_ParticipantBuiltinTopicData::metatraffic_multicast_locators

Multicast locators used for discovery metatraffic.

6.39.2.10 struct DDS_Duration_t DDS_ParticipantBuiltinTopicData::liveliness_lease_duration

Liveliness lease duration of participant.

6.39.2.11 struct DDS_ProductVersion_t DDS_ParticipantBuiltinTopicData::product_version

This is a vendor specific parameter. It gives the current version for RTI Connext Micro.

6.39.2.12 struct DDS_ChecksumProperty DDS_ParticipantBuiltinTopicData::checksum

Checksum properties for the particpant.

6.40 DDS_PresentationQosPolicy Struct Reference

Specifies how the samples representing changes to data instances are presented to a subscribing application.

Data Fields

- DDS_PresentationQosPolicyAccessScopeKind access_scope

Determines the largest scope spanning the entities for which the order and coherency of changes can be preserved. This value cannot be changed in RTI Connext Micro.

- DDS_Boolean coherent_access

  Specifies support for coherent access. Controls whether coherent access is supported within the scope `access_scope`*. This is not supported by RTI Connext Micro.*

- DDS_Boolean ordered_access

  Specifies support for ordered access to the samples received at the subscription end. Controls whether ordered access is supported within the scope `access_scope`*. This value cannot be changed in RTI Connext Micro.*

6.40.1 Detailed Description

Specifies how the samples representing changes to data instances are presented to a subscribing application.

NOTE: RTI Connext Micro *does not* support setting this QoS policy by the application.

A RTI Connext Micro DDS_Publisher always offers:

- access_scope = DDS_TOPIC_PRESENTATION_QOS
- ordered_access = DDS_BOOLEAN_TRUE
- coherent_access = DDS_BOOLEAN_FALSE

These settings enable a DDS_Publisher to match with a DDS_Subscriber that requests either DDS_INSTANCE_PRESENTATION_QOS or DDS_TOPIC_PRESENTATION_QOS, increasing interoperability.

A RTI Connext Micro DDS_Subscriber always requests:

- access_scope = DDS_INSTANCE_PRESENTATION_QOS
- ordered_access = DDS_BOOLEAN_FALSE
- coherent_access = DDS_BOOLEAN_FALSE

The following is a general description of the DDS_PresentationQosPolicy.

This QoS policy controls the extent to which changes to data instances can be made dependent on each other and also the kind of dependencies that can be propagated and maintained by RTI Connext Micro. Specifically, this policy affects the application's ability to:

- specify and receive coherent changes to instances
- specify the relative order in which changes are presented

Entity:

DDS_Publisher, DDS_Subscriber

Status:

DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS

A DDS_DataReader will usually receive data in the order that it was sent by a DDS_DataWriter, and the data is presented to the DDS_DataReader as soon as the application receives the next expected value. However, sometimes you may want a set of data for the same DDS_DataWriter to be presented to the DDS_DataReader only after *all* of the elements of the set have been received. Or you may want the data to be presented in a different order than that in which it was received. Specifically for keyed data, you may want the middleware to present the data in keyed – or *instance* – order, such that samples pertaining to the same instance are presented together.

The Presentation QoS policy allows you to specify different *scopes* of presentation: within a DDS_DataWriter, across instances of a single DDS_DataWriter, and even across multiple DataWriters used by different writers of a publisher. It also controls whether or not a set of changes within the scope is delivered at the same time or can be delivered as soon as each element is received.

- `coherent_access` controls whether RTI Connext Micro will preserve the groupings of changes made by a publishing application by means of the operations begin_coherent_changes() and end_coherent_changes(). These operations are not supported.

- `ordered_access` controls whether RTI Connext Micro will preserve the order of changes.

- `access_scope` controls the granularity of the other settings. See below:

If `coherent_access` is set, then the `access_scope` controls the maximum extent of coherent changes. The behavior is as follows:

- If `access_scope` is set to DDS_INSTANCE_PRESENTATION_QOS (the default), the use of begin_coherent_changes() and end_coherent_changes() has no effect on how the subscriber can access the data, because with the scope limited to each instance, changes to separate instances are considered independent and thus cannot be grouped into a coherent set.

- If `access_scope` is set to DDS_TOPIC_PRESENTATION_QOS, then coherent changes (indicated by their enclosure within calls to begin_coherent_changes() and end_coherent_changes() will be made available as such to each remote DDS_DataReader independently. That is, changes made to instances within each individual DDS_DataWriter will be available as coherent with respect to other changes to instances in that same DDS_DataWriter, but will not be grouped with changes made to instances belonging to a different DDS_DataWriter.

- If access_scope is set to DDS_GROUP_PRESENTATION_QOS, then coherent changes made to instances through a DDS_DataWriter attached to a common DDS_Publisher are made available as a unit to remote subscribers.

If `ordered_access` is set, then the `access_scope` controls the maximum extent for which order will be preserved by RTI Connext Micro.

- If `access_scope` is set to DDS_INSTANCE_PRESENTATION_QOS (the lowest level), then changes to each instance are considered unordered relative to changes to any other instance. That means that changes (creations, deletions, modifications) made to two instances are not necessarily seen in the order they occur. This is the case even if it is the same application thread making the changes using the same DDS_DataWriter.

- If `access_scope` is set to DDS_TOPIC_PRESENTATION_QOS, changes (creations, deletions, modifications) made by a single DDS_DataWriter are made available to subscribers in the same order they occur. Changes made to instances though different DDS_DataWriter entities are not necessarily seen in the order they occur. This is the case, even if the changes are made by a single application thread using DDS_DataWriter objects attached to the same DDS_Publisher.

- Finally, if access_scope is set to DDS_GROUP_PRESENTATION_QOS, changes made to instances via DDS_DataWriter entities attached to the same DDS_Publisher object are made available to subscribers on the same order they occur.

Note that this QoS policy controls the scope at which related changes are made available to the subscriber. This means the subscriber **can** access the changes in a coherent manner and in the proper order; however, it does not necessarily imply that the DDS_Subscriber **will** indeed access the changes in the correct order. For that to occur, the application at the subscriber end must use the proper logic in reading the DDS_DataReader objects.

6.40.2 Compatibility

The value offered is considered compatible with the value requested if and only if the following conditions are met:

- the inequality *offered* `access_scope` >= *requested* `access_scope` evaluates to 'TRUE'. For the purposes of this inequality, the values of `access_scope` are considered ordered such that DDS_INSTANCE_PRESENTATION_QOS < DDS_TOPIC_PRESENTATION_QOS < DDS_GROUP_PRESENTATION_QOS.
- requested `coherent_access` is DDS_BOOLEAN_FALSE, or else both offered and requested `coherent_access` are DDS_BOOLEAN_TRUE.
- requested `ordered_access` is DDS_BOOLEAN_FALSE, or else both offered and requested `ordered_access` are DDS_BOOLEAN_TRUE.

6.40.3 Field Documentation

6.40.3.1 DDS_PresentationQosPolicyAccessScopeKind DDS_PresentationQosPolicy::access_scope

Determines the largest scope spanning the entities for which the order and coherency of changes can be preserved. This value cannot be changed in RTI Connext Micro.

[default] DDS_Publisher DDS_TOPIC_PRESENTATION_QOS

[default] DDS_Subscriber DDS_INSTANCE_PRESENTATION_QOS

6.40.3.2 DDS_Boolean DDS_PresentationQosPolicy::coherent_access

Specifies support for *coherent* access. Controls whether coherent access is supported within the scope `access_scope`. This is not supported by RTI Connext Micro.

That is, the ability to group a set of changes as a unit on the publishing end such that they are received as a unit at the subscribing end.

Note: To use this feature, the DataWriter must be configured for RELIABLE communication (see DDS_RELIABLE_RELIABILITY_QOS).

[default] DDS_BOOLEAN_FALSE

6.40.3.3 DDS_Boolean DDS_PresentationQosPolicy::ordered_access

Specifies support for *ordered* access to the samples received at the subscription end. Controls whether ordered access is supported within the scope `access_scope`. This value cannot be changed in RTI Connext Micro.

That is, the ability of the subscriber to see changes in the same order as they occurred on the publishing end.

[default] DDS_Publisher DDS_BOOLEAN_TRUE

[default] DDS_Subscriber DDS_BOOLEAN_FALSE

6.41 DDS_ProductVersion Struct Reference

<<eXtension>> *<<cert>>* Type used to represent the current version of RTI Connext Micro.

Data Fields

- DDS_Char major

  Major product version.
- DDS_Char minor

  Minor product version.
- DDS_Char release

  Release letter for product version.
- DDS_Char revision

  Revision number of product.

6.41.1 Detailed Description

<<eXtension>> *<<cert>>* Type used to represent the current version of RTI Connext Micro.

6.41.2 Field Documentation

6.41.2.1 DDS_Char DDS_ProductVersion::major

Major product version.

6.41.2.2 DDS_Char DDS_ProductVersion::minor

Minor product version.

6.41.2.3 DDS_Char DDS_ProductVersion::release

Release letter for product version.

6.41.2.4 DDS_Char DDS_ProductVersion::revision

Revision number of product.

6.42 DDS_ProductVersion_t Struct Reference

<<eXtension>> *<<cert>>* Type used to represent the current version of RTI Connext Micro.

Data Fields

- DDS_Char major
  Major product version.
- DDS_Char minor
  Minor product version.
- DDS_Char release
  Release letter for product version.
- DDS_Char revision
  Revision number of product.

6.42.1 Detailed Description

<<eXtension>> *<<cert>>* Type used to represent the current version of RTI Connext Micro.

6.42.2 Field Documentation

6.42.2.1 DDS_Char DDS_ProductVersion_t::major

Major product version.

6.42.2.2 DDS_Char DDS_ProductVersion_t::minor

Minor product version.

6.42.2.3 DDS_Char DDS_ProductVersion_t::release

Release letter for product version.

6.42.2.4 DDS_Char DDS_ProductVersion_t::revision

Revision number of product.

6.43 DDS_ProtocolVersion Struct Reference

<<eXtension>> <<cert>> Type used to represent the version of the RTPS protocol.

Data Fields

- DDS_Octet major

  Major protocol version number.
- DDS_Octet minor

  Minor protocol version number.

6.43.1 Detailed Description

<<eXtension>> <<cert>> Type used to represent the version of the RTPS protocol.

6.43.2 Field Documentation

6.43.2.1 DDS_Octet DDS_ProtocolVersion::major

Major protocol version number.

6.43.2.2 DDS_Octet DDS_ProtocolVersion::minor

Minor protocol version number.

6.44 DDS_PublicationBuiltinTopicData Struct Reference

Object describing a remote Publication.

Data Fields

- struct DDS_BuiltinTopicKey_t key

  DCPS key to distinguish entries. This is used to configure the object ID of the remote DDS_DataWriter.
- struct DDS_BuiltinTopicKey_t participant_key

  DCPS key of the participant to which the DataWriter belongs.
- char * topic_name

  Name of the related DDS_Topic.
- char * type_name

  Name of the type attached to the DDS_Topic.
- struct DDS_DeadlineQosPolicy deadline

  Policy of the corresponding DataWriter.
- struct DDS_OwnershipQosPolicy ownership

  Policy of the corresponding DataWriter.
- struct DDS_OwnershipStrengthQosPolicy ownership_strength

  Policy of the corresponding DataWriter.
- struct DDS_ReliabilityQosPolicy reliability

Policy of the corresponding DataWriter.

- struct DDS_LivelinessQosPolicy liveliness

  Policy of the corresponding DataWriter.

- struct DDS_DurabilityQosPolicy durability

  durability policy of the corresponding DataWriter

- struct DDS_DestinationOrderQosPolicy destination_order

  Policy of the corresponding DataWriter.

- struct DDS_LocatorSeq unicast_locator

  Custom unicast locator sequence that the endpoint can specify. The default locator sequence from the DDS_DomainParticipant will be used if this is not specified.

6.44.1 Detailed Description

Object describing a remote Publication.

This object contains all of the information about a remote DDS_DataWriter and its DDS_Publisher that is necessary for matching the remote DDS_DataWriter to a local DDS_DataReader during discovery.

This object is used to pre-configure remote entities for Dynamic Participant/Static Endpoint discovery.

For example: Application A is designed to discover a remote DDS_DataWriter in application B, and that remote DDS_DataWriter has an object ID of 100, and a deadline of three seconds. In Dynamic Participant/Static Endpoint discovery, this discovery data is not sent over the network, so the user must create an instance of a DDS_PublicationBuiltinTopicData structure in application A. The user then must fill in the QoS describing this remote DDS_DataWriter, including the writer's object ID and three-second deadline. Then the user must assert this DDS_PublicationBuiltinTopicData describing the remote entity with the discovery plugin. The pre-configured QoS must match the actual QoS of the remote entity, or communication will not occur.

6.44.2 Field Documentation

6.44.2.1 struct DDS_BuiltinTopicKey_t DDS_PublicationBuiltinTopicData::key

DCPS key to distinguish entries. This is used to configure the object ID of the remote DDS_DataWriter.

6.44.2.2 struct DDS_BuiltinTopicKey_t DDS_PublicationBuiltinTopicData::participant_key

DCPS key of the participant to which the DataWriter belongs.

6.44.2.3 char* DDS_PublicationBuiltinTopicData::topic_name

Name of the related DDS_Topic.

The length of this string is limited to 255 characters.

The memory for this field is managed as described in String Support.

See also

String Support

6.44.2.4 char* DDS_PublicationBuiltinTopicData::type_name

Name of the type attached to the DDS_Topic.

The length of this string is limited to 255 characters.

The memory for this field is managed as described in String Support.

See also

String Support

6.44.2.5 struct DDS_DeadlineQosPolicy DDS_PublicationBuiltinTopicData::deadline

Policy of the corresponding DataWriter.

6.44.2.6 struct DDS_OwnershipQosPolicy DDS_PublicationBuiltinTopicData::ownership

Policy of the corresponding DataWriter.

6.44.2.7 struct DDS_OwnershipStrengthQosPolicy DDS_PublicationBuiltinTopicData::ownership_strength

Policy of the corresponding DataWriter.

6.44.2.8 struct DDS_ReliabilityQosPolicy DDS_PublicationBuiltinTopicData::reliability

Policy of the corresponding DataWriter.

6.44.2.9 struct DDS_LivelinessQosPolicy DDS_PublicationBuiltinTopicData::liveliness

Policy of the corresponding DataWriter.

6.44.2.10 struct DDS_DurabilityQosPolicy DDS_PublicationBuiltinTopicData::durability

durability policy of the corresponding DataWriter

6.44.2.11 struct DDS_DestinationOrderQosPolicy DDS_PublicationBuiltinTopicData::destination_order

Policy of the corresponding DataWriter.

6.44.2.12 struct DDS_LocatorSeq DDS_PublicationBuiltinTopicData::unicast_locator

Custom unicast locator sequence that the endpoint can specify. The default locator sequence from the DDS_DomainParticipant will be used if this is not specified.

6.45 DDS_PublicationMatchedStatus Struct Reference

<<*cert*>> DDS_PUBLICATION_MATCHED_STATUS

Data Fields

- DDS_Long total_count

  The total cumulative number of times the concerned DDS_DataWriter discovered a "match" with a DDS_DataReader.
- DDS_Long total_count_change

  The incremental changes in total_count since the last time the listener was called or the status was read.
- DDS_Long current_count

  The current number of readers with which the DDS_DataWriter is matched.
- DDS_Long current_count_change

  The change in current_count since the last time the listener was called or the status was read.
- DDS_InstanceHandle_t last_subscription_handle

  A handle to the last DDS_DataReader that caused the the DDS_DataWriter's status to change.

6.45.1 Detailed Description

<<*cert*>> DDS_PUBLICATION_MATCHED_STATUS

A "match" happens when the DDS_DataWriter finds a DDS_DataReader for the same DDS_Topic and common partition with a requested QoS that is compatible with that offered by the DDS_DataWriter.

This status is also changed (and the listener, if any, called) when a match is ended. A local DDS_DataWriter will become "unmatched" from a remote DDS_DataReader when that DDS_DataReader goes away for any reason.

6.45.2 Field Documentation

6.45.2.1 DDS_Long DDS_PublicationMatchedStatus::total_count

The total cumulative number of times the concerned DDS_DataWriter discovered a "match" with a DDS_DataReader.

This number increases whenever a new match is discovered. It does not change when an existing match goes away.

6.45.2.2 DDS_Long DDS_PublicationMatchedStatus::total_count_change

The incremental changes in total_count since the last time the listener was called or the status was read.

6.45.2.3 DDS_Long DDS_PublicationMatchedStatus::current_count

The current number of readers with which the DDS_DataWriter is matched.

This number increases when a new match is discovered and decreases when an existing match goes away.

6.45.2.4 DDS_Long DDS_PublicationMatchedStatus::current_count_change

The change in current_count since the last time the listener was called or the status was read.

6.45.2.5 DDS_InstanceHandle_t DDS_PublicationMatchedStatus::last_subscription_handle

A handle to the last DDS_DataReader that caused the the DDS_DataWriter's status to change.

6.46 DDS_PublisherListener Struct Reference

<<interface>> <<cert>> DDS_Listener for DDS_Publisher status.

Data Fields

- struct DDS_DataWriterListener as_datawriterlistener

  The superclass instance of this DDS_PublisherListener.

6.46.1 Detailed Description

<<interface>> <<cert>> DDS_Listener for DDS_Publisher status.

Entity:

DDS_Publisher

Status:

DDS_LIVELINESS_LOST_STATUS, DDS_LivelinessLostStatus;
DDS_OFFERED_DEADLINE_MISSED_STATUS, DDS_OfferedDeadlineMissedStatus;
DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_OfferedIncompatibleQosStatus;
DDS_PUBLICATION_MATCHED_STATUS, DDS_PublicationMatchedStatus;

See also

DDS_Listener
Status Kinds

6.46.2 Field Documentation

6.46.2.1 struct DDS_DataWriterListener DDS_PublisherListener::as_datawriterlistener

The superclass instance of this DDS_PublisherListener.

6.47 DDS_PublisherQos Struct Reference

<<*cert*>> QoS policies supported by a DDS_Publisher entity.

Data Fields

- struct DDS_EntityFactoryQosPolicy entity_factory
  Entity factory policy, ENTITY_FACTORY.

6.47.1 Detailed Description

<<*cert*>> QoS policies supported by a DDS_Publisher entity.

6.47.2 Field Documentation

6.47.2.1 struct DDS_EntityFactoryQosPolicy DDS_PublisherQos::entity_factory

Entity factory policy, ENTITY_FACTORY.

6.48 DDS_ReliabilityQosPolicy Struct Reference

<<*cert*>> Indicates the level of reliability offered/requested by RTI Connext Micro.

Data Fields

- DDS_ReliabilityQosPolicyKind kind
  Kind of reliability.
- struct DDS_Duration_t max_blocking_time
  Not supported. The maximum time must be 0.

6.48.1 Detailed Description

<<*cert*>> Indicates the level of reliability offered/requested by RTI Connext Micro.

Entity:

DDS_Topic, DDS_DataReader, DDS_DataWriter

Status:

DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS

Properties:

RxO = YES
Changeable = NO

6.48.2 Usage

This policy indicates the level of reliability requested by a DDS_DataReader or offered by a DDS_DataWriter.

These levels are ordered, DDS_BEST_EFFORT_RELIABILITY_QOS < DDS_RELIABLE_RELIABILITY_QOS. A DDS_DataWriter offering a level is implicitly offering all levels below.

If the DDS_ReliabilityQosPolicy::kind is set to DDS_RELIABLE_RELIABILITY_QOS , data-samples originating from a single DDS_DataWriter cannot be made available to the DDS_DataReader if there are previous data samples that have not been received yet due to a communication error. In other words, the service will repair the error and re-transmit data-samples as needed in order to reconstruct a correct snapshot of the DDS_DataWriter history before it is accessible by the DDS_DataReader.

If the DDS_ReliabilityQosPolicy::kind is set to DDS_BEST_EFFORT_RELIABILITY_QOS, the service will not re-transmit missing data-samples. However for data-samples originating from any one DataWriter the service will ensure they are stored in the DDS_DataReader history in the same order they originated in the DDS_DataWriter. In other words, the DDS_DataReader may miss some data samples but it will never see the value of a data-object change from a newer value to an older value.

See also

DDS_HistoryQosPolicy
DDS_ResourceLimitsQosPolicy

6.48.3 Compatibility

The value offered is considered compatible with the value requested if and only if the inequality *offered* `kind` >= *requested* `kind` evaluates to 'TRUE'. For the purposes of this inequality, the values of DDS_ReliabilityQosPolicy::kind are considered ordered such that DDS_BEST_EFFORT_RELIABILITY_QOS < DDS_RELIABLE_RELIABILITY_QOS.

6.48.4 Field Documentation

6.48.4.1 DDS_ReliabilityQosPolicyKind DDS_ReliabilityQosPolicy::kind

Kind of reliability.

[default] DDS_BEST_EFFORT_RELIABILITY_QOS

6.48.4.2 struct DDS_Duration_t DDS_ReliabilityQosPolicy::max_blocking_time

Not supported. The maximum time must be 0.

6.49 DDS_ReliableReaderActivityChangedStatus Struct Reference

<<eXtension>> *<<cert>>* Describes the activity (i.e. are acknowledgements forthcoming) of reliable readers matched to a reliable writer.

Data Fields

- DDS_Long active_count

  The current number of reliable readers currently matched with this reliable writer.
- DDS_Long inactive_count

The number of reliable readers that have been dropped by this reliable writer because they failed to send acknowledgements in a timely fashion.

- DDS_Long active_count_change

  The most recent change in the number of active remote reliable readers.

- DDS_Long inactive_count_change

  The most recent change in the number of inactive remote reliable readers.

- DDS_InstanceHandle_t last_instance_handle

  The instance handle of the last reliable remote reader to be determined inactive.

6.49.1 Detailed Description

<<*eXtension*>> <<*cert*>> Describes the activity (i.e. are acknowledgements forthcoming) of reliable readers matched to a reliable writer.

Entity:

DDS_DataWriter

Listener:

DDS_DataWriterListener

This status is the reciprocal status to the DDS_LivelinessChangedStatus on the reader. It is different than the DDS_LivelinessLostStatus on the writer in that the latter informs the writer about its own liveliness; this status informs the writer about the "liveliness" (activity) of its matched readers.

All counts in this status will remain at zero for best effort writers.

6.49.2 Field Documentation

6.49.2.1 DDS_Long DDS_ReliableReaderActivityChangedStatus::active_count

The current number of reliable readers currently matched with this reliable writer.

6.49.2.2 DDS_Long DDS_ReliableReaderActivityChangedStatus::inactive_count

The number of reliable readers that have been dropped by this reliable writer because they failed to send acknowledgements in a timely fashion.

A reader is considered to be inactive after is has been sent heartbeats DDS_RtpsReliableWriterProtocol_t::max_heartbeat_retries times, each heartbeat having been separated from the previous by the current heartbeat period.

6.49.2.3 DDS_Long DDS_ReliableReaderActivityChangedStatus::active_count_change

The most recent change in the number of active remote reliable readers.

6.49.2.4 DDS_Long DDS_ReliableReaderActivityChangedStatus::inactive_count_change

The most recent change in the number of inactive remote reliable readers.

6.49.2.5 DDS_InstanceHandle_t DDS_ReliableReaderActivityChangedStatus::last_instance_handle

The instance handle of the last reliable remote reader to be determined inactive.

6.50 DDS_ReliableSampleUnacknowledgedStatus Struct Reference

<<*cert*>> Status about unacknowledged samples.

Data Fields

- struct DDS_SequenceNumber_t sequence_number
- DDS_Long unacknowledged_count
- DDS_InstanceHandle_t instance_handle

6.50.1 Detailed Description

<<*cert*>> Status about unacknowledged samples.

Entity:

DDS_DataWriter

Listener:

DDS_DataWriterListener

6.50.2 Field Documentation

6.50.2.1 struct DDS_SequenceNumber_t DDS_ReliableSampleUnacknowledgedStatus::sequence_number

6.50.2.2 DDS_Long DDS_ReliableSampleUnacknowledgedStatus::unacknowledged_count

6.50.2.3 DDS_InstanceHandle_t DDS_ReliableSampleUnacknowledgedStatus::instance_handle

6.51 DDS_RequestedDeadlineMissedStatus Struct Reference

<<*cert*>> DDS_REQUESTED_DEADLINE_MISSED_STATUS

Data Fields

- DDS_Long total_count
  Total cumulative count of the deadlines detected for any instance read by the DDS_DataReader.
- DDS_Long total_count_change
  The incremental number of deadlines detected since the last time the listener was called or the status was read.
- DDS_InstanceHandle_t last_instance_handle
  Handle to the last instance in the DDS_DataReader for which a deadline was detected.

6.51.1 Detailed Description

<<*cert*>> DDS_REQUESTED_DEADLINE_MISSED_STATUS

6.51.2 Field Documentation

6.51.2.1 DDS_Long DDS_RequestedDeadlineMissedStatus::total_count

Total cumulative count of the deadlines detected for any instance read by the DDS_DataReader.

6.51.2.2 DDS_Long DDS_RequestedDeadlineMissedStatus::total_count_change

The incremental number of deadlines detected since the last time the listener was called or the status was read.

6.51.2.3 DDS_InstanceHandle_t DDS_RequestedDeadlineMissedStatus::last_instance_handle

Handle to the last instance in the DDS_DataReader for which a deadline was detected.

6.52 DDS_RequestedIncompatibleQosStatus Struct Reference

<<*cert*>> DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS

Data Fields

- DDS_Long total_count

  Total cumulative count of how many times the concerned DDS_DataReader discovered a DDS_DataWriter for the same DDS_Topic with an offered QoS that is incompatible with that requested by the DDS_DataReader.
- DDS_Long total_count_change

  The change in `total_count` *since the last time the listener was called or the status was read.*
- DDS_QosPolicyId_t last_policy_id

  The PolicyId_t of one of the policies that was found to be incompatible the last time an incompatibility was detected.
- struct DDS_QosPolicyCountSeq policies

  A list containing, for each policy, the total number of times that the concerned DDS_DataReader discovered a DDS_DataWriter for the same DDS_Topic with an offered QoS that is incompatible with that requested by the DDS_DataReader.

6.52.1 Detailed Description

<<*cert*>> DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS

See also

RELIABILITY
OWNERSHIP
LIVELINESS
DEADLINE

6.52.2 Field Documentation

6.52.2.1 DDS_Long DDS_RequestedIncompatibleQosStatus::total_count

Total cumulative count of how many times the concerned DDS_DataReader discovered a DDS_DataWriter for the same DDS_Topic with an offered QoS that is incompatible with that requested by the DDS_DataReader.

6.52.2.2 DDS_Long DDS_RequestedIncompatibleQosStatus::total_count_change

The change in `total_count` since the last time the listener was called or the status was read.

6.52.2.3 DDS_QosPolicyId_t DDS_RequestedIncompatibleQosStatus::last_policy_id

The PolicyId_t of one of the policies that was found to be incompatible the last time an incompatibility was detected.

6.52.2.4 struct DDS_QosPolicyCountSeq DDS_RequestedIncompatibleQosStatus::policies

A list containing, for each policy, the total number of times that the concerned DDS_DataReader discovered a DDS_DataWriter for the same DDS_Topic with an offered QoS that is incompatible with that requested by the DDS_DataReader.

6.53 DDS_ResourceLimitsQosPolicy Struct Reference

<<*cert*>> Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS.

Data Fields

- DDS_Long max_samples

  Represents the maximum samples the middleware can store for any one DDS_DataWriter or DDS_DataReader.
- DDS_Long max_instances

  Represents the maximum number of instances a DDS_DataWriter (or DDS_DataReader) can manage.
- DDS_Long max_samples_per_instance

  Represents the maximum number of samples of any one instance a DDS_DataWriter or DDS_DataReader can manage.

6.53.1 Detailed Description

<<cert>> Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS.

Entity:

DDS_Topic, DDS_DataReader, DDS_DataWriter

Status:

DDS_SAMPLE_REJECTED_STATUS, DDS_SampleRejectedStatus

Properties:

RxO = NO
Changeable = NO

6.53.2 Usage

This policy controls the resources that RTI Connext Micro can use in order to meet the requirements imposed by the application and other QoS settings.

If DDS_DataWriter objects are communicating samples faster than they are ultimately taken by the DDS_DataReader objects, the middleware will eventually hit against some of the QoS-imposed resource limits. Note that this may occur when just a single DDS_DataReader cannot keep up with its corresponding DDS_DataWriter. The behavior in this case depends on the setting for the RELIABILITY. If reliability is DDS_BEST_EFFORT_RELIABILITY_QOS then RTI Connext Micro is allowed to drop samples. If the reliability is DDS_RELIABLE_RELIABILITY_QOS, RTI Connext Micro will block the DDS_DataWriter or discard the sample at the DDS_DataReader in order not to lose existing samples.

See also

DDS_ReliabilityQosPolicy
DDS_HistoryQosPolicy

6.53.3 Consistency

All limits must be set to positive finite values. Furthermore, max_samples must be large enough to contain all samples for all instances:

```
max_samples >= max_instances *
     max_samples_per_instance
```

The setting of RESOURCE_LIMITS `max_samples_per_instance` must be consistent with the HISTORY `depth`. For these two QoS to be consistent, they must verify that *depth* <= *max_samples_per_instance*.

See also

DDS_HistoryQosPolicy

6.53.4 Field Documentation

6.53.4.1 DDS_Long DDS_ResourceLimitsQosPolicy::max_samples

Represents the maximum samples the middleware can store for any one DDS_DataWriter or DDS_DataReader.

Specifies the maximum number of data samples a DDS_DataWriter or DDS_DataReader can manage across all the instances associated with it.

For unkeyed types this value has to be equal to max_samples_per_instance.

DDS_DataReader

The DDS_ResourceLimitsQosPolicy::max_samples resource-limit limits the maximum number of samples a DDS_DataReader can allocate and cache. Samples are cached before they are made available to the application; for example, when samples are received out-of-order. Thus, this resource limit may impact throughput and latency.

What this value should be set to is determined by a number of factors:

- For best-effort, it can be set to 1 since samples are not cached. However, if data is polled or samples are loaned longer than the update frequency, this resource-limit should be increased to prevent samples from being reused before the application has a chance to access the data (along with DDS_HistoryQosPolicy::depth). li For reliable data, the value should take into account the data-rate, the heartbeat rate of the DDS_DataWriter, and the number of piggy-backed heartbeats.

DDS_DataWriter

The DDS_ResourceLimitsQosPolicy::max_samples resource-limit limits the maximum number of samples a DDS_DataWriter can allocate and maintain state for at any given time. Every time a write operation is called, a sample is allocated from the cache and filled in with user-data.

A sample allocated from the pool controlled by this resource-limit can be reused based on the following rules:

- If the DDS_DataWriter is using DDS_BEST_EFFORT_RELIABILITY_QOS effort, the sample is returned for reuse after the transport returns from the write call.
- If the DDS_DataWriter is using DDS_RELIABLE_RELIABILITY_QOS and DDS_VOLATILE_DURABILITY_QOS, the sample is returned when all known DataReaders have acknowledged the sample or the DDS_DataWriter has given up on delivering the sample (controlled by DDS_RtpsReliableWriterProtocol_t::max_heartbeat_retries), whichever happens first.
- If the DDS_DataWriter is using DDS_RELIABLE_RELIABILITY_QOS and not using DDS_VOLATILE_DURABILITY_QOS, the sample is returned to the pool for reuse only if the history depth is exceeded and the DDS_HistoryQosPolicy::kind is set to DDS_KEEP_LAST_HISTORY_QOS

For best-effort communication, this resource-limit can be set to 1.

For reliable communication, the value of this resource-limit determines the maximum number of samples a DDS_DataWriter can keep in unacknowledged state before reusing potentially unacknowledged samples (for DDS_KEEP_LAST_HISTORY_QOS). For reliable data, the value should take into account the data-rate, the DDS_RtpsReliableWriterProtocol_t::heartbeat_period, and DDS_RtpsReliableWriterProtocol_t::heartbeats_per_max_samples.

See also

DDS_HistoryQosPolicy::depth, DDS_DataWriterQos

[default] 1

[range] [1, 100 million] >= (max_samples_per_instance * max_instances),

6.53.4.2 DDS_Long DDS_ResourceLimitsQosPolicy::max_instances

Represents the maximum number of instances a DDS_DataWriter (or DDS_DataReader) can manage.

DDS_DataReader

The max_instance resource-limit limits the maximum number of instances a DDS_DataReader can manage. Instance state is one of the most expensive resources to maintain, as well as often being the highest one.

The resources allocated by an instance are normally only released when an instance is disposed *and* all known owners have unregistered the instance; please refer to FooDataWriter_unregister_instance and FooDataWriter_dispose.

The DDS_DataReaderResourceLimitsQosPolicy::instance_replacement policy is useful to maintain state for only the last N instances, and when it is not necessary to keep samples in the DDS_DataReader history cache. When the DDS_DataReaderResourceLimitsQosPolicy::instance_replacement policy is set to replace the oldest instance (DDS_REPLACE_OLDEST_INSTANCE_REPLACEMENT_QOS), information about the oldest instance is automatically removed and the newly discovered instance is received. The disadvantage is that DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE state cannot be maintained. The DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE is maintained for an instance that is accepted (either there is space or another instance is replaced), regardless of whether any samples have previously been received.

See also

DDS_DataReaderResourceLimitsInstanceReplacementKind instance_replacement

DDS_DataWriter

The DDS_ResourceLimitsQosPolicy::max_instances resource limit limits the maximum number of instances a DDS_DataWriter can manage. An instance resource can only be reused when it is unregistered by the DDS_DataWriter.

[default] 1

[range] [1, 1 million]

6.53.4.3 DDS_Long DDS_ResourceLimitsQosPolicy::max_samples_per_instance

Represents the maximum number of samples of any one instance a DDS_DataWriter or DDS_DataReader can manage.

DDS_DataReader

This resource-limit limits the maximum number of samples that can be cached for a single instance; it must be less than or equal to max_samples and greater or equal to the history depth. When DDS_ResourceLimitsQosPolicy::max_samples_per_instance is higher than depth, it makes it possible to loan samples and receive up to depth samples before returning loaned samples.

When this resource-limit or DDS_HistoryQosPolicy::depth is exceeded and the DDS_HistoryQosPolicy::kind is DDS_KEEP_LAST_HISTORY_QOS, the oldest sample is removed from the cache.

Note that it is possible to exceed the `max_samples_per_instance` limit while samples are being received and processed, but before they are added to the reader queue. This typically happens when there are holes in the sequence numbers that must be filled before the samples can be committed to the reader queue. If a sample cannot be committed due to the `max_samples_per_instance` limit, it is `not` dropped. Instead, it is recommitted when space is made available, such as the application taking samples from the reader queue or older samples being removed due to KEEP_LAST.

For unkeyed types this value has to be equal to max_samples.

DDS_DataWriter

This resource-limit limits the maximum number of samples that can be cached for a single instance; it must be less than or equal to max_samples and greater or equal to the DDS_HistoryQosPolicy::depth.

DDS_HistoryQosPolicy::depth determines how many samples are kept in case of DDS_TRANSIENT_LOCAL_DURABILITY_QOS durability and made available to late joining DDS_DataReader, while DDS_ResourceLimitsQosPolicy::max_samples_per_instance is the maximum number of *unacknowledged* samples the DDS_DataWriter can keep per instance. Thus, if max_samples_per_instance $>$ depth, it is possible to keep unacknowledged samples while limiting the number of samples made available to late joining DataReaders.

[default] 1

[range] [1, 100 million], $<=$ max_samples, $>=$ DDS_HistoryQosPolicy::depth

6.54 DDS_RtpsReliableReaderProtocol_t Struct Reference

Qos related to reliable reader protocol defined in RTPS.

Data Fields

- struct DDS_Duration_t nack_period

  The period at which to send NACKs.

6.54.1 Detailed Description

Qos related to reliable reader protocol defined in RTPS.

It is used to configure the reliable reader according to RTPS protocol.

Properties:

RxO = N/A
Changeable = NO

QoS:

DDS_DataReaderProtocolQosPolicy

6.54.2 Field Documentation

6.54.2.1 struct DDS_Duration_t DDS_RtpsReliableReaderProtocol_t::nack_period

The period at which to send NACKs.

A reliable reader will send periodic NACKs at this rate when it first matches with a reliable writer. The reader will stop sending NACKs when it has received all available historical data from the writer. Note that this NACK period only determines the rate at which NACKs are sent until the first HB is received from the writer. When the first HB is received, the writer's HB period determines how fast historical data is sent.

[default] 50 ms

[range] [1 nanosec, 1 year]

6.55 DDS_RtpsReliableWriterProtocol_t Struct Reference

<<cert>> QoS related to the reliable writer protocol defined in RTPS.

Data Fields

- struct DDS_Duration_t heartbeat_period

  The period at which to send heartbeats.
- DDS_Long heartbeats_per_max_samples

  The number of heartbeats per send queue.
- DDS_Long max_send_window

  The maximum size of a reliable send window.
- DDS_Long max_heartbeat_retries

  The maximum number of heartbeat retries before marking a remote reader as inactive.

6.55.1 Detailed Description

<<cert>> QoS related to the reliable writer protocol defined in RTPS.

It is used to configure a reliable writer according to RTPS protocol.

Properties:

RxO = N/A
Changeable = NO

QoS:

DDS_DataWriterProtocolQosPolicy

6.55.2 Field Documentation

6.55.2.1 struct DDS_Duration_t DDS_RtpsReliableWriterProtocol_t::heartbeat_period

The period at which to send heartbeats.

A reliable writer will send periodic heartbeat at this rate.

[default] 3.0 seconds

[range] [1 nanosec,1 year]

6.55.2.2 DDS_Long DDS_RtpsReliableWriterProtocol_t::heartbeats_per_max_samples

The number of heartbeats per send queue.

A piggyback heartbeat will be sent every [DDS_ResourceLimitsQosPolicy::max_samples/heartbeats_per_max_samples] number of samples.

If set to zero, no piggyback heartbeat will be sent. If DDS_ResourceLimitsQosPolicy::max_samples is DDS_LENGTH_UNLIMITED, 100 million is assumed as the maximum value in the calculation.

[default] 1

[range] [0, 100 million]

- For DDS_DataWriterProtocolQosPolicy::rtps_reliable_writer:
  heartbeats_per_max_samples<= DDS_ResourceLimitsQosPolicy::max_samples

6.55.2.3 DDS_Long DDS_RtpsReliableWriterProtocol_t::max_send_window

The maximum size of a reliable send window.

A reliable DataWriter may maintain a send window of samples in-flight that have not been acknowledged by its DataReaders. The max_send_window is the maximum number of unacknowledged samples in-flight. When the window is at max_send_window, subsequent writes will pass but not publish any data over the network.

[default] DDS_LENGTH_UNLIMITED, meaning the window will be as large as resource limits allow

[range] [1, 256] of DDS_LENGTH_UNLIMITED

6.55.2.4 DDS_Long DDS_RtpsReliableWriterProtocol_t::max_heartbeat_retries

The maximum number of heartbeat retries before marking a remote reader as inactive.

When a remote reader has not acked all the samples the reliable writer has in its queue, and max_heartbeat_retries number of periodic heartbeats has been sent without receiving any ack/nack back, the remote reader will be marked as inactive (not alive) and be ignored until it resumes sending ack/nack.

[default] DDS_LENGTH_UNLIMITED

[range] [1, 1 million] or DDS_LENGTH_UNLIMITED

6.56 DDS_RtpsWellKnownPorts_t Struct Reference

RTPS well-known port mapping configuration.

Data Fields

- DDS_Long port_base

  The base port offset.
- DDS_Long domain_id_gain

  Tunable domain gain parameter.
- DDS_Long participant_id_gain

  Tunable participant gain parameter.
- DDS_Long builtin_multicast_port_offset

  *Additional offset for **metatraffic** multicast port.*
- DDS_Long builtin_unicast_port_offset

  *Additional offset for **metatraffic** unicast port.*
- DDS_Long user_multicast_port_offset

  *Additional offset for **usertraffic** multicast port.*
- DDS_Long user_unicast_port_offset

  *Additional offset for **usertraffic** unicast port.*

6.56.1 Detailed Description

RTPS well-known port mapping configuration.

RTI Connext Micro uses the RTPS wire protocol. The discovery protocols defined by RTPS rely on well-known ports to initiate discovery. These well-known ports define the multicast and unicast ports on which a Participant will listen for discovery **metatraffic** from other Participants. The discovery metatraffic contains all the information required to establish the presence of remote DDS entities in the network.

The well-known ports are defined by RTPS in terms of port mapping expressions with several tunable parameters, which allow the user to customize what network ports are used by the middleware. These parameters are exposed in DDS_RtpsWellKnownPorts_t. In order for all Participants in a system to correctly discover eachother, it is important that they all use the same port mapping expressions.

The actual port mapping expressions, as defined by the RTPS specification, can be found below. In addition to the parameters listed in DDS_RtpsWellKnownPorts_t, the port numbers depend on:

- `domain_id`, as specified in DDS_DomainParticipantFactory_create_participant

- `participant_id`, as specified using DDS_WireProtocolQosPolicy::participant_id

The `domain_id` parameter ensures no port conflicts exist between Participants belonging to different domains. This also means that discovery metatraffic in one domain is not visible to Participants in a different domain. The `participant_id` parameter ensures that unique unicast port numbers are assigned to Participants belonging to the same domain on a given host.

The *metatraffic_unicast_port* is used to exchange discovery metatraffic using unicast.

```
metatraffic_unicast_port = port_base + (domain_id_gain * domain_id) + (participant_id_gain * participant_id) +
```

The *metatraffic_multicast_port* is used to exchange discovery metatraffic using multicast.

```
metatraffic_multicast_port = port_base + (domain_id_gain * domain_id) + builtin_multicast_port_offset
```

RTPS also defines the *default* multicast and unicast ports on which DataReaders and DataWriters receive **usertraffic**.

The *usertraffic_unicast_port* is used to exchange user data using unicast.

```
usertraffic_unicast_port = port_base + (domain_id_gain * domain_id) + (participant_id_gain * participant_id) +
```

The *usertraffic_multicast_port* is used to exchange user data using multicast. The corresponding multicast group addresses can be configured using

```
usertraffic_multicast_port = port_base + (domain_id_gain * domain_id) + user_multicast_port_offset
```

By default, the port mapping parameters are configured to compliant with OMG's DDS Interoperability Wire Protocol (see also DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS).

The OMG's DDS Interoperability Wire Protocol compliant port mapping parameters are *not* backwards compatible with previous versions of the RTI Connext Micro middleware.

When modifying the port mapping parameters, care must be taken to avoid port aliasing. This would result in undefined discovery behavior. The chosen parameter values will also determine the maximum possible number of domains in the system and the maximum number of participants per domain. Additionally, any resulting mapped port number must be within the range imposed by the underlying transport. For example, for UDPv4, this range typically equals [1024 - 65535].

QoS:

DDS_WireProtocolQosPolicy

6.56.2 Field Documentation

6.56.2.1 DDS_Long DDS_RtpsWellKnownPorts_t::port_base

The base port offset.

All mapped well-known ports are offset by this value.

[default] 7400

[range] [>= 1], but resulting ports must be within the range imposed by the underlying transport.

6.56.2.2 DDS_Long DDS_RtpsWellKnownPorts_t::domain_id_gain

Tunable domain gain parameter.

Multiplier of the `domain_id`. Together with `participant_id_gain`, it determines the highest `domain_id` and `participant_id` allowed on this network.

In general, there are two ways to setup `domain_id_gain` and `participant_id_gain` parameters.

If `domain_id_gain` > `participant_id_gain`, it results in a port mapping layout where all DDS_DomainParticipant instances within a single domain occupy a consecutive range of `domain_id_gain` ports. Precisely, all ports occupied by the domain fall within:

```
(port_base + (domain_id_gain * domain_id))
```

and:

```
(port_base + (domain_id_gain * (domain_id + 1)) - 1)
```

Under such a case, the highest `domain_id` is limited only by the underlying transport's maximum port. The highest `participant_id`, however, must satisfy:

```
max_participant_id < (domain_id_gain / participant_id_gain)
```

On the contrary, if `domain_id_gain` <= `participant_id_gain`, it results in a port mapping layout where a given domain's DDS_DomainParticipant instances occupy ports spanned across the entire valid port range allowed by the underlying transport. For instance, it results in the following potential mapping:

Mapped Port	Domain Id	Participant ID
higher port number	Domain Id = 1	Participant ID = 2
	Domain Id = 0	Participant ID = 2
	Domain Id = 1	Participant ID = 1
	Domain Id = 0	Participant ID = 1
	Domain Id = 1	Participant ID = 0
lower port number	Domain Id = 0	Participant ID = 0

Under this case, the highest `participant_id` is limited only by the underlying transport's maximum port. The highest `domain_id`, however, must satisfy:

```
max_domain_id < (participant_id_gain / domain_id_gain)
```

Additionally, `domain_id_gain` also determines the range of the port-specific offsets.

```
domain_id_gain > abs(builtin_multicast_port_offset - user_multicast_port_offset)

domain_id_gain > abs(builtin_unicast_port_offset - user_unicast_port_offset)
```

Violating this may result in port aliasing and undefined discovery behavior.

[default] 250

[range] [> 0], but resulting ports must be within the range imposed by the underlying transport.

6.56.2.3 DDS_Long DDS_RtpsWellKnownPorts_t::participant_id_gain

Tunable participant gain parameter.

Multiplier of the `participant_id`. See DDS_RtpsWellKnownPorts_t::domain_id_gain for its implications on the highest `domain_id` and `participant_id` allowed on this network.

Additionally, `participant_id_gain` also determines the range of `builtin_unicast_port_offset` and `user_unicast_port_offset`.

```
participant_id_gain > abs(builtin_unicast_port_offset - user_unicast_port_offset)
```

[default] 2

[range] [> 0], but resulting ports must be within the range imposed by the underlying transport.

6.56.2.4 DDS_Long DDS_RtpsWellKnownPorts_t::builtin_multicast_port_offset

Additional offset for **metatraffic** multicast port.

It must be unique from other port-specific offsets.

[default] 0

[range] [>= 0], but resulting ports must be within the range imposed by the underlying transport.

6.56.2.5 DDS_Long DDS_RtpsWellKnownPorts_t::builtin_unicast_port_offset

Additional offset for **metatraffic** unicast port.

It must be unique from other port-specific offsets.

[default] 10

[range] [>= 0], but resulting ports must be within the range imposed by the underlying transport.

6.56.2.6 DDS_Long DDS_RtpsWellKnownPorts_t::user_multicast_port_offset

Additional offset for **usertraffic** multicast port.

It must be unique from other port-specific offsets.

[default] 1

[range] [>= 0], but resulting ports must be within the range imposed by the underlying transport.

6.56.2.7 DDS_Long DDS_RtpsWellKnownPorts_t::user_unicast_port_offset

Additional offset for **usertraffic** unicast port.

It must be unique from other port-specific offsets.

[default] 11

[range] [>= 0], but resulting ports must be within the range imposed by the underlying transport.

6.57 DDS_SampleInfo Struct Reference

Information that accompanies each sample that is `read` or `taken`.

Data Fields

- DDS_SampleStateKind sample_state

  The sample state of the sample.
- DDS_ViewStateKind view_state

  The view state of the instance.
- DDS_InstanceStateKind instance_state

  The instance state of the instance.
- struct DDS_Time_t source_timestamp

  The timestamp when the sample was written by a DataWriter.
- DDS_InstanceHandle_t instance_handle

  Identifies locally the corresponding instance.
- DDS_InstanceHandle_t publication_handle

  Identifies locally the DataWriter that modified the instance.
- DDS_Boolean valid_data

  Indicates whether the `DataSample` contains data or else it is only used to communicate a change in the `instance_state` of the instance.
- struct DDS_Time_t reception_timestamp

  <<eXtension>> The timestamp when the sample was committed by a DataReader.
- struct DDS_SequenceNumber_t publication_sequence_number

  <<eXtension>> The publication sequence number.

6.57.1 Detailed Description

Information that accompanies each sample that is `read` or `taken`.

6.57.2 Interpretation of the SampleInfo

The DDS_SampleInfo contains information pertaining to the associated `Data` instance sample including:

- the `sample_state` of the `Data` value (i.e., if it has already been read or not)
- the `view_state` of the related instance (i.e., if the instance is new or not)
- the `instance_state` of the related instance (i.e., if the instance is alive or not)
- the `valid_data` flag. This flag indicates whether there is data associated with the sample. Some samples do not contain data indicating only a change on the `instance_state` of the corresponding instance.
- The `source_timestamp` of the sample. This is the timestamp provided by the DDS_DataWriter at the time the sample was produced.

See also

DDS_SampleStateKind, DDS_InstanceStateKind, DDS_ViewStateKind, DDS_SampleInfo::valid_data

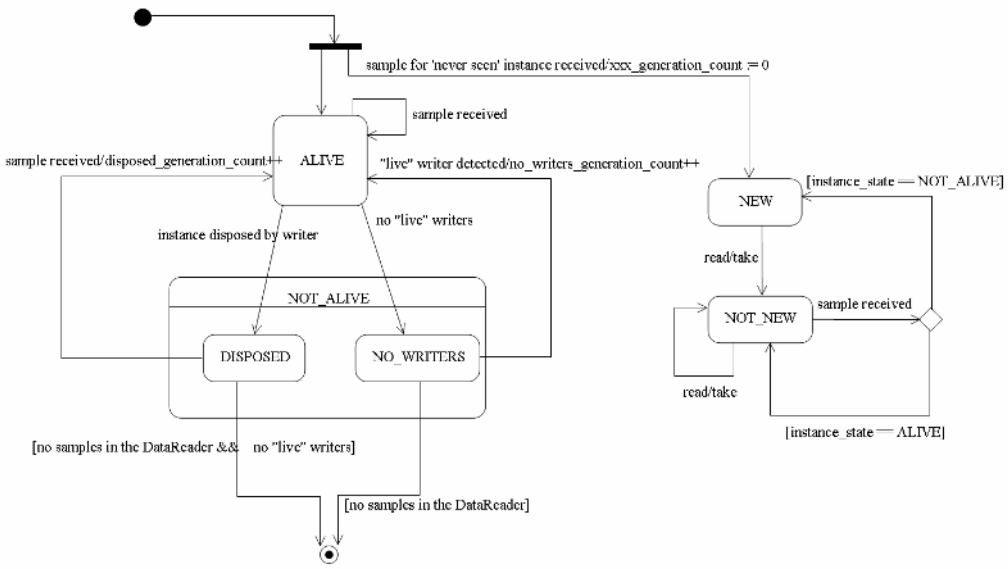

Figure 12: Statechart of the `instance_state` and `view_state` of a single instance

6.57.3 Field Documentation

6.57.3.1 DDS_SampleStateKind DDS_SampleInfo::sample_state

The sample state of the sample.

Indicates whether or not the corresponding data sample has already been read.

See also

DDS_SampleStateKind

6.57.3.2 DDS_ViewStateKind DDS_SampleInfo::view_state

The view state of the instance.

Indicates whether the DDS_DataReader has already seen samples for the most-current generation of the related instance.

See also

DDS_ViewStateKind

6.57.3.3 DDS_InstanceStateKind DDS_SampleInfo::instance_state

The instance state of the instance.

Indicates whether the instance is currently in existence or, if it has been disposed, the reason why it was disposed.

See also

DDS_InstanceStateKind

6.57.3.4 struct DDS_Time_t DDS_SampleInfo::source_timestamp

The timestamp when the sample was written by a DataWriter.

6.57.3.5 DDS_InstanceHandle_t DDS_SampleInfo::instance_handle

Identifies locally the corresponding instance.

6.57.3.6 DDS_InstanceHandle_t DDS_SampleInfo::publication_handle

Identifies locally the DataWriter that modified the instance.

6.57.3.7 DDS_Boolean DDS_SampleInfo::valid_data

Indicates whether the `DataSample` contains data or else it is only used to communicate a change in the `instance_state` of the instance.

Normally each `DataSample` contains both a DDS_SampleInfo and some Data. However there are situations where a `DataSample` contains only the DDS_SampleInfo and does not have any associated data. This occurs when the RTI Connext Micro notifies the application of a change of state for an instance that was caused by some internal mechanism (such as a timeout) for which there is no associated data. An example of this situation is when the RTI Connext Micro detects that an instance has no writers and changes the corresponding instance_state to DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.

The application can distinguish whether a particular `DataSample` has data by examining the value of the `valid_data` flag. If this flag is set to DDS_BOOLEAN_TRUE, then the `DataSample` contains valid Data. If the flag is set to DDS_BOOLEAN_FALSE, the `DataSample` contains no Data.

To ensure correctness and portability, the `valid_data` flag must be examined by the application prior to accessing the Data associated with the `DataSample` and if the flag is set to DDS_BOOLEAN_FALSE, the application should not access the Data associated with the `DataSample`, that is, the application should access only the DDS_SampleInfo.

6.57.3.8 struct DDS_Time_t DDS_SampleInfo::reception_timestamp

<<eXtension>> The timestamp when the sample was committed by a DataReader.

The reception_timestamp is platform dependent and depends on the port of RTI Connext Micro to the platform. Typically, it represents the current local time, but that may not always be the case. E.g, some platforms may not have a real-time clock and the system time is the time since the system was started.

A reception_timestamp equal to zero may mean any of the following:

- RTI Connext Micro was unable to read the current time
- The system time is not incrementing
- Zero time has elapsed between the system was started and the first sample was comitted.

6.57.3.9 struct DDS_SequenceNumber_t DDS_SampleInfo::publication_sequence_number

<<eXtension>> The publication sequence number.

6.58 DDS_SampleInfoSeq Struct Reference

Declares IDL `sequence < DDS_SampleInfo >`.

6.58.1 Detailed Description

Declares IDL `sequence < DDS_SampleInfo >`.

See also

FooSeq

6.59 DDS_SampleLostStatus Struct Reference

<<*cert*>> DDS_SAMPLE_LOST_STATUS

Data Fields

- DDS_Long total_count

  Total cumulative count of all samples lost across all instances of data published under the DDS_Topic.
- DDS_Long total_count_change

  The incremental number of samples lost since the last time the listener was called or the status was read.
- DDS_SampleLostStatusKind reason

  The reason a sample was lost.
- struct DDS_SampleInfo sample_info

  Additional information about the sample lost for reasons other than.

6.59.1 Detailed Description

<<*cert*>> DDS_SAMPLE_LOST_STATUS

6.59.2 Field Documentation

6.59.2.1 DDS_Long DDS_SampleLostStatus::total_count

Total cumulative count of all samples lost across all instances of data published under the DDS_Topic.

6.59.2.2 DDS_Long DDS_SampleLostStatus::total_count_change

The incremental number of samples lost since the last time the listener was called or the status was read.

6.59.2.3 DDS_SampleLostStatusKind DDS_SampleLostStatus::reason

The reason a sample was lost.

6.59.2.4 struct DDS_SampleInfo DDS_SampleLostStatus::sample_info

Additional information about the sample lost for reasons other than.

A sample was lost by the DataWriter, it was never received by the DataReader.

See also

DDS_ResourceLimitsQosPolicy and Sample was not lost, normal removal.

See also

DDS_ResourceLimitsQosPolicy

6.60 DDS_SampleRejectedStatus Struct Reference

DDS_SAMPLE_REJECTED_STATUS

Data Fields

- DDS_Long total_count

  Total cumulative count of samples rejected by the DDS_DataReader.
- DDS_Long total_count_change

  The incremental number of samples rejected since the last time the listener was called or the status was read.
- DDS_SampleRejectedStatusKind last_reason

  Reason for rejecting the last sample rejected.
- DDS_InstanceHandle_t last_instance_handle

  Handle to the instance being updated by the last sample that was rejected.

6.60.1 Detailed Description

DDS_SAMPLE_REJECTED_STATUS

A status structure describing the samples rejected by the reader.

6.60.2 Field Documentation

6.60.2.1 DDS_Long DDS_SampleRejectedStatus::total_count

Total cumulative count of samples rejected by the DDS_DataReader.

6.60.2.2 DDS_Long DDS_SampleRejectedStatus::total_count_change

The incremental number of samples rejected since the last time the listener was called or the status was read.

6.60.2.3 DDS_SampleRejectedStatusKind DDS_SampleRejectedStatus::last_reason

Reason for rejecting the last sample rejected.

See also

DDS_SampleRejectedStatusKind

6.60.2.4 DDS_InstanceHandle_t DDS_SampleRejectedStatus::last_instance_handle

Handle to the instance being updated by the last sample that was rejected.

6.61 DDS_SequenceNumber_t Struct Reference

Type for *sequence* number representation.

Data Fields

- DDS_Long high

  The most significant part of the sequence number.
- DDS_UnsignedLong low

  The least significant part of the sequence number.

6.61.1 Detailed Description

Type for *sequence* number representation.

Represents a 64-bit sequence number.

6.61.2 Field Documentation

6.61.2.1 DDS_Long DDS_SequenceNumber_t::high

The most significant part of the sequence number.

6.61.2.2 DDS_UnsignedLong DDS_SequenceNumber_t::low

The least significant part of the sequence number.

6.62 DDS_SubscriberListener Struct Reference

<<interface>> <<cert>> DDS_Listener for status about a subscriber.

Data Fields

- struct DDS_DataReaderListener as_datareaderlistener

  The superclass instance of this DDS_SubscriberListener.
- DDS_SubscriberListener_DataOnReadersCallback on_data_on_readers

  Handles the DDS_DATA_ON_READERS_STATUS communication status.

6.62.1 Detailed Description

<<interface>> <<cert>> DDS_Listener for status about a subscriber.

Entity:

DDS_Subscriber

Status:

DDS_DATA_AVAILABLE_STATUS;
DDS_DATA_ON_READERS_STATUS;
DDS_LIVELINESS_CHANGED_STATUS, DDS_LivelinessChangedStatus;
DDS_REQUESTED_DEADLINE_MISSED_STATUS, DDS_RequestedDeadlineMissedStatus;
DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS, DDS_RequestedIncompatibleQosStatus;
DDS_SAMPLE_LOST_STATUS, DDS_SampleLostStatus;
DDS_SAMPLE_REJECTED_STATUS, DDS_SampleRejectedStatus;
DDS_SUBSCRIPTION_MATCHED_STATUS, DDS_SubscriptionMatchedStatus;

See also

DDS_Listener
Status Kinds

6.62.2 Field Documentation

6.62.2.1 struct DDS_DataReaderListener DDS_SubscriberListener::as_datareaderlistener

The superclass instance of this DDS_SubscriberListener.

6.62.2.2 DDS_SubscriberListener_DataOnReadersCallback DDS_SubscriberListener::on_data_on_readers

Handles the DDS_DATA_ON_READERS_STATUS communication status.

6.63 DDS_SubscriberQos Struct Reference

<<cert>> QoS policies supported by a DDS_Subscriber entity.

Data Fields

- struct DDS_EntityFactoryQosPolicy entity_factory

  Entity factory policy, ENTITY_FACTORY.

6.63.1 Detailed Description

<<cert>> QoS policies supported by a DDS_Subscriber entity.

You must set the members in a consistent manner. If not DDS_DomainParticipant_create_subscriber will fail.

6.63.2 Field Documentation

6.63.2.1 struct DDS_EntityFactoryQosPolicy DDS_SubscriberQos::entity_factory

Entity factory policy, ENTITY_FACTORY.

6.64 DDS_SubscriptionBuiltinTopicData Struct Reference

Entry created when a DDS_DataReader is discovered in association with its Subscriber.

Data Fields

- struct DDS_BuiltinTopicKey_t key

  DCPS key to distinguish entries. This is used to configure the object ID of the remote DDS_DataReader.
- struct DDS_BuiltinTopicKey_t participant_key

  DCPS key of the participant to which the DataReader belongs.
- char * topic_name

  Name of the related DDS_Topic.
- char * type_name

  Name of the type attached to the DDS_Topic.
- struct DDS_DeadlineQosPolicy deadline

  Policy of the corresponding DataReader.
- struct DDS_OwnershipQosPolicy ownership

  Policy of the corresponding DataReader.
- struct DDS_ReliabilityQosPolicy reliability

  Policy of the corresponding DataReader.
- struct DDS_LivelinessQosPolicy liveliness

  Policy of the corresponding DataReader.
- struct DDS_DurabilityQosPolicy durability

  Policy of the corresponding DataReader.
- struct DDS_DestinationOrderQosPolicy destination_order

  Policy of the corresponding DataReader.
- struct DDS_LocatorSeq unicast_locator

  Custom unicast locator sequence that the endpoint can specify. The default locator sequence from the DDS_DomainParticipant will be used if this is not specified.

- struct DDS_LocatorSeq multicast_locator

  Custom multicast locator that the endpoint can specify. The default multicast locator sequence from the DDS_DomainParticipant will be used if this is not specified.

6.64.1 Detailed Description

Entry created when a DDS_DataReader is discovered in association with its Subscriber.

Data associated with the built-in topic DDS_SUBSCRIPTION_TOPIC_NAME. It contains QoS policies and additional information that apply to the remote DDS_DataReader the related DDS_Subscriber.

6.64.2 Field Documentation

6.64.2.1 struct DDS_BuiltinTopicKey_t DDS_SubscriptionBuiltinTopicData::key

DCPS key to distinguish entries. This is used to configure the object ID of the remote DDS_DataReader.

Key GUID must be first

6.64.2.2 struct DDS_BuiltinTopicKey_t DDS_SubscriptionBuiltinTopicData::participant_key

DCPS key of the participant to which the DataReader belongs.

6.64.2.3 char* DDS_SubscriptionBuiltinTopicData::topic_name

Name of the related DDS_Topic.

The length of this string is limited to 255 characters.

The memory for this field is managed as described in String Support.

See also

String Support

6.64.2.4 char* DDS_SubscriptionBuiltinTopicData::type_name

Name of the type attached to the DDS_Topic.

The length of this string is limited to 255 characters.

The memory for this field is managed as described in String Support.

See also

String Support

6.64.2.5 struct DDS_DeadlineQosPolicy DDS_SubscriptionBuiltinTopicData::deadline

Policy of the corresponding DataReader.

6.64.2.6 struct DDS_OwnershipQosPolicy DDS_SubscriptionBuiltinTopicData::ownership

Policy of the corresponding DataReader.

6.64.2.7 struct DDS_ReliabilityQosPolicy DDS_SubscriptionBuiltinTopicData::reliability

Policy of the corresponding DataReader.

6.64.2.8 struct DDS_LivelinessQosPolicy DDS_SubscriptionBuiltinTopicData::liveliness

Policy of the corresponding DataReader.

6.64.2.9 struct DDS_DurabilityQosPolicy DDS_SubscriptionBuiltinTopicData::durability

Policy of the corresponding DataReader.

6.64.2.10 struct DDS_DestinationOrderQosPolicy DDS_SubscriptionBuiltinTopicData::destination_order

Policy of the corresponding DataReader.

6.64.2.11 struct DDS_LocatorSeq DDS_SubscriptionBuiltinTopicData::unicast_locator

Custom unicast locator sequence that the endpoint can specify. The default locator sequence from the DDS_DomainParticipant will be used if this is not specified.

6.64.2.12 struct DDS_LocatorSeq DDS_SubscriptionBuiltinTopicData::multicast_locator

Custom multicast locator that the endpoint can specify. The default multicast locator sequence from the DDS_DomainParticipant will be used if this is not specified.

6.65 DDS_SubscriptionMatchedStatus Struct Reference

<<*cert*>> DDS_SUBSCRIPTION_MATCHED_STATUS

Data Fields

- DDS_Long total_count

  The total cumulative number of times the concerned DDS_DataReader discovered a "match" with a DDS_DataWriter.
- DDS_Long total_count_change

  The change in total_count since the last time the listener was called or the status was read.
- DDS_Long current_count

  The current number of writers with which the DDS_DataReader is matched.
- DDS_Long current_count_change

  The change in current_count since the last time the listener was called or the status was read.
- DDS_InstanceHandle_t last_publication_handle

  A handle to the last DDS_DataWriter that caused the status to change.

6.65.1 Detailed Description

<<*cert*>> DDS_SUBSCRIPTION_MATCHED_STATUS

A "match" happens when the DDS_DataReader finds a DDS_DataWriter for the same DDS_Topic with an offered QoS that is compatible with that requested by the DDS_DataReader.

This status is also changed (and the listener, if any, called) when a match is ended. A local DDS_DataReader will become "unmatched" from a remote DDS_DataWriter when that DDS_DataWriter goes away for any reason.

6.65.2 Field Documentation

6.65.2.1 DDS_Long DDS_SubscriptionMatchedStatus::total_count

The total cumulative number of times the concerned DDS_DataReader discovered a "match" with a DDS_DataWriter.

This number increases whenever a new match is discovered. It does not change when an existing match goes away.

6.65.2.2 DDS_Long DDS_SubscriptionMatchedStatus::total_count_change

The change in total_count since the last time the listener was called or the status was read.

6.65.2.3 DDS_Long DDS_SubscriptionMatchedStatus::current_count

The current number of writers with which the DDS_DataReader is matched.

This number increases when a new match is discovered and decreases when an existing match goes away.

6.65.2.4 DDS_Long DDS_SubscriptionMatchedStatus::current_count_change

The change in current_count since the last time the listener was called or the status was read.

6.65.2.5 DDS_InstanceHandle_t DDS_SubscriptionMatchedStatus::last_publication_handle

A handle to the last DDS_DataWriter that caused the status to change.

6.66 DDS_SystemResourceLimitsQosPolicy Struct Reference

<<cert>> Resource limits that apply only to DDS_DomainParticipantFactory

Data Fields

- DDS_Long max_participants

  The maximum number of DomainParticipants (DDS_DomainParticipant) the DDS_DomainParticipantFactory can manange.
- DDS_Long max_components

  The maximum number of components that may be registered on the DDS_DomainParticipantFactory.

6.66.1 Detailed Description

<<cert>> Resource limits that apply only to DDS_DomainParticipantFactory

This QoS policy is an extension to the DDS standard.

Entity:

DDS_DomainParticipantFactory

Properties:

RxO = N/A
Changeable = NO

6.66.2 Field Documentation

6.66.2.1 DDS_Long DDS_SystemResourceLimitsQosPolicy::max_participants

The maximum number of DomainParticipants (DDS_DomainParticipant) the DDS_DomainParticipantFactory can manange.

This resource-limit limits the number of local DomainParticipants that can be created with the DDS_DomainParticipantFactory_create_participant() API. For the maximum number of participants that can be discovered, please refer to DDS_DomainParticipantResourceLimitsQosPolicy::remote_participant_allocation.

The **actual** number of DomainParticipants that can be **created** with DDS_DomainParticipantFactory_create_participant is further limited by the following constraints:

- Available memory and system resources such as network sockets and shared memory resources.
- The maximum number of system timers available on the platform. Each DDS_DomainParticipant requires one timer plus one timer if Zero Copy v2 is enabled as a transport in DDS_TransportQosPolicy::enabled_transports. The Reference Platform Support Library (PSL) for POSIX platforms out of the box has a built-in limit of 8 timers.
- DDS uses The DDS_WireProtocolQosPolicy::rtps_well_known_ports to discovery other participant. The number of DomainParticipants that can be created is limited by port aliasing where ports calculated for different DDS domain IDs may overlap. RTI Connext Micro does not determine if port aliasing may occur and the application must ensure that port aliasing does not occur.

[default] 1

[range] [1, 16]

6.66.2.2 DDS_Long DDS_SystemResourceLimitsQosPolicy::max_components

The maximum number of components that may be registered on the DDS_DomainParticipantFactory.

RTI Connext Micro consists of a number of plugins and components that perform various tasks. For example, the UDP transport and the DPDE discovery plugins are RTI Connext Micro components. This resource-limit limits the number of components that can be registered with RTI Connext Micro.

Note that changing this value is not supported.

[default] 64

6.67 DDS_Time_t Struct Reference

Type for *time* representation.

Data Fields

- DDS_LongLong sec
  seconds
- DDS_UnsignedLong nanosec
  nanoseconds

6.67.1 Detailed Description

Type for *time* representation.

A DDS_Time_t represents a moment in time.

6.67.2 Field Documentation

6.67.2.1 DDS_LongLong DDS_Time_t::sec

seconds

6.67.2.2 DDS_UnsignedLong DDS_Time_t::nanosec

nanoseconds

6.68 DDS_TopicListener Struct Reference

<<*interface*>> DDS_Listener for DDS_Topic entities.

Data Fields

- struct DDS_Listener as_listener

  The superclass instance of this DDS_TopicListener.
- DDS_TopicListener_InconsistentTopicCallback on_inconsistent_topic

  Handle the DDS_INCONSISTENT_TOPIC_STATUS status.

6.68.1 Detailed Description

<<interface>> DDS_Listener for DDS_Topic entities.

Entity:

DDS_Topic

Status:

DDS_INCONSISTENT_TOPIC_STATUS, DDS_InconsistentTopicStatus

This is the interface that can be implemented by an application-provided class and then registered with the DDS_-Topic such that the application can be notified by RTI Connext Micro of relevant status changes.

See also

Status Kinds
DDS_Listener

6.68.2 Field Documentation

6.68.2.1 struct DDS_Listener DDS_TopicListener::as_listener

The superclass instance of this DDS_TopicListener.

6.68.2.2 DDS_TopicListener_InconsistentTopicCallback DDS_TopicListener::on_inconsistent_topic

Handle the DDS_INCONSISTENT_TOPIC_STATUS status.

This callback is called when a remote DDS_Topic is discovered but is inconsistent with the locally created DDS_-Topic of the same topic name.

6.69 DDS_TopicQos Struct Reference

QoS policies supported by a DDS_Topic entity.

6.69.1 Detailed Description

QoS policies supported by a DDS_Topic entity.

Getting and setting the topic QoS is not supported. DDS_TOPIC_QOS_DEFAULT is the only value supported as an argument to DDS_DomainParticipant_create_topic .

Entity:

DDS_Topic

6.70 DDS_TransportQosPolicy Struct Reference

<<eXtension>> *<<cert>>* Settings for available transports

Data Fields

- struct DDS_StringSeq enabled_transports

  The transports available for use.

6.70.1 Detailed Description

<<eXtension>> *<<cert>>* Settings for available transports

Entity:

DDS_DomainParticipant, DDS_DataWriter, DDS_DataReader

Properties:

RxO = N/A
Changeable = NO

6.70.2 Field Documentation

6.70.2.1 struct DDS_StringSeq DDS_TransportQosPolicy::enabled_transports

The transports available for use.

A sequence of transport identifiers that defines the set of transports available for use by the associated DDS entity.

See also

DDS_DomainParticipantQos, DDS_DataWriterQos, DDS_DataReaderQos

[default] Empty sequence.

[range] Sequence of non-null,non-empty strings.

6.71 DDS_UserTrafficQosPolicy Struct Reference

A QosPolicy to configure properties of network traffic caused by the exchange of user data.

Data Fields

- struct DDS_StringSeq enabled_transports

  Configures the default transports enabled to transmit and received user data traffic.

6.71.1 Detailed Description

A QosPolicy to configure properties of network traffic caused by the exchange of user data.

6.71.2 Field Documentation

6.71.2.1 struct DDS_StringSeq DDS_UserTrafficQosPolicy::enabled_transports

Configures the default transports enabled to transmit and received user data traffic.

The memory for the strings in this sequence is managed according to the conventions described in String Support. In particular, be careful to avoid a situation in which RTI Connext Micro allocates a string on your behalf and you

then reuse that string in such a way that RTI Connext Micro believes it to have more memory allocated to it than it actually does.

[default] Empty sequence.

When an empty sequence is passed to DDS_DomainParticipantFactory_create_participant in the DDS_DomainParticipantQos, the domain participant will automatically add the following locators to listen for user-traffic on: "_udp://" and "_intra://".

[range] Sequence of non-null,non-empty strings.

6.72 DDS_VendorId Struct Reference

<<eXtension>> *<<cert>>* Type used to represent the vendor of the service implementing the RTPS protocol.

Data Fields

- DDS_Octet vendorId [DDS_VENDOR_ID_LENGTH_MAX]

  The vendor Id.

6.72.1 Detailed Description

<<eXtension>> *<<cert>>* Type used to represent the vendor of the service implementing the RTPS protocol.

6.72.2 Field Documentation

6.72.2.1 DDS_Octet DDS_VendorId::vendorId[DDS_VENDOR_ID_LENGTH_MAX]

The vendor Id.

6.73 DDS_VendorId_t Struct Reference

<<eXtension>> *<<cert>>* Type used to represent the vendor of the service implementing the RTPS protocol.

Data Fields

- DDS_Octet vendorId [DDS_VENDOR_ID_LENGTH_MAX]

  The vendor Id.

6.73.1 Detailed Description

<<eXtension>> *<<cert>>* Type used to represent the vendor of the service implementing the RTPS protocol.

6.73.2 Field Documentation

6.73.2.1 DDS_Octet DDS_VendorId_t::vendorId[DDS_VENDOR_ID_LENGTH_MAX]

The vendor Id.

6.74 DDS_WireProtocolQosPolicy Struct Reference

<<cert>> Specifies the wire-protocol-related attributes for the DDS_DomainParticipant.

Data Fields

- DDS_Long participant_id

  A value used to distinguish between different participants belonging to the same domain on the same host.
- DDS_UnsignedLong rtps_host_id

  The RTPS Host ID of the domain participant.
- DDS_UnsignedLong rtps_app_id

  The RTPS App ID of the domain participant.
- DDS_UnsignedLong rtps_instance_id

  The RTPS Instance ID of the domain participant.
- struct DDS_RtpsWellKnownPorts_t rtps_well_known_ports

  Configures the RTPS well-known port mappings.
- DDS_Boolean compute_crc

  Adds RTPS checksum submessage to every message when this field is set to DDS_BOOLEAN_TRUE.
- DDS_Boolean check_crc

  Checks if the received RTPS message is valid by comparing the computed checksum with the received RTPS checksum submessage when this field is set to DDS_BOOLEAN_TRUE.
- DDS_Boolean require_crc

  Require that discovered participants send one of the checksums enabled in DDS_WireProtocolQosPolicy::allowed_crc_mask.
- DDS_ChecksumKind_t computed_crc_kind

  A bitmask indicating which checksum is used for sent messages when compute_crc is DDS_BOOLEAN_TRUE. Only one mask is supported.
- DDS_ChecksumKindMask_t allowed_crc_mask

  The checksums which are enabled for validating incoming packets.

6.74.1 Detailed Description

<<cert>> Specifies the wire-protocol-related attributes for the DDS_DomainParticipant.

Entity:

DDS_DomainParticipant

Properties:

RxO = N/A
Changeable = NO

6.74.2 Usage

The default Qos policies returned by RTI Connext Micro contain the correctly initialized wire protocol attributes.

The defaults are not expected to be modified normally, but are available to the advanced user customizing the implementation behavior.

The default values should not be modified without an understanding of the underlying Real-Time Publish Subscribe (RTPS) wire protocol.

6.74.3 Field Documentation

6.74.3.1 DDS_Long DDS_WireProtocolQosPolicy::participant_id

A value used to distinguish between different participants belonging to the same domain on the same host.

Determines the unicast port on which meta-traffic is received. Also defines the *default* unicast port for receiving user-traffic for DataReaders and DataWriters .

For more information on port mapping, please refer to DDS_RtpsWellKnownPorts_t.

Each DDS_DomainParticipant in the same domain, running on the same host, must have a unique `participant_id`. The participants may be in the same address space or in distinct address spaces.

A negative number (-1) means that RTI Connext Micro will *automatically* resolve the participant ID as follows.

- RTI Connext Micro will pick the *smallest* participant ID, based on the unicast ports available on the transports enabled for discovery.
- RTI Connext Micro will attempt to resolve an automatic port index either when a DomainPartcipant is enabled, or when a DataReader or a DataWriter is created. Therefore, all the transports enabled for discovery must have been registered by this time. Otherwise, the discovery transports registered after resolving the automatic port index may produce port conflicts when the DomainPartcipant is enabled.

[default] -1 [automatic], i.e. RTI Connext Micro will automatically pick the `participant_id`, as described above.

[range] [>= 0], or -1, and does not violate guidelines stated in DDS_RtpsWellKnownPorts_t.

See also

DDS_Entity_enable()

6.74.3.2 DDS_UnsignedLong DDS_WireProtocolQosPolicy::rtps_host_id

The RTPS Host ID of the domain participant.

Either all of DDS_WireProtocolQosPolicy::rtps_host_id, DDS_WireProtocolQosPolicy::rtps_app_id, and DDS_WireProtocolQosPolicy::rtps_instance_id for a DomainParticipant are set to DDS_RTPS_AUTO_ID, or none of them are set to DDS_RTPS_AUTO_ID.

[default] DDS_RTPS_AUTO_ID.

[range] [0,0xffffffff]

6.74.3.3 DDS_UnsignedLong DDS_WireProtocolQosPolicy::rtps_app_id

The RTPS App ID of the domain participant.

Either all of DDS_WireProtocolQosPolicy::rtps_host_id, DDS_WireProtocolQosPolicy::rtps_app_id, and DDS_WireProtocolQosPolicy::rtps_instance_id for a DomainParticipant are set to DDS_RTPS_AUTO_ID, or none of them are set to DDS_RTPS_AUTO_ID.

[default] DDS_RTPS_AUTO_ID.

[range] [0,0xffffffff]

6.74.3.4 DDS_UnsignedLong DDS_WireProtocolQosPolicy::rtps_instance_id

The RTPS Instance ID of the domain participant.

Either all of DDS_WireProtocolQosPolicy::rtps_host_id, DDS_WireProtocolQosPolicy::rtps_app_id, and DDS_WireProtocolQosPolicy::rtps_instance_id for a DomainParticipant are set to DDS_RTPS_AUTO_ID, or none of them are set to DDS_RTPS_AUTO_ID.

[default] DDS_RTPS_AUTO_ID.

[range] [0,0xffffffff]

6.74.3.5 struct DDS_RtpsWellKnownPorts_t DDS_WireProtocolQosPolicy::rtps_well_known_ports

Configures the RTPS well-known port mappings.

Determines the well-known multicast and unicast port mappings for discovery (meta) traffic and user traffic.

[default] DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS

6.74.3.6 DDS_Boolean DDS_WireProtocolQosPolicy::compute_crc

Adds RTPS checksum submessage to every message when this field is set to DDS_BOOLEAN_TRUE.

The computed checksum covers the entire RTPS message including the RTPS header itself.

[default] DDS_BOOLEAN_FALSE

6.74.3.7 DDS_Boolean DDS_WireProtocolQosPolicy::check_crc

Checks if the received RTPS message is valid by comparing the computed checksum with the received RTPS checksum submessage when this field is set to DDS_BOOLEAN_TRUE.

DDS_WireProtocolQosPolicy::compute_crc must be enabled at the publishing application for validating the message at the subscribing application.

If a message is reeived with an invalid checksum, the message is discarded. If no checksum is received or found, the message is accepted but may still be corrupted.

[default] DDS_BOOLEAN_FALSE

6.74.3.8 DDS_Boolean DDS_WireProtocolQosPolicy::require_crc

Require that discovered participants send one of the checksums enabled in DDS_WireProtocolQosPolicy::allowed_crc_mask.

If an invalid checksum is received, the message is dropped.

if the value is DDS_BOOLEAN_TRUE the participant requires that discovered participants send one of the checksums enabled in DDS_WireProtocolQosPolicy::allowed_crc_mask. If this is not the case the participant is ignored and not matching will take place.

This is a stronger check than idref_WireProtocolQosPolicy_check_crc. If an invalid checksum is received or not found, the message is dropped.

[default] DDS_BOOLEAN_FALSE

6.74.3.9 DDS_ChecksumKind_t DDS_WireProtocolQosPolicy::computed_crc_kind

A bitmask indicating which checksum is used for sent messages when compute_crc is DDS_BOOLEAN_TRUE. Only one mask is supported.

- DDS_CHECKSUM_AUTO
- DDS_CHECKSUM_BUILTIN32
- DDS_CHECKSUM_BUILTIN64
- DDS_CHECKSUM_BUILTIN128

[default] DDS_CHECKSUM_AUTO.

6.74.3.10 DDS_ChecksumKindMask_t DDS_WireProtocolQosPolicy::allowed_crc_mask

The checksums which are enabled for validating incoming packets.

The following bitmask can be OR'ed for all enabled checksums:

- DDS_CHECKSUM_AUTO
- DDS_CHECKSUM_BUILTIN32
- DDS_CHECKSUM_BUILTIN64

- DDS_CHECKSUM_BUILTIN128

Note that the selected checksums must be a subset of what is supported. If a bitmask that is not supported is selected participant creation will fail.

[default] DDS_CHECKSUM_AUTO.

6.75 DPSE_DiscoveryPluginProperty Struct Reference

<<eXtension>> <<cert>> Properties for the Dynamic Participant/Static Endpoint (DPSE) discovery plugin. This includes all discovery timing properties for participant discovery.

Data Fields

- struct DDS_Duration_t participant_liveliness_assert_period

  The period to assert liveliness for the participant.
- struct DDS_Duration_t participant_liveliness_lease_duration

  The liveliness lease duration for the participant.
- DDS_Long initial_participant_announcements

  The number of initial announcements sent when a participant is first enabled or when a remote participant is newly discovered.
- struct DDS_Duration_t initial_participant_announcement_period

  The period between initial announcements when a participant is first enabled or when a remote participant is newly discovered.
- DDS_Long max_participant_locators

  The maximum number of locators RTI Connext Micro can send to to discover other participants in the same domain.
- DDS_Long max_locators_per_discovered_participant

  The maximum number of locators RTI Connext Micro can keep track of per discovered participant.

6.75.1 Detailed Description

<<eXtension>> <<cert>> Properties for the Dynamic Participant/Static Endpoint (DPSE) discovery plugin. This includes all discovery timing properties for participant discovery.

6.75.2 Field Documentation

6.75.2.1 struct DDS_Duration_t DPSE_DiscoveryPluginProperty::participant_liveliness_assert_period

The period to assert liveliness for the participant.

Specifies how often the DomainParticipant's discovery DataWriter will assert the liveliness of the DomainParticipant to its peers.

Should be strictly less than participant_liveliness_lease_duration.

[default] 30 seconds

[range] [1 nanosec, 1 year], < participant_liveliness_lease_duration

6.75.2.2 struct DDS_Duration_t DPSE_DiscoveryPluginProperty::participant_liveliness_lease_duration

The liveliness lease duration for the participant.

This is the same as the expiration time of the DomainParticipant as defined in the RTPS protocol.

If the participant has not refreshed its own liveliness to other participants at least once within this period, it may be considered as stale by other participants in the network.

Should be strictly greater than participant_liveliness_assert_period.

[default] 100 seconds

[range] [1 nanosec, 1 year], > participant_liveliness_assert_period

6.75.2.3 DDS_Long DPSE_DiscoveryPluginProperty::initial_participant_announcements

The number of initial announcements sent when a participant is first enabled or when a remote participant is newly discovered.

The number of initial announcements sent when a participant is first enabled or when a remote participant is newly discovered.

Also, when a new remote participant appears, the local participant will announce itself to the peers a number of times controlled by this parameter.

[default] 5

6.75.2.4 struct DDS_Duration_t DPSE_DiscoveryPluginProperty::initial_participant_announcement_period

The period between initial announcements when a participant is first enabled or when a remote participant is newly discovered.

[default] 1 second

6.75.2.5 DDS_Long DPSE_DiscoveryPluginProperty::max_participant_locators

The maximum number of locators RTI Connext Micro can send to to discover other participants in the same domain.

The DPSE builtin participant topic DDS_DataWriter allocates max_participant_locators to send participant announcements to. The default value DDS_LENGTH_AUTO calculates max_participant_locators as remote_participant_allocation + (DPSE_MAX_ANON_PARTICIPANT ∗ (peer_length + 1)) where the constant DPSE_MAX_ANON_PARTICIPANT = 6.

A participant locator is defined as a unique index@address to send to where index is an integer >= 0 and only applies to unicast addresses. For multicast address the index is 0. This index is the highest participant_id that will receive participant announcements at the address.

The maximum number of addresses to send to is defined by the peer list (either specified in DDS_DiscoveryQosPolicy::initial_peers or by calling DDS_DomainParticipant_add_peer. The maximum number of addresses needed is determined by adding all indices for all unicast addresses and adding one for each multicast address.

If memory is limited it is advised to set this value as low as possible.

[default] DDS_LENGTH_AUTO;

6.75.2.6 DDS_Long DPSE_DiscoveryPluginProperty::max_locators_per_discovered_participant

The maximum number of locators RTI Connext Micro can keep track of per discovered participant.

When a participant discover another participant, it must keep track of which addresses to respond to for each participant. NOTE: If two participants respond with the same addresses, for example multicast, only one address is saved.

[default] 4;

6.76 FooDataReader Struct Reference

<<interface>> <<generic>> User data type-specific data reader.

6.76.1 Detailed Description

<<interface>> <<generic>> User data type-specific data reader.

Defines the user data type specific reader interface generated for each application class.

The concrete user data type reader automatically generated by the implementation is an incarnation of this class.

See also

DDS_DataReader
FooDataWriter

6.77 FooDataWriter Struct Reference

<<interface>> *<<generic>>* User data type specific data writer.

6.77.1 Detailed Description

<<interface>> *<<generic>>* User data type specific data writer.

Defines the user data type specific writer interface generated for each application class.

The concrete user data type writer automatically generated by the implementation is an incarnation of this class.

See also

DDS_DataWriter
FooDataReader

6.78 FooSeq Struct Reference

<<interface>> *<<generic>>* *<<cert>>* A type-safe, ordered collection of elements. The type of these elements is referred to in this documentation as "Foo".

Data Fields

- Foo * _contiguous_buffer

  Pointer to array of contiguous or discontiguous data.
- DDS_UnsignedLong _maximum

  Maximum size of the sequence.
- DDS_UnsignedLong _length

  Actual length of the sequence that contains data.
- DDS_Long _element_size

  Size of data element in the sequence.
- void * _token1

  Reserved for internal use.
- void * _token2

  Reserved for internal use.
- DDS_UnsignedLong max_alloc_size

  The maximum allocation for pointer elements (string,wstring) Only in debug librarires.
- DDS_Char _flags

  Reserved for internal use.

6.78.1 Detailed Description

<<interface>> *<<generic>>* *<<cert>>* A type-safe, ordered collection of elements. The type of these elements is referred to in this documentation as "Foo".

We refer to an IDL sequence<Foo> as FooSeq.

The state of a sequence is described by the properties 'maximum', and 'length'.

- The 'maximum' represents the size of the underlying buffer; this is the maximum number of elements it can possibly hold. It is returned by the FooSeq_get_maximum operation.
- The 'length' represents the actual number of elements it currently holds. It is returned by the FooSeq_get_length operation.

6.78.2 Field Documentation

6.78.2.1 Foo* FooSeq::_contiguous_buffer

Pointer to array of contiguous or discontiguous data.

6.78.2.2 DDS_UnsignedLong FooSeq::_maximum

Maximum size of the sequence.

The allocated length of this sequence. It applies to whichever of the above buffers is non-NULL, if any. If both a NULL, its value must be 0.

If _maximum == 0, _owned == true.

6.78.2.3 DDS_UnsignedLong FooSeq::_length

Actual length of the sequence that contains data.

The current logical length of this seqeunce, i.e. the number of valid elements it contains. It applies to whichever of the above buffers is non-null, if any. If both are NULL, its value must be 0.

6.78.2.4 DDS_Long FooSeq::_element_size

Size of data element in the sequence.

Each element in the sequence has this size.

6.78.2.5 void* FooSeq::_token1

Reserved for internal use.

6.78.2.6 void* FooSeq::_token2

Reserved for internal use.

6.78.2.7 DDS_UnsignedLong FooSeq::max_alloc_size

The maximum allocation for pointer elements (string,wstring) Only in debug librarires.

6.78.2.8 DDS_Char FooSeq::_flags

Reserved for internal use.

6.79 FooTypeSupport Struct Reference

<<interface>> *<<generic>>* User data type specific interface.

6.79.1 Detailed Description

<<interface>> *<<generic>>* User data type specific interface.

Defines the user data type specific interface generated for each application class.

The concrete user data type automatically generated by the implementation is an incarnation of this class.

6.80 NDDS_Discovery_Property Struct Reference

Specifies internal information used to store discovery information about a DDS_DomainParticipant in the registry.

6.80.1 Detailed Description

Specifies internal information used to store discovery information about a DDS_DomainParticipant in the registry.

6.81 NDDS_Type_Plugin Struct Reference

<<interface>> Type plugin interface.

Data Fields

- CDR_Stream_SerializeFunction serialize_data

  Function to handle serialization of user data.
- CDR_Stream_DeserializeFunction deserialize_data

  Function to handle deserialization of user data.
- NDDS_Type_GetSerializedSizeFunc get_serialized_sample_max_size

  Function to get the maximum size of the serialized sample.
- NDDS_Type_GetSerializedSizeFunc get_serialized_key_max_size

  Function to determine the maximum serialized size of the data-type's key.
- NDDS_Type_CreateSampleFunc create_sample

  Function to create a sample of user data.
- NDDS_Type_DeleteSampleFunc delete_sample

  Function to delete a sample of user data.
- NDDS_Type_CopySampleFunc copy_sample

  Function to copy a sample of user data.
- NDDS_PluginHelper_GetKeyKindFunc get_key_kind

  Function to determine if user data is keyed.
- NDDS_PluginHelper_InstanceToKeyHashFunc instance_to_keyhash

  Function to create a 16-byte keyhash from an instance of data.

6.81.1 Detailed Description

<<interface>> Type plugin interface.

6.81.2 Field Documentation

6.81.2.1 CDR_Stream_SerializeFunction NDDS_Type_Plugin::serialize_data

Function to handle serialization of user data.

6.81.2.2 CDR_Stream_DeserializeFunction NDDS_Type_Plugin::deserialize_data

Function to handle deserialization of user data.

6.81.2.3 NDDS_Type_GetSerializedSizeFunc NDDS_Type_Plugin::get_serialized_sample_max_size

Function to get the maximum size of the serialized sample.

6.81.2.4 NDDS_Type_GetSerializedSizeFunc NDDS_Type_Plugin::get_serialized_key_max_size

Function to determine the maximum serialized size of the data-type's key.

6.81.2.5 NDDS_Type_CreateSampleFunc NDDS_Type_Plugin::create_sample

Function to create a sample of user data.

6.81.2.6 NDDS_Type_DeleteSampleFunc NDDS_Type_Plugin::delete_sample

Function to delete a sample of user data.

6.81.2.7 NDDS_Type_CopySampleFunc NDDS_Type_Plugin::copy_sample

Function to copy a sample of user data.

6.81.2.8 NDDS_PluginHelper_GetKeyKindFunc NDDS_Type_Plugin::get_key_kind

Function to determine if user data is keyed.

6.81.2.9 NDDS_PluginHelper_InstanceToKeyHashFunc NDDS_Type_Plugin::instance_to_keyhash

Function to create a 16-byte keyhash from an instance of data.

6.82 NDDS_Type_PluginVersion Struct Reference

Data type used to keep track of versioning information about a NDDS_Type_Plugin.

Data Fields

- char majorRev

  Major version number.
- char minorRev

  Minor version number.

6.82.1 Detailed Description

Data type used to keep track of versioning information about a NDDS_Type_Plugin.

6.82.2 Field Documentation

6.82.2.1 char NDDS_Type_PluginVersion::majorRev

Major version number.

6.82.2.2 char NDDS_Type_PluginVersion::minorRev

Minor version number.

6.83 NETIO_Address Struct Reference

Data Fields

- RTI_INT32 kind

  The address kind.
- RTI_UINT32 port

  The address port.
- union NETIO_AddressValue value

  The transport address.

6.83.1 Detailed Description

<<*eXtension*>> <<*cert*>> The NETIO Address.

6.83.2 Field Documentation

6.83.2.1 RTI_INT32 NETIO_Address::kind

The address kind.

6.83.2.2 RTI_UINT32 NETIO_Address::port

The address port.

6.83.2.3 union NETIO_AddressValue NETIO_Address::value

The transport address.

6.84 NETIO_AddressSeq Struct Reference

<<*eXtension*>> <<*cert*>> The NETIO_AddressSeq sequence type.

6.84.1 Detailed Description

<<*eXtension*>> <<*cert*>> The NETIO_AddressSeq sequence type.

6.85 NETIO_AddressUInt32 Struct Reference

<<*eXtension*>> <<*cert*>> NETIO_AddressUInt32

Data Fields

- RTI_UINT32 value [4]

  An array of unsigned 32-bit integers.

6.85.1 Detailed Description

<<*eXtension*>> <<*cert*>> NETIO_AddressUInt32

6.85.2 Field Documentation

6.85.2.1 RTI_UINT32 NETIO_AddressUInt32::value[4]

An array of unsigned 32-bit integers.

6.86 NETIO_AddressValue Union Reference

<<eXtension>> <<cert>> NETIO_AddresValue

Data Fields

- struct NETIO_AddressIpv6 address

  Address as array of octets address.octet[0-15].
- struct NETIO_AddressUInt32 as_uint32

  Address as array of unsigned 32-bit integers.

6.86.1 Detailed Description

<<eXtension>> <<cert>> NETIO_AddresValue

6.86.2 Field Documentation

6.86.2.1 struct NETIO_AddressIpv6 NETIO_AddressValue::address

Address as array of octets address.octet[0-15].

6.86.2.2 struct NETIO_AddressUInt32 NETIO_AddressValue::as_uint32

Address as array of unsigned 32-bit integers.

6.87 NETIO_DGRAM_InterfaceI Struct Reference

<<eXtension>> <<cert>>.

Data Fields

- NETIO_DGRAM_Interface_createFunc create_instance

  Creates an instance of the user defined datagram Interface.
- NETIO_DGRAMUserInterface_get_interface_listFunc get_interface_list

  Read available interfaces from the user DGRAM interface.
- NETIO_Interface_release_addressFunc release_address

  <<eXtension>> <<cert>> The definition of the NETIO_InterfaceI release_address method.
- NETIO_DGRAMUserInterface_resolve_addressFunc resolve_address

  <<eXtension>> <<cert>> Resolve an IPv4 address string to NETIO_Address if possible.
- NETIO_Interface_sendFunc send
- NETIO_Interface_get_route_tableFunc get_route_table

  <<eXtension>> <<cert>> The definition of the::NETIO_InterfaceI get_route_table method.
- NETIO_DGRAMUserInterface_bind_addressFunc bind_address

  Bind to an address.

6.87.1 Detailed Description

<<eXtension>> *<<cert>>*.

6.87.2 Field Documentation

6.87.2.1 NETIO_DGRAM_Interface_createFunc NETIO_DGRAM_InterfaceI::create_instance

Creates an instance of the user defined datagram Interface.

A user defined datagram function that is called by RTI Connext Micro to create an instance of the NETIO_DGRAM transport.

Parameters

upstream	*<<in>>* The upstream interface.
property	*<<in>>* The user defined interface property.

Returns

The function must return a pointer to a new instance on success and NULL on failure.

6.87.2.2 NETIO_DGRAMUserInterface_get_interface_listFunc NETIO_DGRAM_InterfaceI::get_interface_list

Read available interfaces from the user DGRAM interface.

A user defined datagram function that is called by RTI Connext Micro to read the available interfaces from a transport.

Parameters

netio_intf	*<<in>>* The DGRAM interface to use.
if_table	*<<in>>* The interface table.

Returns

TRUE on success, FALSE on failure.

6.87.2.3 NETIO_Interface_release_addressFunc NETIO_DGRAM_InterfaceI::release_address

<<eXtension>> *<<cert>>* The definition of the NETIO_InterfaceI release_address method.

Parameters

self	*<<in>>* The NETIO interface to release addresses on.
src_addr	*<<in>>* The address to release.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

6.87.2.4 NETIO_DGRAMUserInterface_bind_addressFunc NETIO_DGRAM_InterfaceI::bind_address

Bind to an address.

A user defined datagram function called by RTI Connext Micro to start listening on the specified address.

Parameters

user_intf	*<<in>>* The DGRAM interface to to use.
source	*<<in>>* The address to bind.

Returns

RTI_TRUE if successful, RTI_FALSE if not.

6.88 NETIO_DGRAM_InterfaceMultiCastGroup Struct Reference

<<eXtension>> <<cert>>.

Data Fields

- struct NETIO_Address address_low

  The lowest multicast address.
- struct NETIO_Address address_high

  The highest multicast address.
- struct NETIO_Netmask netmask

  The netmask to apply to the group.

6.88.1 Detailed Description

<<eXtension>> <<cert>>.

6.88.2 Field Documentation

6.88.2.1 struct NETIO_Address NETIO_DGRAM_InterfaceMultiCastGroup::address_low

The lowest multicast address.

6.88.2.2 struct NETIO_Address NETIO_DGRAM_InterfaceMultiCastGroup::address_high

The highest multicast address.

6.88.2.3 struct NETIO_Netmask NETIO_DGRAM_InterfaceMultiCastGroup::netmask

The netmask to apply to the group.

The netmask is a sequence of N consecutive 1s, starting with the left most bit. In order to determine if an address is a multicast address, the netmask is first applied to the address and the high and low address of the multicast group. If the masked address is within the multicast group, it is treated as a multicast address.

6.89 NETIO_DGRAM_InterfaceRouteEntry Struct Reference

<<eXtension>> <<cert>>.

Data Fields

- struct NETIO_Address address

  The address of the entry.
- struct NETIO_Netmask netmask

  The netmask of the entry used for comparing addresses.

6.89.1 Detailed Description

<<eXtension>> <<cert>>.

6.89.2 Field Documentation

6.89.2.1 struct NETIO_Address NETIO_DGRAM_InterfaceRouteEntry::address

The address of the entry.

The NETIO_DGRAM_InterfaceRouteEntry structure provides a convenient method to add routes to a NETIO_DGRAM transport. The address attribute is the addresses that can be reached after the netmask attribute has been applied.

6.89.2.2 struct NETIO_Netmask NETIO_DGRAM_InterfaceRouteEntry::netmask

The netmask of the entry used for comparing addresses.

6.90 NETIO_DGRAM_InterfaceTableEntry Struct Reference

<<eXtension>> <<cert>>.

Data Fields

- RTI_UINT32 flags

  The flags attribute is a bitmap used to indicate capabilities of the transport.
- RTI_UINT32 mtu

  The maximum message size that can be sent and received by the transport.
- struct NETIO_Address address

  The address of the interface.
- const char * ifname

  The interface name.

6.90.1 Detailed Description

<<eXtension>> <<cert>>.

6.90.2 Field Documentation

6.90.2.1 RTI_UINT32 NETIO_DGRAM_InterfaceTableEntry::flags

The flags attribute is a bitmap used to indicate capabilities of the transport.

6.90.2.2 RTI_UINT32 NETIO_DGRAM_InterfaceTableEntry::mtu

The maximum message size that can be sent and received by the transport.

The mtu is a hint to RTI Connext Micro for what the maximum message size that can be sent and received over the transport is. This attributes should be the same of the payload portion; that is the maximum number of bytes that can be encapsulated by the transport.

Note: This attribute is currently ignored by RTI Connext Micro.

6.90.2.3 struct NETIO_Address NETIO_DGRAM_InterfaceTableEntry::address

The address of the interface.

This is the address that is announced to other DDS DomainParticipant that this transport can receive messages on.

6.90.2.4 const char* NETIO_DGRAM_InterfaceTableEntry::ifname

The interface name.

Note: This attribute is not used by RTI Connext Micro.

6.91 NETIO_DGRAM_InterfaceTableEntrySeq Struct Reference

A sequence of NETIO_DGRAMInterfaceTableEntry elements.

6.91.1 Detailed Description

A sequence of NETIO_DGRAMInterfaceTableEntry elements.

6.92 NETIO_Interface Struct Reference

Base-class definition for all NETIO interfaces.

6.92.1 Detailed Description

Base-class definition for all NETIO interfaces.

<<*eXtension*>> <<*cert*>> The NETIO_Interface base-class.

All NETIO layers are RT components and are derived from RT components. All NETIO layers also share a common base class, NETIO_Interface, and share common properties.

6.93 NETIO_InterfaceI Struct Reference

6.93.1 Detailed Description

<<*eXtension*>> <<*cert*>> NETIO_InterfaceI. This type is internal to RTI

6.94 NETIO_Netmask Struct Reference

<<*eXtension*>> <<*cert*>> NETIO_Netmask

Data Fields

- RTI_UINT32 bits

  The number of bits in the netmask.
- RTI_UINT32 mask [NETIO_NETMASK_MASK_LENGTH]

  The netmask as a sequence of consecutive 1s from left to right.

6.94.1 Detailed Description

<<*eXtension*>> <<*cert*>> NETIO_Netmask

6.94.2 Field Documentation

6.94.2.1 RTI_UINT32 NETIO_Netmask::bits

The number of bits in the netmask.

6.94.2.2 RTI_UINT32 NETIO_Netmask::mask[NETIO_NETMASK_MASK_LENGTH]

The netmask as a sequence of consecutive 1s from left to right.

6.95 NETIO_NetmaskSeq Struct Reference

<<eXtension>> <<cert>> The NETIO_NetmaskSeq sequence type.

6.95.1 Detailed Description

<<eXtension>> <<cert>> The NETIO_NetmaskSeq sequence type.

6.96 NETIO_Packet Struct Reference

Implementation of the abstract NETIO_Packet type.

6.96.1 Detailed Description

Implementation of the abstract NETIO_Packet type.

<<eXtension>> <<cert>> The NETIO Packet API.

The NETIO layers passed NETIO_Packet structures upstream/downstream. Any meta-data about the packet must be encapsulated either in the packet payload or the packet info field.

6.97 OSAPI_LogProperty Struct Reference

Configuration of logging functionality.

Data Fields

- RTI_SIZE_T max_buffer_size

  The maximum number of bytes allocated to the log buffer.
- RTI_UINT32 log_detail

  Bitmap to control the fidelity of what is stored in the log entry.
- OSAPI_Log_write_buffer_T write_buffer

  Function pointer to output a log/trace buffer.

6.97.1 Detailed Description

Configuration of logging functionality.

6.97.2 Field Documentation

6.97.2.1 RTI_SIZE_T OSAPI_LogProperty::max_buffer_size

The maximum number of bytes allocated to the log buffer.

Log entries are of variable length. When this limit has been reached, no more log entries can be stored unless the buffer is cleared.

6.97.2.2 RTI_UINT32 OSAPI_LogProperty::log_detail

Bitmap to control the fidelity of what is stored in the log entry.

6.97.2.3 OSAPI_Log_write_buffer_T OSAPI_LogProperty::write_buffer

Function pointer to output a log/trace buffer.

6.98 OSAPI_SharedMemorySegmentHandle Struct Reference

Data Fields

- struct OSAPI_SharedMemorySegmentHeader * ptr_header
- void * ptr_user_data
- OSAPI_SHMEM_MODE mode

6.98.1 Detailed Description

Opaque handle initialized by the create and attach methods and used to interact with a shared memory segment.

6.98.2 Field Documentation

6.98.2.1 struct OSAPI_SharedMemorySegmentHeader* OSAPI_SharedMemorySegmentHandle::ptr_header

Pointer to segment header in shared memory

6.98.2.2 void* OSAPI_SharedMemorySegmentHandle::ptr_user_data

Pointer to where user data can be stored in shared memory

6.98.2.3 OSAPI_SHMEM_MODE OSAPI_SharedMemorySegmentHandle::mode

Mode requested by the caller of create or attach

6.99 OSAPI_SharedMemorySegmentHeader Struct Reference

Data Fields

- RTI_UINT64 size
- OSAPI_SHMEM_MODE mode

6.99.1 Detailed Description

Opaque header at beginning of shared memory segment to share information about the segment between processes.

6.99.2 Field Documentation

6.99.2.1 RTI_UINT64 OSAPI_SharedMemorySegmentHeader::size

Size requested by the caller of create

6.99.2.2 OSAPI_SHMEM_MODE OSAPI_SharedMemorySegmentHeader::mode

Mode requested by the caller of create

6.100 OSAPI_SystemI Struct Reference

<<*cert*>> The System API.

Data Fields

- OSAPI_System_start_timer_T start_timer

  Implementation of function to start a timer.
- OSAPI_System_get_timer_resolution_T get_timer_resolution

  Implementation of function to get the timer resolution.
- OSAPI_System_get_time_T get_time

  Implementation of function to get the current time.
- OSAPI_System_generate_uuid_T generate_uuid

  Implementation of function to generate a UUID.
- OSAPI_System_get_hostname_T get_hostname

  Implementation of function to get the hostname.
- OSAPI_System_get_ticktime_T get_ticktime

  Implementation of function to get the ticktime.

6.100.1 Detailed Description

<<*cert*>> The System API.

The methods in this interface are called by the corresponding OSAPI_System function. For example, OSAPI_System_get_time() calls system -> get_time().

6.100.2 Field Documentation

6.100.2.1 OSAPI_System_start_timer_T OSAPI_SystemI::start_timer

Implementation of function to start a timer.

6.100.2.2 OSAPI_System_get_timer_resolution_T OSAPI_SystemI::get_timer_resolution

Implementation of function to get the timer resolution.

6.100.2.3 OSAPI_System_get_time_T OSAPI_SystemI::get_time

Implementation of function to get the current time.

6.100.2.4 OSAPI_System_generate_uuid_T OSAPI_SystemI::generate_uuid

Implementation of function to generate a UUID.

6.100.2.5 OSAPI_System_get_hostname_T OSAPI_SystemI::get_hostname

Implementation of function to get the hostname.

6.100.2.6 OSAPI_System_get_ticktime_T OSAPI_SystemI::get_ticktime

Implementation of function to get the ticktime.

6.101 OSAPI_SystemProperty Struct Reference

<<cert>> The system properties.

Data Fields

- char hostname [OSAPI_SYSTEM_MAX_HOSTNAME]
  The hostname for the node.

6.101.1 Detailed Description

<<cert>> The system properties.

System properties

6.101.2 Field Documentation

6.101.2.1 char OSAPI_SystemProperty::hostname[OSAPI_SYSTEM_MAX_HOSTNAME]

The hostname for the node.

RTI Connext Micro uses this name when it announces itself on the network. This name is used by other RTI products. The hostname must not exceed OSAPI_SYSTEM_MAX_HOSTNAME, including the \0 character.

6.102 OSAPI_SystemTime Struct Reference

System time.

Data Fields

- RTI_INT64 sec
  Seconds.
- RTI_UINT32 nanosec
  Nanoseconds.

6.102.1 Detailed Description

System time.

Expresses time in seconds and nanoseconds. Assumes that Nanoseconds is less than 1000000000.

6.102.2 Field Documentation

6.102.2.1 RTI_INT64 OSAPI_SystemTime::sec

Seconds.

6.102.2.2 RTI_UINT32 OSAPI_SystemTime::nanosec

Nanoseconds.

6.103 OSAPI_ThreadProperty Struct Reference

The properties for thread creation.

Data Fields

- RTI_UINT32 stack_size

  Hint on required stack-size in bytes. The special value OSAPI_THREAD_USE_OSDEFAULT_STACKSIZE indicates to not set the stack-size and rely on the OS defaults.
- RTI_INT32 priority

  Hint on required priority. A value of 0 or greater is used as is; negative values are mapped according to the platform documentation.
- OSAPI_ThreadOptions options

  Hint for thread creation to the OS.

6.103.1 Detailed Description

The properties for thread creation.

6.103.2 Field Documentation

6.103.2.1 RTI_UINT32 OSAPI_ThreadProperty::stack_size

Hint on required stack-size in bytes. The special value OSAPI_THREAD_USE_OSDEFAULT_STACKSIZE indicates to not set the stack-size and rely on the OS defaults.

6.103.2.2 RTI_INT32 OSAPI_ThreadProperty::priority

Hint on required priority. A value of 0 or greater is used as is; negative values are mapped according to the platform documentation.

6.103.2.3 OSAPI_ThreadOptions OSAPI_ThreadProperty::options

Hint for thread creation to the OS.

6.104 OSAPI_TimerProperty Struct Reference

Timer properies.

6.104.1 Detailed Description

Timer properies.

6.105 RTI_ManagementQosPolicy Struct Reference

<<cert>> Policy used determine how DDS_DataReader, DDS_DataWriter, DDS_Publisher, and DDS_Subscriber is managed.

Data Fields

- DDS_Boolean is_anonymous

  Whether the entity is an anonymous entity or not.

6.105.1 Detailed Description

<<cert>> Policy used determine how DDS_DataReader, DDS_DataWriter, DDS_Publisher, and DDS_Subscriber is managed.

6.105.2 Field Documentation

6.105.2.1 DDS_Boolean RTI_ManagementQosPolicy::is_anonymous

Whether the entity is an anonymous entity or not.

Properties:

Changeable = NO

If DDS_BOOLEAN_TRUE, the created entity is anonymous and state, such as which sequence number to send with, is not maintained. This option should only be used to create a built-in ParticipantBuiltTopicDataWriter.

[default] DDS_BOOLEAN_FALSE

6.106 RTPS_Checksum Union Reference

<<eXtension>> *<<cert>>* RTPS checksum type

Data Fields

- RTI_UINT32 checksum32

  A 32-bit unsigned checksum in host endianess order.
- RTI_UINT64 checksum64

  A 64-bit unsigned integer checksum in host endianess order.
- RTI_UINT8 checksum128 [16]

  A 128-bit checksum as a 16-byte array.

6.106.1 Detailed Description

<<eXtension>> *<<cert>>* RTPS checksum type

6.107 RTPS_ChecksumClass Struct Reference

<<eXtension>> *<<cert>>* Definition of a checksum function

Data Fields

- RTPS_ChecksumClassId_T class_id

  The checksum class ID.
- void * context

  User defined context.
- RTPS_ChecksumCalculate_T checksum_calculate

  User defined function to calculate a checksum.

6.107.1 Detailed Description

<<eXtension>> *<<cert>>* Definition of a checksum function

6.108 RTPS_ChecksumProperty Struct Reference

<<eXtension>> *<<cert>>* Checksum properties for the RTPS plugin

Data Fields

- RTPS_ChecksumClass_T builtin_checksum32_class

  The builtin 32-bit checksum class.
- RTPS_ChecksumClass_T builtin_checksum64_class

  The builtin 64-bit checksum class.
- RTPS_ChecksumClass_T builtin_checksum128_class

  The builtin 128-bit checksum class.
- RTPS_ChecksumTxMode_T checksum_tx_mode

  The transmit mode for checksums.
- RTI_BOOL allow_builtin_override

  If TRUE, allow overriding the builtin checksum functions.

6.108.1 Detailed Description

<<eXtension>> *<<cert>>* Checksum properties for the RTPS plugin

6.109 RTPS_InterfaceFactoryProperty Struct Reference

<<eXtension>> *<<cert>>* Properties for the RTPS Transport.

Data Fields

- struct RTPS_ChecksumProperty checksum

  The checksum properties for RTPS.

6.109.1 Detailed Description

<<eXtension>> *<<cert>>* Properties for the RTPS Transport.

6.110 UDP_InterfaceFactoryProperty Struct Reference

<<eXtension>> *<<cert>>* Properties for the UDP Transport.

Data Fields

- struct REDA_StringSeq allow_interface

  Sequence of allowed interface names.
- struct REDA_StringSeq deny_interface

  Sequence of denied interface names. This list is checked after the allow_interface list.
- RTI_INT32 max_send_buffer_size

The size of the send socket buffer.

- RTI_INT32 max_receive_buffer_size

  The size of the receive socket buffer.
- RTI_INT32 max_message_size

  The maximum size of the message which can be sent or received.
- RTI_INT32 max_send_message_size

  The maximum size of the message which can be sent. This is only a hint and not enforced.
- RTI_INT32 multicast_ttl

  The maximum TTL.
- struct UDP_InterfaceTableEntrySeq if_table

  The interface table if interfaces are added manually.
- REDA_String_T multicast_interface

  The network interface to use to send to multicast.
- RTI_BOOL is_default_interface

  If this should be considered the default UDP interfaces if no other UDP interface is found to handle a route.
- struct OSAPI_ThreadProperty recv_thread

  Thread properties for each receive thread created by this NETIO interface.

6.110.1 Detailed Description

<<eXtension>> <<cert>> Properties for the UDP Transport.

6.110.2 Field Documentation

6.110.2.1 struct REDA_StringSeq UDP_InterfaceFactoryProperty::allow_interface

Sequence of allowed interface names.

6.110.2.2 struct REDA_StringSeq UDP_InterfaceFactoryProperty::deny_interface

Sequence of denied interface names. This list is checked after the allow_interface list.

6.110.2.3 RTI_INT32 UDP_InterfaceFactoryProperty::max_send_buffer_size

The size of the send socket buffer.

6.110.2.4 RTI_INT32 UDP_InterfaceFactoryProperty::max_receive_buffer_size

The size of the receive socket buffer.

6.110.2.5 RTI_INT32 UDP_InterfaceFactoryProperty::max_message_size

The maximum size of the message which can be sent or received.

The UDP_InterfaceFactoryProperty::max_message_size resource limit limits the size of the maximum message that can be received.

RTI Connext Micro does not support RTPS fragmentation. Thus, this resource-limit must be at least large enough to receive the largest serialized data-type plus overhead. A reasonable rule-of-thumb for overhead is 256 bytes per sample.

The UDP transport allocated on buffer for each resource in DDS_DomainParticipantResourceLimitsQosPolicy::max_receive_ports. Thus, the total amount of memory is (max_message_size $*$ max_receive_ports)

6.110.2.6 RTI_INT32 UDP_InterfaceFactoryProperty::max_send_message_size

The maximum size of the message which can be sent. This is only a hint and not enforced.

This parameters is useful to UDP transformation functions which must allocate a buffer to put the result in.

6.110.2.7 RTI_INT32 UDP_InterfaceFactoryProperty::multicast_ttl

The maximum TTL.

6.110.2.8 struct UDP_InterfaceTableEntrySeq UDP_InterfaceFactoryProperty::if_table

The interface table if interfaces are added manually.

6.110.2.9 REDA_String_T UDP_InterfaceFactoryProperty::multicast_interface

The network interface to use to send to multicast.

6.110.2.10 RTI_BOOL UDP_InterfaceFactoryProperty::is_default_interface

If this should be considered the default UDP interfaces if no other UDP interface is found to handle a route.

6.110.2.11 struct OSAPI_ThreadProperty UDP_InterfaceFactoryProperty::recv_thread

Thread properties for each receive thread created by this NETIO interface.

6.111 UDP_InterfaceTableEntry Struct Reference

Generic structure to describe a network interface.

Data Fields

- RTI_UINT32 flags

  Flags to indicate if the interface is up etc.
- RTI_UINT32 address

  The address of the interface as configured by the OS.
- RTI_UINT32 netmask

  The netmask of the interface as configured by the OS.
- char ifname [UDP_INTERFACE_MAX_IFNAME]

  The name of the interface as configured by the OS.

6.111.1 Detailed Description

Generic structure to describe a network interface.

6.111.2 Field Documentation

6.111.2.1 RTI_UINT32 UDP_InterfaceTableEntry::flags

Flags to indicate if the interface is up etc.

6.111.2.2 RTI_UINT32 UDP_InterfaceTableEntry::address

The address of the interface as configured by the OS.

6.111.2.3 RTI_UINT32 UDP_InterfaceTableEntry::netmask

The netmask of the interface as configured by the OS.

6.111.2.4 char UDP_InterfaceTableEntry::ifname[UDP_INTERFACE_MAX_IFNAME]

The name of the interface as configured by the OS.

6.112 UDP_InterfaceTableEntrySeq Struct Reference

<<*eXtension*>> Declares IDL sequence<UDP_InterfaceTableEntry>

6.112.1 Detailed Description

<<*eXtension*>> Declares IDL sequence<UDP_InterfaceTableEntry>

6.113 ZCOPY_NotifInterfaceFactoryProperty Struct Reference

Notification interface factory specific property.

Data Fields

- RTI_UINT32 max_samples_per_notif

  Maximum samples to process per notification per DR.
- struct ZCOPY_NotifUserInterfaceI * user_intf

  The downstream transport interface.
- void * user_property

  Pass through pointer to the downstream transport. Must be valid for the life-cycle of the transport.

6.113.1 Detailed Description

Notification interface factory specific property.

6.113.2 Field Documentation

6.113.2.1 RTI_UINT32 ZCOPY_NotifInterfaceFactoryProperty::max_samples_per_notif

Maximum samples to process per notification per DR.

6.113.2.2 struct ZCOPY_NotifUserInterfaceI* ZCOPY_NotifInterfaceFactoryProperty::user_intf

The downstream transport interface.

6.113.2.3 void* ZCOPY_NotifInterfaceFactoryProperty::user_property

Pass through pointer to the downstream transport. Must be valid for the life-cycle of the transport.

6.114 ZCOPY_NotifUserInterfaceI Struct Reference

Notification mechanism user interface.

Data Fields

- ZCOPY_NotifUserInterface_createFunc create_instance

  Create an instance of the user interface.
- NETIO_Interface_resolve_addressFunc resolve_address

  Resolve an address.
- NETIO_Interface_get_route_tableFunc get_route_table

  Get the route table.

- ZCOPY_NotifUserInterface_reserve_addressFunc reserve_address

  Reserve an address on the user interface.
- ZCOPY_NotifUserInterface_release_addressFunc release_address

  Release an address on the user interface.
- ZCOPY_NotifUserInterface_add_routeFunc add_route

  Add a route on the user interface.
- ZCOPY_NotifUserInterface_delete_routeFunc delete_route

  Delete a route on the user interface.
- ZCOPY_NotifUserInterface_bindFunc bind

  Bind on the user interface.
- ZCOPY_NotifUserInterface_unbindFunc unbind

  Unbind on the user interface.
- ZCOPY_NotifUserInterface_sendFunc send

  Send a notification using the user interface.
- ZCOPY_NotifUserInterface_notify_portFunc notify_recv_port

  Notify the receive port.

6.114.1 Detailed Description

Notification mechanism user interface.

6.114.2 Field Documentation

6.114.2.1 ZCOPY_NotifUserInterface_createFunc ZCOPY_NotifUserInterface::create_instance

Create an instance of the user interface.

6.114.2.2 NETIO_Interface_resolve_addressFunc ZCOPY_NotifUserInterface::resolve_address

Resolve an address.

6.114.2.3 NETIO_Interface_get_route_tableFunc ZCOPY_NotifUserInterface::get_route_table

Get the route table.

6.114.2.4 ZCOPY_NotifUserInterface_reserve_addressFunc ZCOPY_NotifUserInterface::reserve_address

Reserve an address on the user interface.

6.114.2.5 ZCOPY_NotifUserInterface_release_addressFunc ZCOPY_NotifUserInterface::release_address

Release an address on the user interface.

6.114.2.6 ZCOPY_NotifUserInterface_add_routeFunc ZCOPY_NotifUserInterface::add_route

Add a route on the user interface.

6.114.2.7 ZCOPY_NotifUserInterface_delete_routeFunc ZCOPY_NotifUserInterface::delete_route

Delete a route on the user interface.

6.114.2.8 ZCOPY_NotifUserInterface_bindFunc ZCOPY_NotifUserInterface::bind

Bind on the user interface.

6.114.2.9 ZCOPY_NotifUserInterface_unbindFunc ZCOPY_NotifUserInterfaceI::unbind

Unbind on the user interface.

6.114.2.10 ZCOPY_NotifUserInterface_sendFunc ZCOPY_NotifUserInterfaceI::send

Send a notification using the user interface.

6.114.2.11 ZCOPY_NotifUserInterface_notify_portFunc ZCOPY_NotifUserInterfaceI::notify_recv_port

Notify the receive port.

7 File Documentation

7.1 cdr_log.h File Reference

Macros

- #define CDR_LOG_STREAM_ALLOC_EC (CDR_LOG_BASE + 1)
  Failed to allocate stream or stream buffer.
- #define CDR_LOG_SET_OFFSET_EC (CDR_LOG_BASE + 2)
  Failed to set stream's current offset.
- #define CDR_LOG_INCR_OFFSET_EC (CDR_LOG_BASE + 3)
  Failed to increament stream's current offset.
- #define CDR_LOG_UNSUPPORTED_OPTIONS_EC (CDR_LOG_BASE + 5)
  Unsupported CDR encapsulations were received. The sample is dropped.

7.1.1 Detailed Description

7.2 db_dll.h File Reference

External linkage definitions for DB API.

7.2.1 Detailed Description

External linkage definitions for DB API.

7.3 db_log.h File Reference

DB module log codes.

Macros

- #define DB_LOG_SORTED_ALLOC_EC (DB_LOG_BASE + 1)
  Not sufficient memory to allocate sorted list for an index.
- #define DB_LOG_NAME_TOO_LONG_EC (DB_LOG_BASE + 2)
  The specified database name is too long.
- #define DB_LOG_ILLEGAL_TABLE_SIZE_EC (DB_LOG_BASE + 3)
  Illegal table size specified.
- #define DB_LOG_ILLEGAL_LOCK_MODE_EC (DB_LOG_BASE + 4)
  Illegal combination of lock mode and mutex given.

- #define DB_LOG_MUTEX_ALLOC_EC (DB_LOG_BASE + 5)

  Failed to allocate database mutex.
- #define DB_LOG_ALLOC_TABLE_POOL_EC (DB_LOG_BASE + 6)

  Failed to allocate buffer pool.
- #define DB_LOG_TABLES_INUSE_EC (DB_LOG_BASE + 7)

  Table is in use.
- #define DB_LOG_TABLE_NAME_TOO_LONG_EC (DB_LOG_BASE + 8)

  Specified table name too long.
- #define DB_LOG_ILLEGAL_RECORD_COUNT_EC (DB_LOG_BASE + 9)

  Illegal record count specified.
- #define DB_LOG_TABLE_EXISTS_EC (DB_LOG_BASE + 10)

  Specified table already exists.
- #define DB_LOG_OUT_OF_TABLES_EC (DB_LOG_BASE + 11)

  Table resources exceeded.
- #define DB_LOG_OUT_OF_RECORDS_EC (DB_LOG_BASE + 12)

  Table record resources exceeded.
- #define DB_LOG_OUT_OF_INDICES_EC (DB_LOG_BASE + 13)

  Table index resources exceeded.
- #define DB_LOG_OUT_OF_CURSORS_EC (DB_LOG_BASE + 14)

  Table cursor resources exceeded.
- #define DB_LOG_RECORDS_INUSE_EC (DB_LOG_BASE + 15)

  Cannot delete table, records are still in table.
- #define DB_LOG_CURSORS_INUSE_EC (DB_LOG_BASE + 16)

  Cannot delete table, cursors are still in use.
- #define DB_LOG_INDEX_INUSE_EC (DB_LOG_BASE + 17)

  Cannot delete table, indices are still in use.
- #define DB_LOG_ALLOC_CURSOR_POOL_EC (DB_LOG_BASE + 18)

  Failed to allocate cursor buffer pool.
- #define DB_LOG_ALLOC_INDEX_POOL_EC (DB_LOG_BASE + 19)

  Failed to allocate index buffer pool.
- #define DB_LOG_ALLOC_RECORD_POOL_EC (DB_LOG_BASE + 20)

  Failed to allocate record buffer pool.
- #define DB_LOG_ALLOC_DATABASE_EC (DB_LOG_BASE + 21)

  Failed to allocate database.
- #define DB_LOG_TABLE_NOT_INUSE_EC (DB_LOG_BASE + 22)

  An attempt was made to delete a table not in use.
- #define DB_LOG_RECORD_ALREADY_EXISTS_EC (DB_LOG_BASE + 23)

  An attempt was made to insert an already existing record.
- #define DB_LOG_RECORD_DOES_NOT_EXIST_EC (DB_LOG_BASE + 24)

  An attempt was made to delete a record that does not exist.
- #define DB_LOG_CURSOR_INVALIDATED_EC (DB_LOG_BASE + 25)

  An attempt was made to use a cursor that has been invalidated.
- #define DB_LOG_LOCK_FAILURE_EC (DB_LOG_BASE + 26)

  Failed to lock the database.
- #define DB_LOG_UNLOCK_FAILURE_EC (DB_LOG_BASE + 27)

  Failed to unlock the database.

7.3.1 Detailed Description

DB module log codes. This file defines log codes for all messages logged by the database modules. All the log codes in this file are publicly documented.

7.4 dds_c_common.h File Reference

DDS_C common definitions.

Macros

- #define DDS_BOOLEAN_TRUE

  Defines TRUE value of IDL boolean data type.
- #define DDS_BOOLEAN_FALSE

  Defines FALSE value of IDL boolean data type.

Typedefs

- typedef CDR_Char DDS_Char

  Defines IDL char data type.
- typedef CDR_Wchar DDS_Wchar

  Defines IDL wchar data type.
- typedef CDR_Octet DDS_Octet

  Defines IDL octet data type.
- typedef CDR_Short DDS_Short

  Defines IDL short data type.
- typedef CDR_UnsignedShort DDS_UnsignedShort

  Defines IDL unsigned short data type.
- typedef CDR_Long DDS_Long

  Defines IDL long data type.
- typedef CDR_UnsignedLong DDS_UnsignedLong

  Defines IDL unsigned long data type.
- typedef CDR_LongLong DDS_LongLong

  Defines IDL long long data type.
- typedef CDR_UnsignedLongLong DDS_UnsignedLongLong

  Defines IDL unsigned long long data type.
- typedef CDR_Float DDS_Float

  Defines IDL float data type.
- typedef CDR_Double DDS_Double

  Defines IDL double data type.
- typedef CDR_LongDouble DDS_LongDouble

  Defines IDL long double data type.
- typedef CDR_Boolean DDS_Boolean

  Defines IDL boolean data type.
- typedef CDR_Enum DDS_Enum

  Defines IDL enum data type.
- typedef CDR_String DDS_String

  IDL `string` type representation.
- typedef CDR_Wstring DDS_Wstring

  Defines IDL wstring data type.

7.4.1 Detailed Description

DDS_C common definitions.

7.5 dds_c_common_impl.h File Reference

DDS_C common definitions.

7.5.1 Detailed Description

DDS_C common definitions.

7.6 dds_c_config.h File Reference

DDS C configuration file.

7.6.1 Detailed Description

DDS C configuration file.

7.7 dds_c_discovery.h File Reference

DDS discovery data definitions.

Data Structures

- struct DDS_ParticipantBuiltinTopicData

  <<cert>> *Object representing a remote DomainParticipant.*
- struct DDS_PublicationBuiltinTopicData

  Object describing a remote Publication.
- struct DDS_SubscriptionBuiltinTopicData

  Entry created when a DDS_DataReader is discovered in association with its Subscriber.

Functions

- DDS_Boolean DDS_ParticipantBuiltinTopicData_initialize (struct DDS_ParticipantBuiltinTopicData *self)

  <<cert>> *Initialize a DDS_ParticipantBuiltinTopicData object.*
- DDS_Boolean DDS_ParticipantBuiltinTopicData_copy (struct DDS_ParticipantBuiltinTopicData *out, const struct DDS_ParticipantBuiltinTopicData *in)

  <<cert>> *Copy a DDS_ParticipantBuiltinTopicData object.*
- DDS_Boolean DDS_ParticipantBuiltinTopicData_is_equal (const struct DDS_ParticipantBuiltinTopicData *left, const struct DDS_ParticipantBuiltinTopicData *right)

  <<cert>> *Compare two DDS_ParticipantBuiltinTopicData objects for equality.*
- DDS_Boolean DDS_PublicationBuiltinTopicData_initialize (struct DDS_PublicationBuiltinTopicData *self)

  <<cert>> *Initialize a DDS_PublicationBuiltinTopicData object.*
- DDS_Boolean DDS_PublicationBuiltinTopicData_copy (struct DDS_PublicationBuiltinTopicData *out, const struct DDS_PublicationBuiltinTopicData *in)

  <<cert>> *Copy a DDS_PublicationBuiltinTopicData object.*
- DDS_Boolean DDS_PublicationBuiltinTopicData_is_equal (const struct DDS_PublicationBuiltinTopicData *left, const struct DDS_PublicationBuiltinTopicData *right)

  <<cert>> *Compare two DDS_PublicationBuiltinTopicData objects for equality*
- DDS_Boolean DDS_SubscriptionBuiltinTopicData_initialize (struct DDS_SubscriptionBuiltinTopicData *self)

  <<cert>> *Initialize a DDS_SubscriptionBuiltinTopicData object.*

- DDS_Boolean DDS_SubscriptionBuiltinTopicData_copy (struct DDS_SubscriptionBuiltinTopicData *out, const struct DDS_SubscriptionBuiltinTopicData *in)

  <<cert>> Copy a DDS_SubscriptionBuiltinTopicData object.
- DDS_Boolean DDS_SubscriptionBuiltinTopicData_is_equal (const struct DDS_SubscriptionBuiltinTopicData *left, const struct DDS_SubscriptionBuiltinTopicData *right)

  <<cert>> Compare two DDS_SubscriptionBuiltinTopicData objects for equality

7.7.1 Detailed Description

DDS discovery data definitions.

7.8 dds_c_discovery_plugin.h File Reference

DDS discovery plugin interface.

Macros

- #define DDS_PARTICIPANT_TOPIC_NAME DDS_PARTICIPANT_BUILTIN_TOPIC_NAME

  <<cert>> Participant topic name
- #define DDS_PUBLICATION_TOPIC_NAME DDS_PUBLICATION_BUILTIN_TOPIC_NAME

  Publication topic name.
- #define DDS_SUBSCRIPTION_TOPIC_NAME DDS_SUBSCRIPTION_BUILTIN_TOPIC_NAME

  Subscription topic name.

7.8.1 Detailed Description

DDS discovery plugin interface.

7.9 dds_c_dll.h File Reference

DDS_C external linkage definitions.

7.9.1 Detailed Description

DDS_C external linkage definitions.

7.10 dds_c_domain.h File Reference

DDS Domain Module definitions.

Data Structures

- struct NDDS_Discovery_Property

  Specifies internal information used to store discovery information about a DDS_DomainParticipant in the registry.
- struct DDS_DiscoveryComponent

  Specifies the discovery plugin the DDS_DomainParticipant should use.
- struct DDS_DiscoveryQosPolicy

  <<cert>> Specifies the attributes required to discover participants in the domain.
- struct DDS_UserTrafficQosPolicy

A QosPolicy to configure properties of network traffic caused by the exchange of user data.

- struct DDS_DomainParticipantListener

  <<interface>> <<cert>> Listener for participant status.
- struct DDS_DomainParticipantQos

  <<cert>> QoS policies supported by a DDS_DomainParticipant entity.
- struct DDS_DomainParticipantFactoryQos

  <<cert>> QoS policies supported by a DDS_DomainParticipantFactory.

Macros

- #define DDS_DiscoveryComponent_INITIALIZER

  Initializer for new DDS_DiscoveryComponent instances.
- #define DDS_DomainParticipantListener_INITIALIZER

  <<cert>> Initializer for new DDS_DomainParticipantListener.
- #define DDS_DomainParticipantQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_DomainParticipantFactoryQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_TheParticipantFactory DDS_DomainParticipantFactory_get_instance()

  <<cert>> Alias for singleton participant factory.

Typedefs

- typedef DDS_DOMAINID_TYPE_NATIVE DDS_DomainId_t

  <<cert>> An integer that indicates in which domain a DDS_DomainParticipant communicates.
- typedef struct DDS_DomainParticipantFactoryImpl DDS_DomainParticipantFactory

  <<singleton>> <<interface>> <<cert>> Allows creation and destruction of DDS_DomainParticipant objects.

Functions

- DDS_ReturnCode_t DDS_DomainParticipantQos_initialize (struct DDS_DomainParticipantQos *self)

  <<cert>> Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_DomainParticipantQos_copy (struct DDS_DomainParticipantQos *self, const struct DDS_DomainParticipantQos *source)

  <<cert>> Copy the contents of the given QoS into this QoS. The destination structure must be preallocated and initialized.
- DDS_Boolean DDS_DomainParticipantQos_is_equal (const struct DDS_DomainParticipantQos *left, const struct DDS_DomainParticipantQos *right)

  <<cert>> Compare two DDS_DomainParticipantQos policies for equality.
- DDS_ReturnCode_t DDS_DomainParticipantFactoryQos_initialize (struct DDS_DomainParticipantFactoryQos *self)

  <<cert>> Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_DomainParticipantFactoryQos_copy (struct DDS_DomainParticipantFactoryQos *self, const struct DDS_DomainParticipantFactoryQos *source)

  <<cert>> Copy the contents of the given QoS into this QoS. The destination structure must be preallocated and initialized.
- DDS_Boolean DDS_DomainParticipantFactoryQos_is_equal (const struct DDS_DomainParticipantFactoryQos *left, const struct DDS_DomainParticipantFactoryQos *right)

  <<cert>> Compare two DDS_DomainParticipantFactoryQos policies for equality.
- DDS_Entity * DDS_DomainParticipant_as_entity (DDS_DomainParticipant *self)

<<cert>> Access a DDS_DomainParticipant's supertype instance.

- DDS_Publisher * DDS_DomainParticipant_create_publisher (DDS_DomainParticipant *self, const struct DDS_PublisherQos *qos, const struct DDS_PublisherListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_Publisher with the desired QoS policies and attaches to it the specified DDS_PublisherListener.
- DDS_Subscriber * DDS_DomainParticipant_create_subscriber (DDS_DomainParticipant *self, const struct DDS_SubscriberQos *qos, const struct DDS_SubscriberListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_Subscriber with the desired QoS policies and attaches to it the specified DDS_SubscriberListener.
- DDS_Topic * DDS_DomainParticipant_create_topic (DDS_DomainParticipant *self, const char *topic_name, const char *type_name, const struct DDS_TopicQos *qos, const struct DDS_TopicListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_Topic with the desired QoS policies
- DDS_TopicDescription * DDS_DomainParticipant_lookup_topicdescription (DDS_DomainParticipant *self, const char *topic_name)

  <<cert>> Lookup an existing locally-created DDS_TopicDescription, based on its name.
- DDS_ReturnCode_t DDS_DomainParticipant_add_peer (DDS_DomainParticipant *self, const char *peer)

  <<eXtension>> <<cert>> Attempt to contact one or more additional peer participants.
- DDS_DomainId_t DDS_DomainParticipant_get_domain_id (DDS_DomainParticipant *self)

  <<cert>> Get the unique domain identifier.
- DDS_ReturnCode_t DDS_DomainParticipant_get_current_time (DDS_DomainParticipant *self, struct DDS_Time_t *current_time)

  <<cert>> Returns the current value of the time.
- DDS_ReturnCode_t DDS_DomainParticipant_register_type (DDS_DomainParticipant *self, const char *type_name, struct NDDS_Type_Plugin *plugin)

  <<eXtension>> <<cert>> Allows an application to communicate to RTI Connext Micro the existence of a data type.
- DDS_DomainParticipantFactory * DDS_DomainParticipantFactory_get_instance (void)

  <<cert>> Gets the singleton instance of this class.
- DDS_DomainParticipant * DDS_DomainParticipantFactory_create_participant (DDS_DomainParticipantFactory *self, DDS_DomainId_t domainId, const struct DDS_DomainParticipantQos *qos, const struct DDS_DomainParticipantListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a new DDS_DomainParticipant object.
- DDS_DomainParticipant * DDS_DomainParticipantFactory_lookup_participant (DDS_DomainParticipantFactory *self, DDS_DomainId_t domainId)

  <<cert>> Locates an existing DDS_DomainParticipant.
- DDS_ReturnCode_t DDS_DomainParticipantFactory_get_qos (DDS_DomainParticipantFactory *self, struct DDS_DomainParticipantFactoryQos *qos)

  <<cert>> Gets the value for participant factory QoS.
- DDS_ReturnCode_t DDS_DomainParticipantFactory_set_qos (DDS_DomainParticipantFactory *self, const struct DDS_DomainParticipantFactoryQos *qos)

  <<cert>> Sets the value for a participant factory QoS.
- RT_Registry_T * DDS_DomainParticipantFactory_get_registry (DDS_DomainParticipantFactory *self)

  <<cert>> Retrieves the singleton registry instance associated with the factory.

Variables

- const char *const DDS_DISCOVERY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DiscoveryQosPolicy.
- struct DDS_TopicQos DDS_TOPIC_QOS_DEFAULT

  <<cert>> Special value for creating a DDS_Topic with default QoS. Note that this is the only supported parameter in RTI Connext Micro when creating a topic.
- struct DDS_PublisherQos DDS_PUBLISHER_QOS_DEFAULT

<<cert>> *Special value for creating a DDS_Publisher with default QoS.*

- struct DDS_SubscriberQos DDS_SUBSCRIBER_QOS_DEFAULT

  <<cert>> *Special value for creating a DDS_Subscriber with default QoS.*

- struct DDS_DomainParticipantFactoryQos DDS_PARTICIPANT_FACTORY_QOS_DEFAULT

  <<cert>> *Special value for creating domain participant with default QoS.*

- struct DDS_DomainParticipantQos DDS_PARTICIPANT_QOS_DEFAULT

  <<cert>> *Special value for creating domain participant with default QoS.*

7.10.1 Detailed Description

DDS Domain Module definitions.

7.10.2 Macro Definition Documentation

7.10.2.1 #define DDS_DiscoveryComponent_INITIALIZER

Value:

```
{\
  RT_ComponentFactoryId_INITIALIZER,\
  NDDS_Discovery_Property_INITIALIZER\
}
```

Initializer for new DDS_DiscoveryComponent instances.

7.11 dds_c_infrastructure.h File Reference

DDS Domain Module definitions.

Data Structures

- struct DDS_Time_t

  Type for time representation.

- struct DDS_Duration_t

  Type for duration representation.

- struct DDS_GUID_t

  Type for GUID (Global Unique Identifier) representation.

- struct DDS_SequenceNumber_t

  Type for sequence number representation.

- struct DDS_DeadlineQosPolicy

  <<cert>> *Expresses the maximum duration (deadline) within which an instance is expected to be updated.*

- struct DDS_OwnershipQosPolicy

  <<cert>> *Specifies whether it is allowed for multiple DDS_DataWriter (s) to write the same instance of the data and if so, how these modifications should be arbitrated.*

- struct DDS_OwnershipStrengthQosPolicy

  <<cert>> *Specifies the value of the strength used to arbitrate among multiple DDS_DataWriter objects that attempt to modify the same instance of a data type (identified by DDS_Topic + key).*

- struct DDS_LivelinessQosPolicy

  <<cert>> *Determines the mechanism and parameters used by the application to determine whether a DDS_Entity is alive.*

- struct DDS_ReliabilityQosPolicy

  <<cert>> *Indicates the level of reliability offered/requested by RTI Connext Micro.*

- struct DDS_HistoryQosPolicy

  <<cert>> Specifies the behavior of RTI Connext Micro in the case where the value of a sample changes (one or more times) before it can be successfully communicated to one or more existing subscribers.
- struct DDS_DurabilityQosPolicy

  Durability properties that affect the life-cycle of published data.
- struct DDS_ResourceLimitsQosPolicy

  <<cert>> Specifies the resources that RTI Connext Micro can consume in order to meet the requested QoS.
- struct DDS_PresentationQosPolicy

  Specifies how the samples representing changes to data instances are presented to a subscribing application.
- struct DDS_DestinationOrderQosPolicy

  Destination order policies that affect the ordering of subsribed data.
- struct DDS_DataReaderResourceLimitsQosPolicy

  <<cert>> Resource limits that apply only to DDS_DataReader instances.
- struct DDS_DataWriterResourceLimitsQosPolicy

  <<cert>> Resource limits that apply only to DDS_DataWriter instances.
- struct DDS_EntityFactoryQosPolicy

  <<cert>> A QoS policy for all DDS_Entity types that can act as factories for one or more other DDS_Entity types.
- struct DDS_SystemResourceLimitsQosPolicy

  <<cert>> Resource limits that apply only to DDS_DomainParticipantFactory
- struct DDS_RtpsWellKnownPorts_t

  RTPS well-known port mapping configuration.
- struct DDS_WireProtocolQosPolicy

  <<cert>> Specifies the wire-protocol-related attributes for the DDS_DomainParticipant.
- struct DDS_ChecksumProperty

  <<eXtension>> Type to define the checksum properties of the participant RTI Connext Micro.
- struct DDS_ChecksumProperty_t

  <<eXtension>> Type to define the checksum properties of the participant RTI Connext Micro.
- struct DDS_Locator_t

  <<eXtension>> <<cert>> Type used to represent the addressing information needed to send a message to an RTPS Endpoint using one of the supported transports.
- struct DDS_LocatorSeq

  Declares IDL `sequence` *<* *DDS_Locator_t* *>*
- struct DDS_ProtocolVersion

  <<eXtension>> <<cert>> Type used to represent the version of the RTPS protocol.
- struct DDS_VendorId

  <<eXtension>> <<cert>> Type used to represent the vendor of the service implementing the RTPS protocol.
- struct DDS_VendorId_t

  <<eXtension>> <<cert>> Type used to represent the vendor of the service implementing the RTPS protocol.
- struct DDS_ProductVersion

  <<eXtension>> <<cert>> Type used to represent the current version of RTI Connext Micro.
- struct DDS_ProductVersion_t

  <<eXtension>> <<cert>> Type used to represent the current version of RTI Connext Micro.
- struct DDS_RtpsReliableReaderProtocol_t

  Qos related to reliable reader protocol defined in RTPS.
- struct DDS_DataReaderProtocolQosPolicy

  <<eXtension>> Protocol that applies only to DDS_DataReader instances.
- struct DDS_RtpsReliableWriterProtocol_t

  <<cert>> QoS related to the reliable writer protocol defined in RTPS.
- struct DDS_DataWriterProtocolQosPolicy

  <<eXtension>> <<cert>> Protocol that applies only to DDS_DataWriter instances.
- struct DDS_TransportQosPolicy

<<eXtension>> <<cert>> Settings for available transports

- struct DDS_DomainParticipantResourceLimitsQosPolicy

  <<cert>> Resource limits that apply only to DDS_DomainParticipant instances.
- struct DDS_BuiltinTopicKey_t

  The key type of the built-in topic types.
- struct DDS_ConditionSeq

  A sequence object containing references to one or more DDS_Condition objects.
- struct DDS_Listener

  <<interface>> Abstract base class for all Listener interfaces.
- struct DDS_EntityNameQosPolicy

  <<cert>> Policy used to name an entity.
- struct RTI_ManagementQosPolicy

  <<cert>> Policy used determine how DDS_DataReader, DDS_DataWriter, DDS_Publisher, and DDS_Subscriber is managed.

Macros

- #define DDS_TIME_ZERO

  The default instant in time: zero seconds and zero nanoseconds.
- #define T DDS_InstanceHandle_t

  A sequence object containing references to one or more DDS_Condition objects.
- #define DDS_STATUS_MASK_NONE

  No bits are set.
- #define DDS_STATUS_MASK_ALL

  All bits are set.
- #define T struct DDS_QosPolicyCount

  A sequence object containing references to one or more DDS_Condition objects.
- #define DDS_CHECKSUM_NONE (0)

  Bitmap indicating no checksum selection.
- #define DDS_CHECKSUM_AUTO (0xffffU)

  Bitmap indicating automatic checksum selection.
- #define DDS_CHECKSUM_BUILTIN32 (0x1U)

  Bitmap for selecting the builtin 32bit checksum.
- #define DDS_CHECKSUM_BUILTIN64 (0x2U)

  Bitmap for selecting the builtin 64bit checksum.
- #define DDS_CHECKSUM_BUILTIN128 (0x4U)

  Bitmap for selecting the builtin 128bit checksum.
- #define T struct RTPS_Locator

  A sequence object containing references to one or more DDS_Condition objects.
- #define DDS_LOCATOR_INVALID RTPS_LOCATOR_INVALID

  An invalid locator.
- #define DDS_LOCATOR_KIND_UDPv4

  A locator for a UDPv4 address.
- #define DDS_LOCATOR_KIND_UDPv6

  A locator for a UDPv6 address.
- #define DDS_LOCATOR_KIND_RESERVED

  Locator of this kind is reserved.
- #define DDS_LOCATOR_KIND_SHMEM

  A locator for an address acessed via shared memory.
- #define DDS_PROTOCOLVERSION_1_0 { 1, 0 }

  The protocol version 1.0.

- #define DDS_PROTOCOLVERSION_1_1 { 1, 1 }

  The protocol version 1.1.
- #define DDS_PROTOCOLVERSION_1_2 { 1, 2 }

  The protocol version 1.2.
- #define DDS_PROTOCOLVERSION_2_0 { 2, 0 }

  The most recent protocol version. Currently 1.2.
- #define DDS_PROTOCOLVERSION_2_1 { 2, 1 }

  The protocol version 2.1.
- #define DDS_PROTOCOLVERSION { 2, 1 }

  The most recent protocol version. Currently 1.2.
- #define DDS_VENDOR_ID_LENGTH_MAX 2

  Length of vendor id.
- #define DDS_PRODUCTVERSION_UNKNOWN { 0, 0, '0', 0 }

  The value used when the product version is unknown.
- #define T struct DDS_ConditionImpl*

  A sequence object containing references to one or more DDS_Condition objects.
- #define DDS_Listener_INITIALIZER

  Initialize the DDS_Listener::listener_data pointer to NULL.
- #define DDS_ENTITYNAME_QOS_NAME_MAX 255

  Constant determining the maximum length of the name string contained in a DDS_EntityNameQosPolicy instance.

Typedefs

- typedef DDS_HANDLE_TYPE_NATIVE DDS_InstanceHandle_t

  Type definition for an instance handle.
- typedef DDS_UnsignedLong DDS_StatusMask

  A bit-mask (list) of concrete status types, i.e. DDS_StatusKind[].
- typedef DDS_UnsignedShort DDS_ChecksumKind_t

  Datatype for checksum kind.
- typedef DDS_UnsignedShort DDS_ChecksumKindMask_t

  Datatype for an OR'ing for DDS_ChecksumKind_t .
- typedef struct DDS_ProtocolVersion DDS_ProtocolVersion_t

  <<eXtension>> <<cert>> Type used to represent the version of the RTPS protocol.
- typedef struct DDS_BuiltinTopicKey_t DDS_BuiltinTopicKey_t

  The key type of the built-in topic types.
- typedef struct DDS_EntityImpl DDS_Entity

  <<interface>> Abstract base class for all the DDS objects that support QoS policies, and a listener.
- typedef struct DDS_ConditionImpl DDS_Condition

  <<interface>> <<cert>> Root class for all the conditions that may be attached to a DDS_WaitSet.
- typedef struct DDS_GuardConditionImpl DDS_GuardCondition

  <<interface>> <<cert>> A specific DDS_Condition whose `trigger_value` is completely under the control of the application.
- typedef struct DDS_StatusConditionImpl DDS_StatusCondition

  <<interface>> <<cert>> A specific DDS_Condition that is associated with each DDS_Entity.
- typedef struct DDS_WaitSetImpl DDS_WaitSet

  <<interface>> <<cert>> Allows an application to wait until one or more of the attached DDS_Condition objects has a `trigger_value` of DDS_BOOLEAN_TRUE or else until the timeout expires.
- typedef struct DDS_DomainEntityImpl DDS_DomainEntity

<<interface>> Abstract base class for all DDS entities except for the DDS_DomainParticipant.

- typedef struct DDS_DomainParticipantImpl DDS_DomainParticipant

  <<interface>> <<cert>> Container for all DDS_DomainEntity objects.

Enumerations

- enum DDS_ReturnCode_t { DDS_RETCODE_OK, DDS_RETCODE_ERROR, DDS_RETCODE_UNSUPPORTED, DDS_RETCODE_BAD_PARAMETER, DDS_RETCODE_PRECONDITION_NOT_MET, DDS_RETCODE_OUT_OF_RESOURCES, DDS_RETCODE_NOT_ENABLED, DDS_RETCODE_IMMUTABLE_POLICY, DDS_RETCODE_INCONSISTENT_POLICY, DDS_RETCODE_ALREADY_DELETED, DDS_RETCODE_TIMEOUT, DDS_RETCODE_NO_DATA, DDS_RETCODE_ILLEGAL_OPERATION }

  Type for return codes.
- enum DDS_StatusKind { DDS_INCONSISTENT_TOPIC_STATUS, DDS_OFFERED_DEADLINE_MISSED_STATUS, DDS_REQUESTED_DEADLINE_MISSED_STATUS, DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS, DDS_SAMPLE_LOST_STATUS, DDS_SAMPLE_REJECTED_STATUS, DDS_DATA_ON_READERS_STATUS, DDS_DATA_AVAILABLE_STATUS, DDS_LIVELINESS_LOST_STATUS, DDS_LIVELINESS_CHANGED_STATUS, DDS_PUBLICATION_MATCHED_STATUS, DDS_SUBSCRIPTION_MATCHED_STATUS, DDS_INSTANCE_REPLACED_STATUS, DDS_RELIABLE_READER_ACTIVITY_CHANGED_STATUS }

  Type for status kinds.
- enum DDS_QosPolicyId_t { DDS_INVALID_QOS_POLICY_ID , DDS_PRESENTATION_QOS_POLICY_ID, DDS_DEADLINE_QOS_POLICY_ID , DDS_OWNERSHIP_QOS_POLICY_ID, DDS_OWNERSHIPSTRENGTH_QOS_POLICY_ID, DDS_LIVELINESS_QOS_POLICY_ID , DDS_RELIABILITY_QOS_POLICY_ID , DDS_HISTORY_QOS_POLICY_ID , DDS_ENTITYFACTORY_QOS_POLICY_ID }

  Type to identify `QosPolicies`.
- enum DDS_EntityKind_t { DDS_UNKNOWN_ENTITY_KIND = 0, DDS_PARTICIPANT_ENTITY_KIND = 1, DDS_PUBLISHER_ENTITY_KIND = 2, DDS_SUBSCRIBER_ENTITY_KIND = 3, DDS_TOPIC_ENTITY_KIND = 4, DDS_DATAREADER_ENTITY_KIND = 5, DDS_DATAWRITER_ENTITY_KIND = 6 }

  Type of a DDS entity.
- enum DDS_OwnershipQosPolicyKind { DDS_SHARED_OWNERSHIP_QOS, DDS_EXCLUSIVE_OWNERSHIP_QOS }

  Kinds of ownership.
- enum DDS_LivelinessQosPolicyKind { DDS_AUTOMATIC_LIVELINESS_QOS, DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS, DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS }

  Kinds of liveliness.
- enum DDS_ReliabilityQosPolicyKind { DDS_BEST_EFFORT_RELIABILITY_QOS, DDS_RELIABLE_RELIABILITY_QOS }

  Kinds of reliability.
- enum DDS_HistoryQosPolicyKind { DDS_KEEP_LAST_HISTORY_QOS }

  Kinds of history.
- enum DDS_DurabilityQosPolicyKind { DDS_VOLATILE_DURABILITY_QOS, DDS_TRANSIENT_LOCAL_DURABILITY_QOS }

  Enumerated values used to describe the durability of published data.

- enum DDS_PresentationQosPolicyAccessScopeKind { DDS_INSTANCE_PRESENTATION_QOS, DDS_TOPIC_PRESENTATION_QOS, DDS_GROUP_PRESENTATION_QOS, DDS_HIGHEST_OFFERED_PRESENTATION_QOS }

  Kinds of presentation "access scope".
- enum DDS_DestinationOrderQosPolicyKind { DDS_BY_RECEPTION_TIMESTAMP_DESTINATIONORDER_QOS, DDS_BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS }

  Enumerated values used to describe the destination order of published data.
- enum DDS_DataReaderResourceLimitsInstanceReplacementKind { DDS_NO_INSTANCE_REPLACEMENT_QOS, DDS_REPLACE_OLDEST_INSTANCE_REPLACEMENT_QOS }

  Kinds of instance replacement.
- enum { DDS_RTPS_AUTO_ID = 0 }

Functions

- DDS_Boolean DDS_InstanceHandle_equals (const DDS_InstanceHandle_t *self, const DDS_InstanceHandle_t *other)

  <<cert>> Compares this instance handle with another handle for equality.
- DDS_Boolean DDS_InstanceHandle_is_nil (const DDS_InstanceHandle_t *self)

  <<cert>> Compare this handle to DDS_HANDLE_NIL
- DDS_Boolean DDS_Condition_get_trigger_value (DDS_Condition *self)

  <<cert>> Retrieve the `trigger_value`.
- DDS_Condition * DDS_GuardCondition_as_condition (DDS_GuardCondition *guard_cond)

  <<cert>> Access a DDS_GuardCondition's supertype instance.
- DDS_GuardCondition * DDS_GuardCondition_new (void)

  <<cert>> No argument constructor.
- DDS_ReturnCode_t DDS_GuardCondition_set_trigger_value (DDS_GuardCondition *self, DDS_Boolean value)

  <<cert>> Set the guard condition trigger value.
- DDS_Condition * DDS_StatusCondition_as_condition (DDS_StatusCondition *statusCondition)

  <<cert>> Access a DDS_StatusCondition's supertype instance.
- DDS_StatusMask DDS_StatusCondition_get_enabled_statuses (DDS_StatusCondition *self)

  <<cert>> Get the list of statuses enabled on an DDS_Entity.
- DDS_ReturnCode_t DDS_StatusCondition_set_enabled_statuses (DDS_StatusCondition *self, DDS_StatusMask mask)

  <<cert>> This operation defines the list of communication statuses that determine the `trigger_value` of the DDS_StatusCondition.
- DDS_Entity * DDS_StatusCondition_get_entity (DDS_StatusCondition *self)

  <<cert>> Get the DDS_Entity associated with the DDS_StatusCondition.
- DDS_WaitSet * DDS_WaitSet_new (void)

  <<cert>> Create a new, empty waitset.
- DDS_ReturnCode_t DDS_WaitSet_wait (DDS_WaitSet *self, struct DDS_ConditionSeq *active_conditions, const struct DDS_Duration_t *timeout)

  <<cert>> Allows an application thread to wait for the occurrence of certain conditions.
- DDS_ReturnCode_t DDS_WaitSet_attach_condition (DDS_WaitSet *self, DDS_Condition *cond)

  <<cert>> Attaches a DDS_Condition to the DDS_WaitSet.
- DDS_ReturnCode_t DDS_WaitSet_get_conditions (DDS_WaitSet *self, struct DDS_ConditionSeq *attached_conditions)

  <<cert>> Retrieves the list of attached DDS_Condition (s).
- DDS_ReturnCode_t DDS_Entity_enable (DDS_Entity *self)

  <<cert>> Enables the DDS_Entity.
- DDS_InstanceHandle_t DDS_Entity_get_instance_handle (DDS_Entity *self)

  <<cert>> Allows access to the DDS_InstanceHandle_t associated with the DDS_Entity.

- DDS_StatusCondition * DDS_Entity_get_statuscondition (DDS_Entity *self)

  <<cert>> Return the DDS_StatusCondition associated with a particular DDS_Entity.
- DDS_StatusMask DDS_Entity_get_status_changes (DDS_Entity *self)

  <<cert>> Retrieves the list of communication statuses in the DDS_Entity that are triggered.
- DDS_Boolean DDS_EntityNameQosPolicy_set_name (struct DDS_EntityNameQosPolicy *const self, const char *const name)

  Set the EntityName policy.

Variables

- const DDS_Long DDS_TIME_INVALID_SEC

  A sentinel indicating an invalid second of time.
- const DDS_UnsignedLong DDS_TIME_INVALID_NSEC

  A sentinel indicating an invalid nano-second of time.
- struct DDS_Time_t DDS_TIME_INVALID

  A sentinel indicating an invalid time.
- const DDS_Long DDS_DURATION_INFINITE_SEC

  An infinite second period of time.
- const DDS_UnsignedLong DDS_DURATION_INFINITE_NSEC

  An infinite nano-second period of time.
- struct DDS_Duration_t DDS_DURATION_INFINITE

  An infinite period of time.
- const DDS_Long DDS_DURATION_ZERO_SEC

  A zero-length second period of time.
- const DDS_UnsignedLong DDS_DURATION_ZERO_NSEC

  A zero-length nano-second period of time.
- struct DDS_Duration_t DDS_DURATION_ZERO

  A zero-length period of time.
- const DDS_InstanceHandle_t DDS_HANDLE_NIL

  <<cert>> The NIL instance handle.
- struct DDS_GUID_t DDS_GUID_UNKNOWN

  Unknown GUID.
- const char *const DDS_DEADLINE_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DeadlineQosPolicy.
- const char *const DDS_OWNERSHIP_QOS_POLICY_NAME

  Stringified human-readable name for DDS_OwnershipQosPolicy.
- const char *const DDS_OWNERSHIPSTRENGTH_QOS_POLICY_NAME

  Stringified human-readable name for DDS_OwnershipStrengthQosPolicy.
- const char *const DDS_LIVELINESS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_LivelinessQosPolicy.
- const char *const DDS_RELIABILITY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_ReliabilityQosPolicy.
- const char *const DDS_HISTORY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_HistoryQosPolicy.
- const char *const DDS_DURABILITY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DurabilityQosPolicy.
- const char *const DDS_RESOURCELIMITS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_ResourceLimitsQosPolicy.
- const DDS_Long DDS_LENGTH_UNLIMITED

  A special value indicating an unlimited quantity.
- const DDS_Long DDS_LENGTH_AUTO

A special value indicating the resource-limit is determined based on other settings.

- const char *const DDS_DATAREADERRESOURCELIMITS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DataReaderResourceLimitsQosPolicy.
- const char *const DDS_DATAWRITERRESOURCELIMITS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DataWriterResourceLimitsQosPolicy.
- const char *const DDS_ENTITYFACTORY_QOS_POLICY_NAME

  Stringified human-readable name for DDS_EntityFactoryQosPolicy.
- const char *const DDS_SYSTEMRESOURCELIMITS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_SystemResourceLimitsQosPolicy.
- struct DDS_RtpsWellKnownPorts_t DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WELL_KNOWN_PORTS

  Assign to use well-known port mappings which are compatible with previous versions of the RTI Connext Micro middleware.
- struct DDS_RtpsWellKnownPorts_t DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS

  Assign to use well-known port mappings which are compliant with OMG's DDS Interoperability Wire Protocol.
- const char *const DDS_WIREPROTOCOL_QOS_POLICY_NAME

  Stringified human-readable name for DDS_WireProtocolQosPolicy.
- const char *const DDS_DATAREADERPROTOCOL_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DataReaderProtocolQosPolicy.
- const char *const DDS_DATAWRITERPROTOCOL_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DataWriterProtocolQosPolicy.
- const char *const DDS_TRANSPORT_QOS_POLICY_NAME

  Stringified human-readable name for DDS_TransportQosPolicy.
- const char *const DDS_DOMAINPARTICIPANTRESOURCELIMITS_QOS_POLICY_NAME

  Stringified human-readable name for DDS_DomainParticipantResourceLimitsQosPolicy.
- struct DDS_BuiltinTopicKey_t DDS_BUILTINTOPICKEY_AUTO

  Constant indicating an automatically generated keyed instance in one of the built-in topics.
- struct DDS_BuiltinTopicKey_t DDS_BUILTINTOPICKEY_UNKNOWN

  Constant indicating an unknown keyed instance detected on one of the built-in topics.
- const char *const DDS_ENTITYNAME_QOS_POLICY_NAME

  Stringified human-readable name for DDS_EntityNameQosPolicy.
- const char *const DDS_AUTO_NAME_ENTITY

  Constant used to request that an entity be assigned an automatically generated name.

7.11.1 Detailed Description

DDS Domain Module definitions.

7.11.2 Macro Definition Documentation

7.11.2.1 #define T DDS_InstanceHandle_t

A sequence object containing references to one or more DDS_Condition objects.

7.11.2.2 #define T struct DDS_QosPolicyCount

A sequence object containing references to one or more DDS_Condition objects.

7.11.2.3 #define T struct RTPS_Locator

A sequence object containing references to one or more DDS_Condition objects.

7.11.2.4 #define DDS_PRODUCTVERSION_UNKNOWN { 0, 0, '0', 0 }

The value used when the product version is unknown.

7.11.2.5 #define T struct DDS_ConditionImpl*

A sequence object containing references to one or more DDS_Condition objects.

7.11.3 Enumeration Type Documentation

7.11.3.1 enum DDS_EntityKind_t

Type of a DDS entity.

Enumerator:

DDS_UNKNOWN_ENTITY_KIND Entity is of an unknown type.

DDS_PARTICIPANT_ENTITY_KIND Entity is a DDS_DomainParticipant.

DDS_PUBLISHER_ENTITY_KIND Entity is a DDS_Publisher.

DDS_SUBSCRIBER_ENTITY_KIND Entity is a DDS_Subscriber.

DDS_TOPIC_ENTITY_KIND Entity is a DDS_Topic.

DDS_DATAREADER_ENTITY_KIND Entity is a DDS_DataReader.

DDS_DATAWRITER_ENTITY_KIND Entity is a DDS_DataWriter.

7.11.3.2 anonymous enum

Enumerator:

DDS_RTPS_AUTO_ID Indicates that RTI Connext Micro should choose an appropriate host, app, instance or object ID automatically. If this special value is assigned to DDS_WireProtocolQosPolicy::rtps_host_id, DDS_WireProtocolQosPolicy::rtps_app_id, DDS_WireProtocolQosPolicy::rtps_instance_id, DDS_DataWriterProtocolQosPolicy::rtps_object_id or DDS_DataReaderProtocolQosPolicy::rtps_object_id RTI Connext Micro will assign the ID automatically.

QoS:

DDS_WireProtocolQosPolicy::rtps_host_id DDS_WireProtocolQosPolicy::rtps_app_id DDS_WireProtocolQosPolicy::rtps_instance_id

7.11.4 Variable Documentation

7.11.4.1 struct DDS_BuiltinTopicKey_t DDS_BUILTINTOPICKEY_AUTO

Constant indicating an automatically generated keyed instance in one of the built-in topics.

7.11.4.2 struct DDS_BuiltinTopicKey_t DDS_BUILTINTOPICKEY_UNKNOWN

Constant indicating an unknown keyed instance detected on one of the built-in topics.

7.12 dds_c_log.h File Reference

DDS_C module log codes.

Macros

- #define DDSC_LOG_INVALID_DURATION_EC (DDSC_LOG_BASE + 1)

The specified duration is not valid.

- #define DDSC_LOG_INVALID_PARTICIPANT_NAME_EC (DDSC_LOG_BASE + 2)

  An invalid participant name was specified with an unknown GUID.

- #define DDSC_LOG_INVALID_PARTICIPANT_GUID_PREFIX_EC (DDSC_LOG_BASE + 3)

  An invalid participant GUID prefix was specified, typically the GUID prefix does not match an already detected participant.

- #define DDSC_LOG_SYS_GETTIME_EC (DDSC_LOG_BASE + 4)

  A call to OSAPI_System_get_time failed.

- #define DDSC_LOG_GET_NEXT_OBJECT_ID_EC (DDSC_LOG_BASE + 5)

  Failed to get the next automatically generated object ID for an entity's GUID. This typically means the object id pool has been exhausted.

- #define DDSC_LOG_SET_ENTITY_NAME_EC (DDSC_LOG_BASE + 6)

  Failed to set name string for a DDS entity in the DomainParticipantQos entity_name policy.

- #define DDSC_LOG_SYS_GET_HOSTNAME_EC (DDSC_LOG_BASE + 7)

  A call to OSAPI_System_get_hostname failed.

- #define DDSC_LOG_IO_SNPRINTF_FAILED_EC (DDSC_LOG_BASE + 8)

  A call to OSAPI_Stdio_snprintf failed. Typically this means the destination buffer was too small.

- #define DDSC_LOG_TOPIC_FIND_EC (DDSC_LOG_BASE + 9)

  Failed to find a topic created by a DomainParticipant.

- #define DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_NAME_EC (DDSC_LOG_BASE + 10)

  Endpoint discovery failed because the name of the remote participant parent for an endpoint was not found.

- #define DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_KEY_EC (DDSC_LOG_BASE + 11)

  Endpoint discovery failed because the key of the remote participant parent for an endpoint was not found. Note that the key is logged as 4 integers in host endianess format.

- #define DDSC_LOG_REMOTE_PARTICIPANT_KEY_NOT_EQUAL_EC (DDSC_LOG_BASE + 12)

  Failed endpoint discovery when key does not match the remote participant.

- #define DDSC_LOG_INVALID_ENDPOINT_GUID_EC (DDSC_LOG_BASE + 13)

  Failed endpoint discovery due to an invalid or unknown endpoint GUID.

- #define DDSC_LOG_ENDPOINT_NOT_CHILD_OF_PARTICIPANT_EC (DDSC_LOG_BASE + 14)

  Failed endpoint discovery when an endpoint is determined to belong to a different remote participant.

- #define DDSC_LOG_PARTICIPANT_DOES_NOT_EXIST_EC (DDSC_LOG_BASE + 15)

  Failed participant discovery because a remote participant that should have been already asserted locally was not found.

- #define DDSC_LOG_REFRESH_REM_PARTICIPANT_EC (DDSC_LOG_BASE + 16)

  Did not find a remote participant when asserting participant liveliness for it. Note that the key is logged as 4 integers in host endianess format.

- #define DDSC_LOG_REFRESH_REM_PARTICIPANT_TIMEOUT_EC (DDSC_LOG_BASE + 17)

  Failed to assert participant liveliness to a remote participant.

- #define DDSC_LOG_PARTICIPANT_LOOKUP_EC (DDSC_LOG_BASE + 18)

  Failed to find a remote participant as previously discovered.

- #define DDSC_LOG_REMOVE_PUBLICATION_EC (DDSC_LOG_BASE + 19)

  Failed to remove resources for a remote publication.

- #define DDSC_LOG_REMOVE_SUBSCRIPTION_EC (DDSC_LOG_BASE + 20)

  Failed to remove resources for a remote subscription.

- #define DDSC_LOG_FIND_PUBLICATION_PARENT_EC (DDSC_LOG_BASE + 21)

  Cannot determine the participant of a remote publication.

- #define DDSC_LOG_FIND_SUBSCRIPTION_PARENT_EC (DDSC_LOG_BASE + 22)

  Cannot determine the participant of a remote subscription.

- #define DDSC_LOG_MAX_PARTICIPANT_ID_REACHED_EC (DDSC_LOG_BASE + 23)

  Failed to create DomainParticipant due to running out of participant IDs. This is typically caused by all UDP ports being used.

- #define DDSC_LOG_RESERVE_LOCATORS_EC (DDSC_LOG_BASE + 24)

Failed to reserve endpoint locators.

- #define DDSC_LOG_TIMER_CREATE_TIMEOUT_EC (DDSC_LOG_BASE + 25)

  Failed to create a timeout.
- #define DDSC_LOG_ILLEGAL_OBJECTID_EC (DDSC_LOG_BASE + 26)

  Illegal object id specified.
- #define DDSC_LOG_TOPIC_NARROW_EC (DDSC_LOG_BASE + 27)

  Failed to narrow a TopicDescription to the named Topic.
- #define DDSC_LOG_FAILED_UPDATE_STATUS_CONDITION_EC (DDSC_LOG_BASE + 28)

  Failed to update state of StatusCondition.
- #define DDSC_LOG_WS_REMOVE_COND_REFERENCE_EC (DDSC_LOG_BASE + 29)

  Failed to remove a condition reference from a waitset.
- #define DDSC_LOG_WS_ADD_COND_REFERENCE_EC (DDSC_LOG_BASE + 30)

  Failed to add a condition reference to a waitset.
- #define DDSC_LOG_RELEASE_META_MC_EC (DDSC_LOG_BASE + 31)

  Failed to release resources for multicast discovery locators.
- #define DDSC_LOG_RELEASE_META_UC_EC (DDSC_LOG_BASE + 32)

  Failed to release resources for unicast discovery locators.
- #define DDSC_LOG_RELEASE_USER_MC_EC (DDSC_LOG_BASE + 33)

  Failed to release resources for multicast user locators.
- #define DDSC_LOG_RELEASE_USER_UC_EC (DDSC_LOG_BASE + 34)

  Failed to release resources for unicast user locators.
- #define DDSC_LOG_INVALID_DOMAINID_EC (DDSC_LOG_BASE + 35)

  The domain ID specified exceeds what is allowed based on the parameters specified in dp_qos.protocol.rtps_well_known_ports.
- #define DDSC_LOG_RESOLVE_LOCATORS_PRIORITY_EC (DDSC_LOG_BASE + 36)

  Failed to resolve locators based on priority.
- #define DDSC_LOG_DATABASE_CREATE_EC (DDSC_LOG_BASE + 100)

  Failed to create database.
- #define DDSC_LOG_DATABASE_DELETE_EC (DDSC_LOG_BASE + 101)

  Failed to delete database.
- #define DDSC_LOG_TABLE_CREATE_EC (DDSC_LOG_BASE + 102)

  Failed to create database table of the specified name.
- #define DDSC_LOG_TABLE_INUSE_EC (DDSC_LOG_BASE + 103)

  Failed to delete a database table because it is not empty.
- #define DDSC_LOG_TABLE_DELETE_EC (DDSC_LOG_BASE + 104)

  Failed to delete a database table.
- #define DDSC_LOG_TABLE_SELECT_EC (DDSC_LOG_BASE + 105)

  A selection operation failed on the specified database table.
- #define DDSC_LOG_CREATE_INDEX_EC (DDSC_LOG_BASE + 106)

  Failed to create an index on a database table.
- #define DDSC_LOG_DELETE_INDEX_EC (DDSC_LOG_BASE + 107)

  Failed to delete an index on a database table.
- #define DDSC_LOG_DB_CURSOR_INVALIDATED_EC (DDSC_LOG_BASE + 108)

  A database table cursor was invalidated while in use.
- #define DDSC_LOG_SUBSCRIPTION_RECORD 1

  Database record for a Subscription.
- #define DDSC_LOG_PUBLICATION_RECORD 2

  Database record for a Publication.
- #define DDSC_LOG_PARTICIPANT_RECORD 3

  Database record for a Remote Participant.
- #define DDSC_LOG_TOPIC_RECORD 4

Database record for a Topic.

- #define DDSC_LOG_DATAWRITER_RECORD 5

  Database record for a DataWriter.
- #define DDSC_LOG_DATAREADER_RECORD 6

  Database record for a DataReader.
- #define DDSC_LOG_PUBLISHER_RECORD 7

  Database record for a Publisher.
- #define DDSC_LOG_SUBSCRIBER_RECORD 8

  Database record for a Subscriber.
- #define DDSC_LOG_ROUTE_RECORD 9

  Database record for a route record.
- #define DDSC_LOG_BIND_RECORD 10

  Database record for a bind record.
- #define DDSC_LOG_TYPE_RECORD 11

  Database record for a type plugin.
- #define DDSC_LOG_STRING_RECORD 14

  String record.
- #define DDSC_LOG_RECORD_CREATE_EC (DDSC_LOG_BASE + 109)

  Failed to create a database record of the specified kind.
- #define DDSC_LOG_RECORD_DELETE_EC (DDSC_LOG_BASE + 110)

  Failed to delete a database record of the specified kind.
- #define DDSC_LOG_RECORD_INSERT_EC (DDSC_LOG_BASE + 111)

  Failed to insert a database record of the specified kind.
- #define DDSC_LOG_RECORD_ERROR_EC (DDSC_LOG_BASE + 112)

  Unknown error for database record of the specified kind.
- #define DDSC_LOG_RECORD_EXISTS_EC (DDSC_LOG_BASE + 113)

  A database record of the specified kind already exists.
- #define DDSC_LOG_RECORD_LOOKUP_EC (DDSC_LOG_BASE + 114)

  A lookup of a database record of the specified kind failed.
- #define DDSC_LOG_RECORD_NOT_EXISTS_EC (DDSC_LOG_BASE + 115)

  A database record of the specified kind does not exist.
- #define DDSC_LOG_RECORD_SELECT_EC (DDSC_LOG_BASE + 116)

  A database select on the specified record kind failed.
- #define DDSC_LOG_RECORD_REMOVE_EC (DDSC_LOG_BASE + 117)

  Removal a database record of the specified kind failed.
- #define DDSC_LOG_RECORD_INITIALIZE_EC (DDSC_LOG_BASE + 118)

  A database record of the specified kind could not be initialized.
- #define DDSC_LOG_RECORD_FINALIZE_EC (DDSC_LOG_BASE + 119)

  A database record of the specified kind could not be finalized.
- #define DDSC_LOG_PUBLICATIONDATA_OBJECT 1

  DDS PublicationBuiltinTopicData object.
- #define DDSC_LOG_SUBSCRIPTIONDATA_OBJECT 2

  DDS SubscriptionBuiltinTopicData object.
- #define DDSC_LOG_PARTICIPANTDATA_OBJECT 3

  DDS ParticipantBuiltinTopicData object.
- #define DDSC_LOG_DATAREADERQOS_OBJECT 4

  DDS DataReaderQos object.
- #define DDSC_LOG_DATAWRITERQOS_OBJECT 5

  DDS DataWriterQos object.
- #define DDSC_LOG_TOPICQOS_OBJECT 6

  DDS TopicQos object.

- #define DDSC_LOG_PUBLISHERQOS_OBJECT 7

  DDS PublisherQos object.
- #define DDSC_LOG_SUBSCRIBERQOS_OBJECT 8

  DDS SubscriberQos object.
- #define DDSC_LOG_PARTICIPANTQOS_OBJECT 9

  DDS DomainParticipantQos object.
- #define DDSC_LOG_ENTITY_OBJECT 10

  DDS Entity object.
- #define DDSC_LOG_PARTICIPANTFACTORYQOS_OBJECT 11

  DDS DomainParticipantFactoryQos object.
- #define DDSC_LOG_TYPE_OBJECT 12

  A DDS Type object.
- #define DDSC_LOG_BINDRESOLVER_OBJECT 13

  A NETIO BindResolver object.
- #define DDSC_LOG_ROUTERESOLVER_OBJECT 14

  A NETIO RouteResolver object.
- #define DDSC_LOG_ADDRESSRESOLVER_OBJECT 15

  A NETIO AddressResolver object.
- #define DDSC_LOG_DATAWRITERIO_OBJECT 16

  A NETIO DataWriterInterface object.
- #define DDSC_LOG_DATAREADERIO_OBJECT 17

  A NETIO DataReaderInterface object.
- #define DDSC_LOG_LOG_OBJECT 18

  A OSAPI Log object.
- #define DDSC_LOG_SYSTEM_OBJECT 19

  A OSAPI system object.
- #define DDSC_LOG_MUTEX_OBJECT 20

  A OSAPI mutex object.
- #define DDSC_LOG_CONDREF_OBJECT 21

  A DDS Waitset condition reference object.
- #define DDSC_LOG_WSREF_OBJECT 22

  A DDS Condition waitset reference object.
- #define DDSC_LOG_MD5STREAM_OBJECT 23

  A MD5 stream object.
- #define DDSC_LOG_CONDITION_OBJECT 24

  A DDS Condition object.
- #define DDSC_LOG_RT_OBJECT 25

  A RT Condition object.
- #define DDSC_LOG_PARTICIPANT_POOL_OBJECT 26

  A Participant pool object.
- #define DDSC_LOG_TIMER_OBJECT 27

  A OSAPI_Timer object.
- #define DDSC_LOG_TIMEROUT_OBJECT 28

  A OSAPI_Timer timeout object.
- #define DDSC_LOG_TOPIC_OBJECT 29

  A DDS Topic object.
- #define DDSC_LOG_WAITSET_OBJECT 30

  A DDS WaitSet object.
- #define DDSC_LOG_UUID_OBJECT 31

  A UUID object.
- #define DDSC_LOG_STRING_OBJECT 40

A String object.

- #define DDSC_LOG_STRINGMANAGER_OBJECT 43

  A String Manager object.

- #define DDSC_LOG_OBJECT_INITIALIZE_EC (DDSC_LOG_BASE + 200)

  Out of resources to initialize object of the specified kind.

- #define DDSC_LOG_OBJECT_ALLOCATE_EC (DDSC_LOG_BASE + 201)

  Out of resources to allocate an object of the specified kind.

- #define DDSC_LOG_OBJECT_FINALIZE_EC (DDSC_LOG_BASE + 202)

  Failed to finalize object of specified kind.

- #define DDSC_LOG_OBJECT_DELETE_EC (DDSC_LOG_BASE + 203)

  Failed to delete object of specified kind.

- #define DDSC_LOG_OBJECT_COPY_EC (DDSC_LOG_BASE + 204)

  Failed to copy object of specified kind.

- #define DDSC_LOG_OBJECT_REFCOUNT_EC (DDSC_LOG_BASE + 205)

  Failed to delete/finalize an object because other objects are referencing it.

- #define DDSC_LOG_OBJECT_GET_PROPERTY_EC (DDSC_LOG_BASE + 206)

  Failed to get the object properties.

- #define DDSC_LOG_OBJECT_SET_PROPERTY_EC (DDSC_LOG_BASE + 207)

  Failed to set the object properties.

- #define DDSC_LOG_OBJECT_EMPTY_EC (DDSC_LOG_BASE + 208)

  An object is empty, typically applies only to buffer-pool objects.

- #define DDSC_LOG_OBJECT_INUSECOUNT_EC (DDSC_LOG_BASE + 209)

  Failed to delete/finalize an object because other objects are using it.

- #define DDSC_LOG_OBJECT_CREATE_EC (DDSC_LOG_BASE + 210)

  Failed to create an object.

- #define DDSC_LOG_NETMASK_SEQUENCE 1

  A NETIO Netmask sequence.

- #define DDSC_LOG_ROUTE_SEQUENCE 2

  A NETIO Route sequence.

- #define DDSC_LOG_RESERVED_SEQUENCE 3

  A NETIO Reserved address sequence.

- #define DDSC_LOG_ENABLED_USER_TRANSPORT_SEQUENCE 4

  A DDS sequence of enabled user NETIO transports.

- #define DDSC_LOG_ENABLED_TRANSPORT_SEQUENCE 5

  A DDS sequence of enabled transports.

- #define DDSC_LOG_ENABLED_DISCVOERY_TRANSPORT_SEQUENCE 6

  A DDS sequence of enabled discovery transports.

- #define DDSC_LOG_INITIAL_PEER_SEQUENCE 7

  A DDS sequence of initial peers.

- #define DDSC_LOG_DESTINATION_SEQUENCE 8

  A NETIO sequence of destinations to send to.

- #define DDSC_LOG_METAUNICAST_SEQUENCE 9

  A DDS sequence of meta-traffic unicast locators.

- #define DDSC_LOG_METAMULTICAST_SEQUENCE 10

  A DDS sequence of meta-traffic multicast locators.

- #define DDSC_LOG_USERUNICAST_SEQUENCE 11

  A DDS sequence of user-traffic unicast locators.

- #define DDSC_LOG_USERMULTICAST_SEQUENCE 12

  A DDS sequence of user-traffic multicast locators.

- #define DDSC_LOG_READTAKE_SEQUENCE 13

  A DDS sequence used in a read/take call.

- #define DDSC_LOG_SEQUENCE_SETMAX_EC (DDSC_LOG_BASE + 300)

  Failed to set the maximum length of a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_SETLENGTH_EC (DDSC_LOG_BASE + 301)

  Failed to set the length of a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_GETREF_EC (DDSC_LOG_BASE + 302)

  Failed to get a reference at the specified index for a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_INITIALIZE_EC (DDSC_LOG_BASE + 303)

  Failed to initialize a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_FINALIZE_EC (DDSC_LOG_BASE + 304)

  Failed to finalize a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_COPY_EC (DDSC_LOG_BASE + 305)

  Failed to copy a sequence of the specified kind.
- #define DDSC_LOG_SEQUENCE_INVALID_EC (DDSC_LOG_BASE + 306)

  The sequence of the specified kind was invalid in the context it is used.
- #define DDSC_LOG_DISCOVERY_COMPONENT 1

  A component of discovery kind.
- #define DDSC_LOG_RTPS_COMPONENT 2

  A component of RTPS kind.
- #define DDSC_LOG_READERHISTORY_COMPONENT 3

  A component of reader-history kind.
- #define DDSC_LOG_WRITERHISTORY_COMPONENT 4

  A component of writer-history kind.
- #define DDSC_LOG_DATAREADERIO_COMPONENT 5

  A component of DataReaderInterface kind.
- #define DDSC_LOG_DATAWRITERIO_COMPONENT 6

  A component of DataWriterInterface kind.
- #define DDSC_LOG_NETIO_COMPONENT 7

  A component of NETIO kind.
- #define DDSC_LOG_TRANSPORT_COMPONENT 8

  A component of transport kind.
- #define DDSC_LOG_COMPONENT_LOOKUP_EC (DDSC_LOG_BASE + 400)

  Did not find a component factory with the given name in the registry.
- #define DDSC_LOG_COMPONENT_CREATE_EC (DDSC_LOG_BASE + 401)

  Could not create a component of the specified kind using the specified factory.
- #define DDSC_LOG_COMPONENT_DELETE_EC (DDSC_LOG_BASE + 402)

  Could not delete a component of the specified kind using the specified factory.
- #define DDSC_LOG_HISTORY_RESOURCE 1

  Exceeded resource limits for writer history queue.
- #define DDSC_LOG_SAMPLE_RESOURCES 2

  Failed to get resource for new sample due to resource limit.
- #define DDSC_LOG_PARTICIPANT_RESOURCES 3

  Failed to allocate a new participant.
- #define DDSC_LOG_RESOURCE_EXCEEDED_EC (DDSC_LOG_BASE + 500)

  Could not allocate a resource of the specified kind.
- #define DDSC_LOG_TOPIC_QOS 1

  The TopicQos kind.
- #define DDSC_LOG_DATAREADER_QOS 2

  The DataReaderQos kind.
- #define DDSC_LOG_DATAWRITER_QOS 3

  The DataWriterQos kind.
- #define DDSC_LOG_SUBSCRIBER_QOS 4

The SubscriberQos kind.

- #define DDSC_LOG_PUBLISHER_QOS 5

  The PublisherQos kind.

- #define DDSC_LOG_PARTICIPANT_QOS 6

  The DomainParticipantQos kind.

- #define DDSC_LOG_PARTICIPANTFACTORY_QOS 7

  The DomainParticipantFactoryQos kind.

- #define DDSC_LOG_DEFAULTTOPIC_QOS 8

  The default TopicQos kind.

- #define DDSC_LOG_DEFAULTDATAREADER_QOS 9

  The default DataReaderQos kind.

- #define DDSC_LOG_DEFAULTDATAWRITER_QOS 10

  The default DataWriterQos kind.

- #define DDSC_LOG_DEFAULTSUBSCRIBER_QOS 11

  The default SubscriberQos kind.

- #define DDSC_LOG_DEFAULTPUBLISHER_QOS 12

  The default PublisherQos kind.

- #define DDSC_LOG_DEFAULTPARTICIPANT_QOS 13

  The default PublisherQos kind.

- #define DDSC_LOG_DEADLINE_QOS_POLICY 14

  The deadline qos policy kind.

- #define DDSC_LOG_LIVELINESS_QOS_POLICY 15

  The liveliness qos policy kind.

- #define DDSC_LOG_HISTORY_QOS_POLICY 16

  The history qos policy kind.

- #define DDSC_LOG_RESOURCE_LIMIT_QOS_POLICY 17

  The resource limits qos policy kind.

- #define DDSC_LOG_PROTOCOL_QOS_POLICY 18

  The protocol qos policy kind.

- #define DDSC_LOG_TYPE_SUPPORT_QOS_POLICY 19

  The type-support qos policy kind.

- #define DDSC_LOG_RELIABILITY_QOS_POLICY 20

  The reliability qos policy kind.

- #define DDSC_LOG_DURABILITY_QOS_POLICY 21

  The durability qos policy kind.

- #define DDSC_LOG_OWNERSHIP_QOS_POLICY 22

  The ownership qos policy kind.

- #define DDSC_LOG_OWNERSHIP_STRENGTH_QOS_POLICY 23

  The ownership-strength qos policy kind.

- #define DDSC_LOG_TRANSPORT_QOS_POLICY 24

  The transport qos policy kind.

- #define DDSC_LOG_PARTICIPANT_ID_QOS_POLICY 25

  The participant id qos policy kind.

- #define DDSC_LOG_HEARTBEATS_QOS_POLICY 26

  The heartbeat qos policy kind.

- #define DDSC_LOG_DATAWRITER_RESOURCE_QOS_POLICY 27

  The DataWriterQos writer_resource_limits policy.

- #define DDSC_LOG_DATAREADER_RESOURCE_QOS_POLICY 28

  The DataReaderQos reader_resource_limits policy.

- #define DDSC_LOG_DESTINATION_ORDER_POLICY 29

  The DestinationOrderQos destination_order policy.

- #define DDSC_LOG_USER_TRAFFIC_QOS_POLICY 30

  The UserTraffic policy.
- #define DDSC_LOG_DISCOVERY_TRAFFIC_QOS_POLICY 31

  The Discovery.enabled_transport policy.
- #define DDSC_LOG_QOS_INCONSISTENT_POLICY_EC (DDSC_LOG_BASE + 501)

  An inconsistent Qos policy for the specified Qos kind was found.
- #define DDSC_LOG_QOS_INCONSISTENT_POLICIES_EC (DDSC_LOG_BASE + 502)

  Inconsistency between two Qos policies for the specified Qos kind was found.
- #define DDSC_LOG_QOS_INCONSISTENT_EC (DDSC_LOG_BASE + 503)

  Failed to create an entity or set a qos due to inconsistent policy.
- #define DDSC_LOG_QOS_COPY_EC (DDSC_LOG_BASE + 504)

  Failed to copy a Qos of the specified kind.
- #define DDSC_LOG_QOS_INITIALIZE_EC (DDSC_LOG_BASE + 505)

  Failed to initialize a Qos of the specified kind.
- #define DDSC_LOG_QOS_FINALIZE_EC (DDSC_LOG_BASE + 506)

  Failed to finalize a Qos of the specified kind.
- #define DDSC_LOG_QOS_SET_EC (DDSC_LOG_BASE + 507)

  Failed to set a Qos of the specified kind.
- #define DDSC_LOG_QOS_SET_ON_ENABLED_EC (DDSC_LOG_BASE + 508)

  Failed to set a Qos of the specified kind because the entity is already enabled.
- #define DDSC_LOG_QOS_IMMUTABLE_EC (DDSC_LOG_BASE + 509)

  Failed to set a Qos of the specified kind the immutable Qos policies have been changed.
- #define DDSC_LOG_QOS_CHANGED_EC (DDSC_LOG_BASE + 510)

  A discovered Qos changed (the entity already existed)
- #define DDSC_LOG_QOS_GET_EC (DDSC_LOG_BASE + 511)

  Failed to get a Qos of the specified kind.
- #define DDSC_LOG_TOPIC_LISTENER 1

  The topic listener kind.
- #define DDSC_LOG_DATAREADER_LISTENER 2

  The datareader listener kind.
- #define DDSC_LOG_DATAWRITER_LISTENER 3

  The datawriter listener kind.
- #define DDSC_LOG_SUBSCRIBER_LISTENER 4

  The subscriber listener kind.
- #define DDSC_LOG_PUBLISHER_LISTENER 5

  The publisher listener kind.
- #define DDSC_LOG_PARTICIPANT_LISTENER 6

  The participant listener kind.
- #define DDSC_LOG_PARTICIPANTFACTORY_LISTENER 7

  The participant-factory listener kind.
- #define DDSC_LOG_LISTENER_INCONSISTENT_EC (DDSC_LOG_BASE + 512)

  Failed to create an entity due to inconsistent listener and status mask.
- #define DDSC_LOG_LISTENER_SET_EC (DDSC_LOG_BASE + 513)

  Failed to set the listener of the specified kind.
- #define DDSC_LOG_LISTENER_GET_EC (DDSC_LOG_BASE + 514)

  Failed to get the listener of the specified kind.
- #define DDSC_LOG_LISTENER_SET_ILLEGAL_NULL_EC (DDSC_LOG_BASE + 515)

  Illegal combination of NULL listener and non-NONE status mask when setting a listener for an Entity.
- #define DDSC_LOG_TOPIC_ENTITY 1

  The topic entity kind.
- #define DDSC_LOG_DATAWRITER_ENTITY 2

The datawriter entity kind.

- #define DDSC_LOG_DATAREADER_ENTITY 3

  The datareader entity kind.
- #define DDSC_LOG_PUBLISHER_ENTITY 4

  The publisher entity kind.
- #define DDSC_LOG_SUBSCRIBER_ENTITY 5

  The subscriber entity kind.
- #define DDSC_LOG_PARTICIPANT_ENTITY 6

  The participant entity kind.
- #define DDSC_LOG_PUBLICATION_ENTITY 7

  The publication entity kind.
- #define DDSC_LOG_SUBSCRIPTION_ENTITY 8

  The subscription entity kind.
- #define DDSC_LOG_ENTITY_ENABLE_EC (DDSC_LOG_BASE + 600)

  Failed to enable an entity of the specified kind.
- #define DDSC_LOG_ENTITY_NOT_EMPTY_EC (DDSC_LOG_BASE + 601)

  Failed to an delete/finalize an entity of the specified kind because it is not empty.
- #define DDSC_LOG_ENTITY_FINALIZE_EC (DDSC_LOG_BASE + 602)

  Failed to finalize an entity of the specified kind.
- #define DDSC_LOG_ENTITY_INITIALIZE_EC (DDSC_LOG_BASE + 603)

  Failed to initialize an entity of the specified kind.
- #define DDSC_LOG_ENTITY_NOT_ENABLED_EC (DDSC_LOG_BASE + 604)

  An operation was attempted on an entity that is not enabled.
- #define DDSC_LOG_ENTITY_DIFFERENT_FACTORY_EC (DDSC_LOG_BASE + 605)

  Entities are in different factories.
- #define DDSC_LOG_DATAWRITER_CDR 1

  The datawriter CDR kind.
- #define DDSC_LOG_DATAREADER_CDR 2

  The datareader CDR kind.
- #define DDSC_LOG_CDR_POOL_ALLOC_EC (DDSC_LOG_BASE + 700)

  Failed to allocate a pool of the specified kind.
- #define DDSC_LOG_CDR_BUFFER_SET_EC (DDSC_LOG_BASE + 701)

  Failed to set the CDR buffer for a packet.
- #define DDSC_LOG_CDR_POOL_DELETE_EC (DDSC_LOG_BASE + 702)

  Failed to delete the CDR pool.
- #define DDSC_LOG_CDR_SERIALIZE_PID_EC (DDSC_LOG_BASE + 703)

  Failed to serialize a parameter ID.
- #define DDSC_LOG_CDR_SERIALIZE_PID_LENGTH_EC (DDSC_LOG_BASE + 704)

  Failed to serialize a parameter length.
- #define DDSC_LOG_CDR_SERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 705)

  Failed to serialize a key-hash.
- #define DDSC_LOG_CDR_SERIALIZE_DATA_EC (DDSC_LOG_BASE + 706)

  Failed to serialize payload data.
- #define DDSC_LOG_DESERIALIZE_BAD_PID_LENGTH_EC (DDSC_LOG_BASE + 707)

  Deserialized an invalid parameter length for a specific parameter ID.
- #define DDSC_LOG_CDR_DESERIALIZE_PID_EC (DDSC_LOG_BASE + 708)

  Failed to deserialize the ID of an inline parameter.
- #define DDSC_LOG_CDR_DESERIALIZE_PID_LENGTH_EC (DDSC_LOG_BASE + 709)

  Failed to deserialize the length of an inline parameter.
- #define DDSC_LOG_CDR_INCREMENT_POS_EC (DDSC_LOG_BASE + 710)

  Failed to increment to the position of the next inline parameter.

- #define DDSC_LOG_CDR_SET_POS_EC (DDSC_LOG_BASE + 711)

  Failed to set the reception stream position.
- #define DDSC_LOG_CDR_DESERIALIZE_HEADER_EC (DDSC_LOG_BASE + 712)

  Failed to deserialize the encapsulation header.
- #define DDSC_LOG_CDR_DESERIALIZE_DATA_EC (DDSC_LOG_BASE + 713)

  Failed to deserialize CDR payload data.
- #define DDSC_LOG_CDR_INITIALIZE_SAMPLE_EC (DDSC_LOG_BASE + 714)

  Failed to initialize CDR sample.
- #define DDSC_LOG_CDR_FINALIZE_SAMPLE_EC (DDSC_LOG_BASE + 715)

  Failed to finalize sample.
- #define DDSC_LOG_CDR_SERIALIZE_STATUS_INFO_EC (DDSC_LOG_BASE + 716)

  Failed to serialize the status info parameter.
- #define DDSC_LOG_CDR_DESERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 717)

  Failed to deserialize a key-hash.
- #define DDSC_LOG_CDR_DESERIALIZE_KEY_EC (DDSC_LOG_BASE + 718)

  Failed to deserialize CDR payload key.
- #define DDSC_LOG_NETIO_ADD_ANON_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 800)

  Failed to add a route to an anonymous participant discovery datawriter.
- #define DDSC_LOG_NETIO_ADD_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 801)

  Failed to add a route to a topic from a datawriter.
- #define DDSC_LOG_NETIO_DELETE_TOPIC_ROUTE_EC (DDSC_LOG_BASE + 802)

  Failed to delete a route to a topic.
- #define DDSC_LOG_NETIO_FORWARD_TOPIC_EC (DDSC_LOG_BASE + 803)

  Failed to forward a topic.
- #define DDSC_LOG_NETIO_NETIO_KIND 1

  Any NETIO interface kind, typically UDP.
- #define DDSC_LOG_INTRA_NETIO_KIND 2

  The intra interface kind.
- #define DDSC_LOG_RTPS_NETIO_KIND 3

  The RTPS interface kind.
- #define DDSC_LOG_DATAREADER_NETIO_KIND 4

  The DataReader interface kind.
- #define DDSC_LOG_DATAWRITER_NETIO_KIND 5

  The DataWriter interface kind.
- #define DDSC_LOG_NETIO_BIND_EXTERNAL_EC (DDSC_LOG_BASE + 804)

  Failed to bind two external interface of the specified kind.
- #define DDSC_LOG_NETIO_UNBIND_EXTERNAL_EC (DDSC_LOG_BASE + 805)

  Failed to unbind two external interfaces of the specified kind.
- #define DDSC_LOG_NETIO_BIND_EC (DDSC_LOG_BASE + 806)

  Failed to bind an interface to a peer interface.
- #define DDSC_LOG_NETIO_UNBIND_EC (DDSC_LOG_BASE + 807)

  Failed to unbind an interface from a peer interface.
- #define DDSC_LOG_NETIO_ADD_ROUTE_EC (DDSC_LOG_BASE + 808)

  Failed to add a route from an interface to a peer interface.
- #define DDSC_LOG_NETIO_DELETE_ROUTE_EC (DDSC_LOG_BASE + 809)

  Failed to delete a route from an interface to a peer interface.
- #define DDSC_LOG_NETIO_GET_EXTERNAL_INTF_EC (DDSC_LOG_BASE + 810)

  Failed to get an external interface for the specified interface kind.
- #define DDSC_LOG_NETIO_NO_ROUTE_EC (DDSC_LOG_BASE + 811)

  A DataReader failed a bind due to no existing route.
- #define DDSC_LOG_NETIO_ROUTE_LOOKUP_FAILED_EC (DDSC_LOG_BASE + 812)

Lookup a route to a destination failed.

- #define DDSC_LOG_NETIO_GET_ROUTE_TABLE_FAILED_EC (DDSC_LOG_BASE + 813)

  Failed to get the route table for an interface.
- #define DDSC_LOG_NETIO_SEND_FAILED_EC (DDSC_LOG_BASE + 814)

  Failure when sending on an interface.
- #define DDSC_LOG_NETIO_SET_STATE_EC (DDSC_LOG_BASE + 815)

  Failed to set an interface state.
- #define DDSC_LOG_NETIO_PEER_LOOKUP_EC (DDSC_LOG_BASE + 816)

  Datawriter did not find a peer.
- #define DDSC_LOG_NETIO_FORCED_REMOVE_EC (DDSC_LOG_BASE + 817)

  Forced removal of sample downstream failed.
- #define DDSC_LOG_PACKET_INIT_EC (DDSC_LOG_BASE + 818)

  Failed to initialize a packet.
- #define DDSC_LOG_PACKET_SET_HEAD_EC (DDSC_LOG_BASE + 819)

  Failed to set the head of a packet.
- #define DDSC_LOG_PACKET_SET_TAIL_EC (DDSC_LOG_BASE + 820)

  Failed to set the tail of a packet.
- #define DDSC_LOG_NETIO_DISCOVERY_ENABLED_TRANSPORT_EC (DDSC_LOG_BASE + 821)

  Configured discovery enabled transport is not valid.
- #define DDSC_LOG_DW_ACKNACK_FAILED_EC (DDSC_LOG_BASE + 900)

  Failed to ACKNACK sample in the writer history.
- #define DDSC_LOG_DW_COMMIT_EC (DDSC_LOG_BASE + 901)

  Failed to commit a sample to the writer queue.
- #define DDSC_LOG_DW_KEYHASH_CREATE_EC (DDSC_LOG_BASE + 902)

  Failed to create keyhash of instance handle.
- #define DDSC_LOG_DW_ILLEGAL_KEY_KIND_EC (DDSC_LOG_BASE + 903)

  Failed a write due to an invalid key kind for the type being written.
- #define DDSC_LOG_DW_CREATE_TYPED_WRITER_EC (DDSC_LOG_BASE + 904)

  Failed to create a typed writer.
- #define DDSC_LOG_DW_HISTORY_REGISTER_KEY_EC (DDSC_LOG_BASE + 905)

  Failed to register the key of an instance.
- #define DDSC_LOG_DR_CREATE_TYPED_READER_EC (DDSC_LOG_BASE + 1000)

  Failed to create a typed datareader.
- #define DDSC_LOG_DR_COPY_DATA_SAMPLE_EC (DDSC_LOG_BASE + 1001)

  Failed to copy a sample upon reception, read, or take.
- #define DDSC_LOG_DR_COMMIT_SAMPLE_EC (DDSC_LOG_BASE + 1002)

  Failed to commit a sample to be made available to be read or taken.
- #define DDSC_LOG_DR_FILTER_ERROR_EC (DDSC_LOG_BASE + 1003)

  A datareader filter function failed.
- #define DDSC_LOG_DR_DESERIALIZE_KEYHASH_EC (DDSC_LOG_BASE + 1004)

  Failed to deserialize a key-hash parameter.
- #define DDSC_LOG_DR_GET_ENTRY_FAILED_EC (DDSC_LOG_BASE + 1005)

  Failed to get a Reader History entry for a received sample.
- #define DDSC_LOG_DR_COMMIT_ENTRY_EC (DDSC_LOG_BASE + 1006)

  Failed to commit a receive sample to Reader History to be read or taken.
- #define DDSC_LOG_DR_UNREGISTER_KEY_EC (DDSC_LOG_BASE + 1007)

  A DataReader failed to unregister an instance.
- #define DDSC_LOG_DR_DISPOSE_KEY_EC (DDSC_LOG_BASE + 1008)

  A DataReader failed to dispose an instance.
- #define DDSC_LOG_DR_READ_TAKE_FAILURE_EC (DDSC_LOG_BASE + 1009)

  A call to a reader/take function failed.

- #define DDSC_LOG_DR_INSTANCE_TO_KEYHASH_EC (DDSC_LOG_BASE + 1010)

  Failed to create keyhash of instance handle.
- #define DDSC_LOG_TYPE_NAME_CMP_EC (DDSC_LOG_BASE + 1100)

  Two type names are incompatible.
- #define DDSC_LOG_TOPIC_NAME_CMP_EC (DDSC_LOG_BASE + 1101)

  Two topic names are incompatible.
- #define DDSC_LOG_TYPE_FUNCTION_GET_SERIALIZED_SAMPLE_MAX_SIZE 1

  The get_serialized_sample_max_size function kind.
- #define DDSC_LOG_TYPE_FUNCTION_SERIALIZE_DATA 2

  The serialize_data function kind.
- #define DDSC_LOG_TYPE_FUNCTION_DESERIALIZE_DATA 3

  The deserialize_data function kind.
- #define DDSC_LOG_TYPE_FUNCTION_CREATE_SAMPLE 4

  The create_sample function kind.
- #define DDSC_LOG_TYPE_FUNCTION_COPY_SAMPLE 5

  The copy_sample function kind.
- #define DDSC_LOG_TYPE_FUNCTION_DELETE_SAMPLE 6

  The delete_sample function kind.
- #define DDSC_LOG_TYPE_FUNCTION_GET_KEY_KIND 7

  The get_key_kind function kind.
- #define DDSC_LOG_TYPE_FUNCTION_INSTANCE_TO_KEYHASH 8

  The instance_to_keyhash function kind.
- #define DDSC_LOG_TYPE_FUNCTION_NULL_EC (DDSC_LOG_BASE + 1102)

  Invalid type plugin, The specified function pointer is NULL.
- #define DDSC_LOG_TOPIC_TOO_LONG_EC (DDSC_LOG_BASE + 1103)

  Failed to create a topic because the name exceeded the maximum length of 255 octets (excluding the terminating NUL)
- #define DDSC_LOG_TYPE_TOO_LONG_EC (DDSC_LOG_BASE + 1104)

  Failed to create a type because the name exceeded the maximum length of 255 octets (excluding the terminating NUL)
- #define DDSC_LOG_LOOKUP_TYPE_PLUGIN_EC (DDSC_LOG_BASE + 1105)

  A type-plugin for the given type could not be found.
- #define DDSC_LOG_TYPE_KEY_TYPE_EC (DDSC_LOG_BASE + 1106)

  Two types have incompatible keys.
- #define DDSC_LOG_TYPE_MANAGED_POOL_FAILURE_EC (DDSC_LOG_BASE + 1107)

  Managed pool function failed.
- #define DDSC_LOG_TYPE_GET_SAMPLE_ID_EC (DDSC_LOG_BASE + 1108)

  Managed pool failed to get sample id.
- #define DDSC_LOG_DISC_LOCAL_PARTICIPANT_ENABLED_EC (DDSC_LOG_BASE + 1200)

  Discovery plugin failed its update after a local DomainParticipant was enabled.
- #define DDSC_LOG_DISC_BEFORE_LOCAL_PARTICIPANT_CREATED_EC (DDSC_LOG_BASE + 1201)

  Discovery plugin failed its update before a local DomainParticipant was created.
- #define DDSC_LOG_DISC_AFTER_LOCAL_PARTICIPANT_CREATED_EC (DDSC_LOG_BASE + 1202)

  Discovery plugin failed its update after a local DomainParticipant was created.
- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_ENABLED_EC (DDSC_LOG_BASE + 1203)

  Discovery plugin failed its update after a local DataReader was enabled.
- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_DELETED_EC (DDSC_LOG_BASE + 1204)

  Discovery plugin failed its update after a local DataReader was deleted.
- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_ENABLED_EC (DDSC_LOG_BASE + 1205)

Discovery plugin failed its update after a local DataWriter was enabled.

- #define DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_DELETED_EC (DDSC_LOG_BASE + 1206)

  Discovery plugin failed its update after a local DataWriter was deleted.
- #define DDSC_LOG_DISC_REMOTE_PARTICIPANT_EXPIRED_EC (DDSC_LOG_BASE + 1207)

  Discovery plugin failed its update after a remote DomainParticipant's liveliness expired.
- #define DDSC_LOG_DISC_ADD_PEER_EC (DDSC_LOG_BASE + 1208)

  Failed to add a peer with discovery plugin.
- #define DDSC_LOG_CHECKSUM_REQUIRED_EC (DDSC_LOG_BASE + 1301)

  A participant requires checksum, but the discovered participant does not send a checksum.
- #define DDSC_LOG_CHECKSUM_INCOMPATIBLE_EC (DDSC_LOG_BASE + 1302)

  A participant does not support checksum's sent by a discovered participant.
- #define DDSC_LOG_CHECKSUM_INCONSISTENT_CLASS_EC (DDSC_LOG_BASE + 1303)

  The checksum class id for a custom checksum does match.
- #define DDSC_LOG_CHECKSUM_INCONSISTENT_COMPUTE_EC (DDSC_LOG_BASE + 1304)

  The computed_crc_kind is inconsistent with what is allowed or supported.
- #define DDSC_LOG_CHECKSUM_INCONSISTENT_ALLOW_EC (DDSC_LOG_BASE + 1305)

  The allowed_crc_mask is inconsistent with what is supported.

7.12.1 Detailed Description

DDS_C module log codes.

7.13 dds_c_publication.h File Reference

DDS publication module.

Data Structures

- struct DDS_OfferedDeadlineMissedStatus

  <<cert>> DDS_OFFERED_DEADLINE_MISSED_STATUS
- struct DDS_LivelinessLostStatus

  <<cert>> DDS_LIVELINESS_LOST_STATUS
- struct DDS_OfferedIncompatibleQosStatus

  <<cert>> DDS_OFFERED_INCOMPATIBLE_QOS_STATUS
- struct DDS_PublicationMatchedStatus

  <<cert>> DDS_PUBLICATION_MATCHED_STATUS
- struct DDS_ReliableReaderActivityChangedStatus

  <<eXtension>> <<cert>> Describes the activity (i.e. are acknowledgements forthcoming) of reliable readers matched to a reliable writer.
- struct DDS_ReliableSampleUnacknowledgedStatus

  <<cert>> Status about unacknowledged samples.
- struct DDS_DataWriterQos

  <<cert>> QoS policies supported by a DDS_DataWriter entity.
- struct DDS_PublisherQos

  <<cert>> QoS policies supported by a DDS_Publisher entity.
- struct DDS_DataWriterListener

  <<interface>> <<cert>> DDS_Listener for writer status.
- struct DDS_PublisherListener

  <<interface>> <<cert>> DDS_Listener for DDS_Publisher status.

Macros

- #define DDS_OfferedDeadlineMissedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_LivelinessLostStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_OfferedIncompatibleQosStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_PublicationMatchedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_ReliableReaderActivityChangedStatus_INITIALIZER

  <<cert>> Initializer for new status instances.
- #define DDS_DataWriterQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_PublisherQos_INITIALIZER

  <<cert>> Initializer for new QoS instances.
- #define DDS_DataWriterListener_INITIALIZER

  <<cert>> Initializer for new DDS_DataWriterListener.
- #define DDS_PublisherListener_INITIALIZER

  <<cert>> Initializer for new DDS_PublisherListener.

Typedefs

- typedef struct DDS_DataWriterImpl DDS_DataWriter

  <<interface>> <<cert>> Allows an application to set the value of the data to be published under a given DDS_Topic.
- typedef struct DDS_PublisherImpl DDS_Publisher

  <<interface>> <<cert>> A publisher is the object responsible for the actual dissemination of publications.
- typedef void(* DDS_DataWriterListener_OfferedDeadlineMissedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_OfferedDeadlineMissedStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_offered_deadline_missed function.
- typedef void(* DDS_DataWriterListener_LivelinessLostCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_LivelinessLostStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_liveliness_lost function.
- typedef void(* DDS_DataWriterListener_OfferedIncompatibleQosCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_OfferedIncompatibleQosStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_offered_incompatible_qos function.
- typedef void(* DDS_DataWriterListener_PublicationMatchedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_PublicationMatchedStatus *status)

  <<cert>> Prototype of a DDS_DataWriterListener on_publication_matched function.
- typedef void(* DDS_DataWriterListener_ReliableReaderActivityChangedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_ReliableReaderActivityChangedStatus *status)

  <<eXtension>> <<cert>> A matched reliable reader has become active or become inactive.
- typedef void(* DDS_DataWriterListener_ReliableSampleUnacknowledgedCallback)(void *listener_data, DDS_DataWriter *writer, const struct DDS_ReliableSampleUnacknowledgedStatus *status)

  <<eXtension>> <<cert>> Unacknowledged samples are removed from the writer cache.

Functions

- DDS_ReturnCode_t DDS_DataWriterQos_initialize (struct DDS_DataWriterQos *self)

  <<cert>> Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_DataWriterQos_copy (struct DDS_DataWriterQos *self, const struct DDS_DataWriterQos *source)

  Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_DataWriterQos_is_equal (const struct DDS_DataWriterQos *left, const struct DDS_DataWriterQos *right)

  Compare two DDS_DataWriterQos policies for equality.
- DDS_ReturnCode_t DDS_PublisherQos_initialize (struct DDS_PublisherQos *self)

  <<cert>> Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_PublisherQos_copy (struct DDS_PublisherQos *self, const struct DDS_PublisherQos *source)

  <<cert>> Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_PublisherQos_is_equal (const struct DDS_PublisherQos *left, const struct DDS_PublisherQos *right)

  <<cert>> Compare two DDS_PublisherQos policies for equality.
- DDS_Entity * DDS_Publisher_as_entity (DDS_Publisher *publisher)

  <<cert>> Access a DDS_Publisher's supertype instance.
- DDS_DataWriter * DDS_Publisher_create_datawriter (DDS_Publisher *self, DDS_Topic *topic, const struct DDS_DataWriterQos *qos, const struct DDS_DataWriterListener *listener, DDS_StatusMask mask)

  <<cert>> Creates a DDS_DataWriter that will be attached and belong to the DDS_Publisher.
- DDS_DataWriter * DDS_Publisher_lookup_datawriter (DDS_Publisher *self, const char *topic_name)

  <<cert>> Retrieves the DDS_DataWriter for a specific DDS_Topic.
- DDS_DomainParticipant * DDS_Publisher_get_participant (DDS_Publisher *self)

  <<cert>> Returns the DDS_DomainParticipant to which the DDS_Publisher belongs.
- DDS_Entity * DDS_DataWriter_as_entity (DDS_DataWriter *dataWriter)

  <<cert>> Access a DDS_DataWriter's supertype instance.
- DDS_ReturnCode_t DDS_DataWriter_assert_liveliness (DDS_DataWriter *self)

  <<cert>> This operation manually asserts the liveliness of this DDS_DataWriter.
- DDS_Topic * DDS_DataWriter_get_topic (DDS_DataWriter *self)

  <<cert>> This operation returns the DDS_Topic associated with the DDS_DataWriter.
- DDS_Publisher * DDS_DataWriter_get_publisher (DDS_DataWriter *self)

  <<cert>> This operation returns the DDS_Publisher to which the DDS_DataWriter belongs.
- DDS_ReturnCode_t DDS_DataWriter_get_liveliness_lost_status (DDS_DataWriter *self, struct DDS_LivelinessLostStatus *status)

  <<cert>> Accesses the DDS_LIVELINESS_LOST_STATUS communication status.
- DDS_ReturnCode_t DDS_DataWriter_get_offered_deadline_missed_status (DDS_DataWriter *self, struct DDS_OfferedDeadlineMissedStatus *status)

  <<cert>> Accesses the DDS_OFFERED_DEADLINE_MISSED_STATUS communication status.
- DDS_ReturnCode_t DDS_DataWriter_get_offered_incompatible_qos_status (DDS_DataWriter *self, struct DDS_OfferedIncompatibleQosStatus *status)

  <<cert>> Accesses the DDS_OFFERED_INCOMPATIBLE_QOS_STATUS communication status.
- DDS_ReturnCode_t DDS_DataWriter_get_publication_matched_status (DDS_DataWriter *self, struct DDS_PublicationMatchedStatus *status)

  <<cert>> Accesses the DDS_PUBLICATION_MATCHED_STATUS communication status.
- DDS_ReturnCode_t DDS_DataWriter_get_reliable_reader_activity_changed_status (DDS_DataWriter *self, struct DDS_ReliableReaderActivityChangedStatus *status)

  <<cert>> Get the reliable reader activity changed status for this writer.

Variables

- struct DDS_DataWriterQos DDS_DATAWRITER_QOS_DEFAULT

  <<cert>> Special value for creating DDS_DataWriter with default QoS

7.13.1 Detailed Description

DDS publication module.

7.14 dds_c_rh_plugin.h File Reference

DDS reader history interface.

Variables

- const char *const DDSHST_READER_DEFAULT_HISTORY_NAME

  Default reader history plug-in name.

7.14.1 Detailed Description

DDS reader history interface.

7.14.2 Variable Documentation

7.14.2.1 const char* const DDSHST_READER_DEFAULT_HISTORY_NAME

Default reader history plug-in name.

7.15 dds_c_sequence.h File Reference

DDS sequence definitions.

Data Structures

- struct FooSeq

  <<interface>> <<generic>> <<cert>> A type-safe, ordered collection of elements. The type of these elements is referred to in this documentation as "Foo".

Macros

- #define DDS_SEQUENCE_INITIALIZER

  An initializer for new sequence instances.

Functions

- DDS_Boolean FooSeq_initialize (struct FooSeq *self)

  <<cert>> Initialize sequence instances.
- DDS_Long FooSeq_get_maximum (const struct FooSeq *self)

  <<cert>> Get the current maximum number of elements that can be stored in this sequence.
- DDS_Boolean FooSeq_set_maximum (struct FooSeq *self, DDS_Long new_max)

<<cert>> Resize this sequence to a new desired maximum.

- DDS_Boolean FooSeq_set_maximum_w_max (struct FooSeq *self, DDS_Long new_max, DDS_UnsignedLong max_size)

  <<cert>> Resize this sequence to a new desired maximum and allocate `max_size` + 1 bytes for each pointer element.

- DDS_Long FooSeq_get_length (const struct FooSeq *self)

  <<cert>> Get the logical length of this sequence.

- DDS_Boolean FooSeq_set_length (struct FooSeq *self, DDS_Long new_length)

  <<cert>> Change the length of this sequence.

- Foo * FooSeq_get_reference (const struct FooSeq *self, DDS_Long i)

  <<cert>> Get the pointer to the `i-th` element of this sequence.

- struct FooSeq * FooSeq_copy (struct FooSeq *self, const struct FooSeq *src_seq)

  Copy elements from another sequence, resizing the sequence if necessary.

- struct FooSeq * FooSeq_copy_w_max (struct FooSeq *self, const struct FooSeq *src_seq, DDS_UnsignedLong max_size)

  Copy elements from another sequence, resizing the sequence if necessary.

7.15.1 Detailed Description

DDS sequence definitions.

7.16 dds_c_string.h File Reference

DDS string definitions.

Functions

- char * DDS_String_alloc (DDS_UnsignedLong length)

  Create a new empty string that can hold up to `length` characters.

- char * DDS_String_dup (const char *str)

  Clone a string. Creates a new string that duplicates the value of `string`.

- int DDS_String_cmp (const char *s1, const char *s2)

  Compare two strings.

- int DDS_String_ncmp (const char *s1, const char *s2, DDS_UnsignedLong num)

  Compare two strings.

- int DDS_String_length (const char *string)

  Return the length of a string.

- int DDS_Wstring_cmp (const CDR_Wstring *s1, const CDR_Wstring *s2)

  Return the length of a string.

7.16.1 Detailed Description

DDS string definitions.

7.17 dds_c_subscription.h File Reference

DDS subscription module.

Data Structures

- struct DDS_RequestedDeadlineMissedStatus
  <<cert>> DDS_REQUESTED_DEADLINE_MISSED_STATUS
- struct DDS_LivelinessChangedStatus
  <<cert>> DDS_LIVELINESS_CHANGED_STATUS
- struct DDS_RequestedIncompatibleQosStatus
  <<cert>> DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS
- struct DDS_SampleRejectedStatus
  DDS_SAMPLE_REJECTED_STATUS
- struct DDS_DataReaderInstanceReplacedStatus
  <<eXtension>> <<cert>> DDS_INSTANCE_REPLACED_STATUS.
- struct DDS_SubscriptionMatchedStatus
  <<cert>> DDS_SUBSCRIPTION_MATCHED_STATUS
- struct DDS_SampleInfo
  Information that accompanies each sample that is `read` *or* `taken`*.*
- struct DDS_SampleInfoSeq
  Declares IDL `sequence` *<* *DDS_SampleInfo* *>* *.*
- struct DDS_SampleLostStatus
  <<cert>> DDS_SAMPLE_LOST_STATUS
- struct DDS_DataReaderQos
  <<cert>> QoS policies supported by a DDS_DataReader entity.
- struct DDS_SubscriberQos
  <<cert>> QoS policies supported by a DDS_Subscriber entity.
- struct DDS_DataReaderListener
  <<interface>> <<cert>> DDS_Listener for reader status.
- struct DDS_SubscriberListener
  <<interface>> <<cert>> DDS_Listener for status about a subscriber.

Macros

- #define DDS_RequestedDeadlineMissedStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_LivelinessChangedStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_RequestedIncompatibleQosStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_SampleRejectedStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_DataReaderInstanceReplacedStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_SubscriptionMatchedStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_SampleLostStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_DataReaderQos_INITIALIZER
  <<cert>> Initializer for new QoS instances.
- #define DDS_SubscriberQos_INITIALIZER
  <<cert>> Initializer for new QoS instances.
- #define DDS_DataReaderListener_INITIALIZER
  <<cert>> Initializer for new DDS_DataReaderListener.
- #define DDS_SubscriberListener_INITIALIZER
  <<eXtension>> <<cert>> Initializer for new DDS_SubscriberListener.

Typedefs

- typedef struct DDS_DataReaderImpl DDS_DataReader

  <<interface>> <<cert>> Allows the application to: (1) declare the data it wishes to receive (i.e. make a subscription) and (2) access the data received by the attached DDS_Subscriber.

- typedef struct DDS_SubscriberImpl DDS_Subscriber

  <<interface>> A subscriber is the object responsible for actually receiving data from a subscription.

- typedef DDS_UnsignedLong DDS_SampleStateMask

  A bit-mask (list) of sample states, i.e. DDS_SampleStateKind.

- typedef DDS_UnsignedLong DDS_ViewStateMask

  A bit-mask (list) of view states, i.e. DDS_ViewStateKind.

- typedef DDS_UnsignedLong DDS_InstanceStateMask

  A bit-mask (list) of instance states, i.e. DDS_InstanceStateKind.

- typedef void(* DDS_DataReaderListener_RequestedDeadlineMissedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_RequestedDeadlineMissedStatus *status)

  <<cert>> Prototype of a DDS_DataReaderListener requested_deadline_missed function.

- typedef void(* DDS_DataReaderListener_LivelinessChangedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_LivelinessChangedStatus *status)

  <<cert>> Prototype of a DDS_DataReaderListener liveliness_changed function.

- typedef void(* DDS_DataReaderListener_RequestedIncompatibleQosCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_RequestedIncompatibleQosStatus *status)

  <<cert>> Prototype of a DDS_DataReaderListener requested_incompatible_QoS function.

- typedef void(* DDS_DataReaderListener_SampleRejectedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SampleRejectedStatus *status)

  <<cert>> Prototype of a DDS_DataReaderListener sample_rejected function.

- typedef void(* DDS_DataReaderListener_DataAvailableCallback)(void *listener_data, DDS_DataReader *reader)

  <<cert>> Prototype of a DDS_DataReaderListener data_available function.

- typedef void(* DDS_DataReaderListener_SubscriptionMatchedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SubscriptionMatchedStatus *status)

  <<cert>> Prototype of a DDS_DataReaderListener subscription_match function.

- typedef void(* DDS_DataReaderListener_SampleLostCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_SampleLostStatus *status)

  <<cert>> Prototype of a DDS_DataReaderListener subscription_lost function.

- typedef void(* DDS_DataReaderListener_InstanceReplacedCallback)(void *listener_data, DDS_DataReader *reader, const struct DDS_DataReaderInstanceReplacedStatus *status)

  <<cert>> Prototype of a DDS_DataReaderListener instance_replaced function.

- typedef DDS_Boolean(* DDS_DataReaderListener_BeforeSampleDeserializeCallback)(void *listener_data, DDS_DataReader *reader, struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_Boolean *dropped)

  <<cert>> Prototype of a DDS_DataReaderListener before_sample_deserialize function.

- typedef DDS_Boolean(* DDS_DataReaderListener_BeforeSampleCommitCallback)(void *listener_data, DDS_DataReader *reader, const void *const sample, DDS_Boolean *dropped)

  <<cert>> Prototype of a DDS_DataReaderListener before_sample_commit function.

- typedef void(* DDS_SubscriberListener_DataOnReadersCallback)(void *listener_data, DDS_Subscriber *sub)

  Prototype of a DDS_SubscriberListener data_on_readers function.

Enumerations

- enum DDS_SampleRejectedStatusKind { DDS_NOT_REJECTED, DDS_REJECTED_BY_INSTANCES_LIMIT, DDS_REJECTED_BY_SAMPLES_LIMIT, DDS_REJECTED_BY_SAMPLES_PER_INSTANCE_LIMIT, DDS_REJECTED_BY_REMOTE_WRITERS_LIMIT, DDS_REJECTED_BY_SAMPLES_PER_REMOTE_WRITER_LIMIT }

  Kinds of reasons for rejecting a sample.
- enum DDS_SampleStateKind { DDS_READ_SAMPLE_STATE = 0x0001 << 0, DDS_NOT_READ_SAMPLE_STATE = 0x0001 << 1 }

  Indicates whether or not a sample has ever been read.
- enum DDS_ViewStateKind { DDS_NEW_VIEW_STATE = 0x0001 << 0, DDS_NOT_NEW_VIEW_STATE = 0x0001 << 1 }

  Indicates whether or not an instance is new.
- enum DDS_InstanceStateKind { DDS_ALIVE_INSTANCE_STATE = 0x0001 << 0, DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE = 0x0001 << 1, DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE = 0x0001 << 2 }

  Indicates is the samples are from a live DDS_DataWriter or not.
- enum DDS_SampleLostStatusKind { DDS_SAMPLE_LOST_NOT_LOST, DDS_SAMPLE_LOST_BY_DATAWRITER, DDS_SAMPLE_LOST_BY_MAX_SAMPLES_LIMIT, DDS_SAMPLE_LOST_BY_HISTORY_DEPTH_LIMIT, DDS_SAMPLE_LOST_BY_META_SAMPLE_LIMIT, DDS_SAMPLE_LOST_BY_NOT_READ_ON_CACHE_DELETION, DDS_SAMPLE_LOST_BY_MANAGEDPOOL_OVERWRITE }

  Kinds of reasons a sample was lost.

Functions

- DDS_ReturnCode_t DDS_DataReaderQos_initialize (struct DDS_DataReaderQos *self)

  Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_DataReaderQos_copy (struct DDS_DataReaderQos *self, const struct DDS_DataReaderQos *source)

  Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_DataReaderQos_is_equal (const struct DDS_DataReaderQos *left, const struct DDS_DataReaderQos *right)

  Compare two DDS_DataReaderQos policies for equality.
- DDS_ReturnCode_t DDS_SubscriberQos_initialize (struct DDS_SubscriberQos *self)

  Initializer for new QoS instances.
- DDS_ReturnCode_t DDS_SubscriberQos_copy (struct DDS_SubscriberQos *self, const struct DDS_SubscriberQos *source)

  Copy the contents of the given QoS into this QoS.
- DDS_Boolean DDS_SubscriberQos_is_equal (const struct DDS_SubscriberQos *left, const struct DDS_SubscriberQos *right)

  Compare two DDS_SubscriberQos policies for equality.
- DDS_Entity * DDS_Subscriber_as_entity (DDS_Subscriber *subscriber)

  <<cert>> Access a DDS_Subscriber's supertype instance.
- DDS_ReturnCode_t DDS_Subscriber_get_default_datareader_qos (DDS_Subscriber *self, struct DDS_DataReaderQos *qos)

  Copies the default DDS_DataReaderQos values into the provided DDS_DataReaderQos instance.
- DDS_ReturnCode_t DDS_Subscriber_set_default_datareader_qos (DDS_Subscriber *self, const struct DDS_DataReaderQos *qos)

  Sets the default DDS_DataReaderQos values for this subscriber.
- DDS_DataReader * DDS_Subscriber_create_datareader (DDS_Subscriber *self, DDS_TopicDescription *topic, const struct DDS_DataReaderQos *qos, const struct DDS_DataReaderListener *listener, DDS_StatusMask mask)

<<cert>> *Creates a DDS_DataReader that will be attached and belong to the DDS_Subscriber.*

- DDS_DataReader * DDS_Subscriber_lookup_datareader (DDS_Subscriber *self, const char *topic_name)

  <<cert>> *Retrieves an existing DDS_DataReader.*
- DDS_DomainParticipant * DDS_Subscriber_get_participant (DDS_Subscriber *self)

  <<cert>> *Returns the DDS_DomainParticipant to which the DDS_Subscriber belongs.*
- DDS_Entity * DDS_DataReader_as_entity (DDS_DataReader *self)

  <<cert>> *Accesses the DDS_DataReader's supertype instance.*
- DDS_TopicDescription * DDS_DataReader_get_topicdescription (DDS_DataReader *self)

  <<cert>> *Returns the DDS_TopicDescription associated with the DDS_DataReader.*
- DDS_Subscriber * DDS_DataReader_get_subscriber (DDS_DataReader *self)

  <<cert>> *Returns the DDS_Subscriber to which the DDS_DataReader belongs.*
- DDS_ReturnCode_t DDS_DataReader_get_sample_rejected_status (DDS_DataReader *self, struct DDS_SampleRejectedStatus *status)

  <<cert>> *Accesses the DDS_SAMPLE_REJECTED_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataReader_get_liveliness_changed_status (DDS_DataReader *self, struct DDS_LivelinessChangedStatus *status)

  <<cert>> *Accesses the DDS_LIVELINESS_CHANGED_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataReader_get_requested_deadline_missed_status (DDS_DataReader *self, struct DDS_RequestedDeadlineMissedStatus *status)

  <<cert>> *Accesses the DDS_REQUESTED_DEADLINE_MISSED_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataReader_get_requested_incompatible_qos_status (DDS_DataReader *self, struct DDS_RequestedIncompatibleQosStatus *status)

  <<cert>> *Accesses the DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataReader_get_subscription_matched_status (DDS_DataReader *self, struct DDS_SubscriptionMatchedStatus *status)

  <<cert>> *Accesses the DDS_SUBSCRIPTION_MATCHED_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataReader_get_sample_lost_status (DDS_DataReader *self, struct DDS_SampleLostStatus *status)

  <<cert>> *Accesses the DDS_SAMPLE_LOST_STATUS communication status.*
- DDS_ReturnCode_t DDS_DataReader_get_instance_replaced_status (DDS_DataReader *self, struct DDS_DataReaderInstanceReplacedStatus *status)

  <<cert>> *Accesses the DDS_INSTANCE_REPLACED_STATUS communication status.*

Variables

- const DDS_SampleStateMask DDS_ANY_SAMPLE_STATE

  Any sample state DDS_READ_SAMPLE_STATE | DDS_NOT_READ_SAMPLE_STATE.
- const DDS_ViewStateMask DDS_ANY_VIEW_STATE

  Any view state DDS_NEW_VIEW_STATE | DDS_NOT_NEW_VIEW_STATE.
- const DDS_InstanceStateMask DDS_ANY_INSTANCE_STATE

  Any instance state ALIVE_INSTANCE_STATE | NOT_ALIVE_DISPOSED_INSTANCE_STATE | NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.
- const DDS_InstanceStateMask DDS_NOT_ALIVE_INSTANCE_STATE

  Not alive instance state NOT_ALIVE_DISPOSED_INSTANCE_STATE | NOT_ALIVE_NO_WRITERS_INSTANCE_STATE.
- struct DDS_DataReaderQos DDS_DATAREADER_QOS_DEFAULT

  <<cert>> *Special value for creating data reader with default QoS.*

7.17.1 Detailed Description

DDS subscription module.

7.18 dds_c_topic.h File Reference

DDS topic module.

Data Structures

- struct DDS_InconsistentTopicStatus
  DDS_INCONSISTENT_TOPIC_STATUS
- struct DDS_TopicQos
  QoS policies supported by a DDS_Topic entity.
- struct DDS_TopicListener
  <<interface>> DDS_Listener for DDS_Topic entities.

Macros

- #define DDS_InconsistentTopicStatus_INITIALIZER
  <<cert>> Initializer for new status instances.
- #define DDS_TopicListener_INITIALIZER
  <<cert>> Initializer for new DDS_TopicListener.

Typedefs

- typedef struct DDS_BuiltinTopicKey_t DDS_InstanceId_t
  The data structure by which RTI Connext Micro identifies keys in user data types.
- typedef struct DDS_TopicDescriptionImpl DDS_TopicDescription
  <<interface>> Base class for DDS_Topic.
- typedef struct DDS_TopicImpl DDS_Topic
  <<interface>> The most basic description of the data to be published and subscribed.
- typedef void(* DDS_TopicListener_InconsistentTopicCallback)(void *listener_data, DDS_Topic *topic, const struct DDS_InconsistentTopicStatus *status)
  Prototype of DDS_TopicListener::on_inconsistent_topic.

Functions

- const char * DDS_TopicDescription_get_type_name (DDS_TopicDescription *self)
  <<cert>> Get the associated `type_name`.
- const char * DDS_TopicDescription_get_name (DDS_TopicDescription *self)
  <<cert>> Get the name used to create this DDS_TopicDescription.
- DDS_DomainParticipant * DDS_TopicDescription_get_participant (DDS_TopicDescription *self)
  <<cert>> Get the DDS_DomainParticipant to which the DDS_TopicDescription belongs.
- DDS_Entity * DDS_Topic_as_entity (DDS_Topic *self)
  <<cert>> Access a DDS_Topic's DDS_Entity supertype instance.
- DDS_TopicDescription * DDS_Topic_as_topicdescription (DDS_Topic *self)
  <<cert>> Access a DDS_Topic's DDS_TopicDescription supertype instance.
- DDS_Topic * DDS_Topic_narrow (DDS_TopicDescription *self)
  <<cert>> Narrow the given DDS_TopicDescription pointer to a DDS_Topic pointer.
- DDS_ReturnCode_t DDS_Topic_get_inconsistent_topic_status (DDS_Topic *self, struct DDS_InconsistentTopicStatus *status)
  <<cert>> Allows the application to retrieve the DDS_INCONSISTENT_TOPIC_STATUS status of a DDS_Topic.

7.18.1 Detailed Description

DDS topic module.

7.19 dds_c_type.h File Reference

DDS type module.

Data Structures

- struct NDDS_Type_PluginVersion

  Data type used to keep track of versioning information about a NDDS_Type_Plugin.
- struct NDDS_Type_Plugin

  <<interface>> Type plugin interface.

Typedefs

- typedef struct NDDS_Type_PluginVersion NDDS_Type_PluginVersion

  Data type used to keep track of versioning information about a NDDS_Type_Plugin.
- typedef RTI_UINT32(* NDDS_Type_GetSerializedSizeFunc)(struct NDDS_Type_Plugin *plugin, RTI_UINT32 current_alignment, void *param)

  Prototype of the function to determine the size of a data sample in serialized form.
- typedef RTI_BOOL(* NDDS_Type_CreateSampleFunc)(struct NDDS_Type_Plugin *plugin, void **sample, void *param)

  Prototype of the data sample creation function.
- typedef RTI_BOOL(* NDDS_Type_DeleteSampleFunc)(struct NDDS_Type_Plugin *plugin, void *sample, void *param)

  Prototype of the data sample delete function.
- typedef RTI_BOOL(* NDDS_Type_CopySampleFunc)(struct NDDS_Type_Plugin *plugin, void *dst, const void *src, void *param)

  Prototype of the data sample copy function.
- typedef NDDS_TypePluginKeyKind(* NDDS_PluginHelper_GetKeyKindFunc)(struct NDDS_Type_Plugin *plugin, void *param)

  Prototype of the keyed data check function.
- typedef RTI_BOOL(* NDDS_PluginHelper_InstanceToKeyHashFunc)(struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_KeyHash_t *keyHash, const void *instance, void *param)

  Prototype of the function for converting an instance of data into a 16-byte key hash.
- typedef void *(* NDDS_PluginHelper_CreateTypedDataWriterFunc)(void *writer)

  Prototype of the function for creating a language-specific wrapper for a DDS_DataWriter created by the C core layer.
- typedef void(* NDDS_PluginHelper_DeleteTypedDataWriterFunc)(void *wrapper)

  Prototype of the function for deleting a language-specific wrapper of a DDS_DataReader created by the C core layer.
- typedef void *(* NDDS_PluginHelper_CreateTypedDataReaderFunc)(void *reader)

  Prototype of the function for creating a language-specific wrapper for a DDS_DataReader created by the C core layer.
- typedef void(* NDDS_PluginHelper_DeleteTypedDataReaderFunc)(void *wrapper)

  Prototype of the function for deleting a language-specific wrapper of a DDS_DataReader created by the C core layer.
- typedef DDS_ReturnCode_t(* NDDS_PluginHelper_ManagedPoolGetLoanFunc)(void *managed_pool, void **sample_out)

  Prototype of the function for getting a loan from a pool of managed samples.
- typedef RTI_BOOL(* NDDS_PluginHelper_ManagedPoolDiscardLoanFunc)(void *managed_pool, RTI_UINT32 sample_id)

Prototype of the function for discarding a loan and returning a it back to a pool of managed samples.

- typedef RTI_BOOL(* NDDS_PluginHelper_CreateManagedPoolFunc)(struct NDDS_Type_Plugin *plugin, void *datawriter, void *param, void **managed_pool_out)

  Prototype of the function for creating a pool of samples.
- typedef void(* NDDS_PluginHelper_DeleteManagedPoolFunc)(void *managed_pool)

  Prototype of the function for deleting a pool of samples.
- typedef RTI_BOOL(* NDDS_PluginHelper_ManagedPoolGetSampleIDFunc)(void *managed_pool, const void *user_data, RTI_UINT32 *sample_id_out)

  Prototype of the function to getting an internal id associated with a sample loaned from the managed pool.
- typedef RTI_BOOL(* NDDS_PluginHelper_NeedsOpaqueSamples)(void *reader, void *param)

  Prototype of the function to be called by the DDS_DataReader to figure out if it needs to create opaque samples.

Enumerations

- enum NDDS_TypePluginKeyKind { NDDS_TYPEPLUGIN_NO_KEY , NDDS_TYPEPLUGIN_USER_KEY }

  Type key kind: either keyed or unkeyed.

Functions

- RTI_BOOL PluginHelper_instance_to_keyhash (struct NDDS_Type_Plugin *plugin, struct CDR_Stream_t *stream, DDS_KeyHash_t *keyHash, const void *instance, void *param)

  Function to use a NDDS_Type_Plugin to create a 16-byte keyhash from an instance of data.
- NDDS_TypePluginKeyKind PluginHelper_get_key_kind (struct NDDS_Type_Plugin *plugin, void *param)

  Function to determine the type of key used by a NDDS_Type_Plugin's data type.

7.19.1 Detailed Description

DDS type module.

7.20 dds_c_wh_plugin.h File Reference

DDS writer history interface.

Variables

- const char *const DDSHST_WRITER_DEFAULT_HISTORY_NAME

  Default writer history plug-in name.

7.20.1 Detailed Description

DDS writer history interface.

7.20.2 Variable Documentation

7.20.2.1 const char* const DDSHST_WRITER_DEFAULT_HISTORY_NAME

Default writer history plug-in name.

7.21 disc_dpse_dll.h File Reference

DPSE DLL external linkage.

7.21.1 Detailed Description

DPSE DLL external linkage.

7.22 disc_dpse_dpsediscovery.h File Reference

DPSE API Dynamic Participant/Static Endpoint Discovery Plugin.

Data Structures

- struct DPSE_DiscoveryPluginProperty

  <<eXtension>> <<cert>> Properties for the Dynamic Participant/Static Endpoint (DPSE) discovery plugin. This includes all discovery timing properties for participant discovery.

Macros

- #define DPSE_DiscoveryPluginProperty_INITIALIZER

  Initializer for the DPSE_DiscoveryPluginProperty.

Functions

- DDS_ReturnCode_t DPSE_DiscoveryPluginProperty_initialize (struct DPSE_DiscoveryPluginProperty *self)

  Initialize a DiscoveryPluginProperty structure.
- DDS_ReturnCode_t DPSE_RemoteParticipant_assert (DDS_DomainParticipant *const participant, const char *rem_participant_name)

  <<cert>> Assert remote participant by passing in the name of that participant.
- DDS_ReturnCode_t DPSE_RemotePublication_assert (DDS_DomainParticipant *const participant, const char *const rem_participant_name, const struct DDS_PublicationBuiltinTopicData *const data, NDDS_TypePluginKeyKind key_kind)

  <<cert>> Assert remote publication by passing in the builtin data that describes the type, topic, and QoS of that remote publication.
- DDS_ReturnCode_t DPSE_RemoteSubscription_assert (DDS_DomainParticipant *const participant, const char *const rem_participant_name, const struct DDS_SubscriptionBuiltinTopicData *const data, NDDS_TypePluginKeyKind key_kind)

  <<cert>> Assert remote subscription by passing in the builtin data that describes the type, topic, and QoS of that remote subscription.
- struct RT_ComponentFactoryI * DPSE_DiscoveryFactory_get_interface (void)

  Gets the singleton instance of the DPSE discovery plugin factory. The singleton instance of the factory is used when creating a domain participant.

7.22.1 Detailed Description

DPSE API Dynamic Participant/Static Endpoint Discovery Plugin. The Plugin APIs are used to configure and instantiate the RTI Connext Micro Dynamic Participant/Static Endpoint Discovery Plugin.

7.23 disc_dpse_log.h File Reference

Macros

- #define DPSE_LOG_INVALID_PEER_ADDRESS_EC (DPSE_LOG_BASE + 1)

  A peer address string specifies an invalid address.

- #define DPSE_LOG_CREATE_DISCOVERY_PUBLISHER_EC (DPSE_LOG_BASE + 2)

  Failed to create the publisher for a participant discovery writer.
- #define DPSE_LOG_CREATE_DISCOVERY_SUBSCRIBER_EC (DPSE_LOG_BASE + 3)

  Failed to create the subscriber for a participant discovery datareader.
- #define DPSE_LOG_REGISTER_TYPE_EC (DPSE_LOG_BASE + 4)

  Failed to register a built-in participant discovery type.
- #define DPSE_LOG_CREATE_TOPIC_EC (DPSE_LOG_BASE + 5)

  Failed to create a topic for the built-in participant discovery topic.
- #define DPSE_LOG_CREATE_WRITER_EC (DPSE_LOG_BASE + 6)

  Failed to create a DataWriter for participant discovery.
- #define DPSE_LOG_CREATE_READER_EC (DPSE_LOG_BASE + 7)

  Failed to create a DataReader for participant discovery.
- #define DPSE_LOG_DISPOSE_EC (DPSE_LOG_BASE + 8)

  Failed to dispose a participant discovery instance.
- #define DPSE_LOG_ANNOUNCE_LOCAL_PARTICIPANT_DELETION_EC (DPSE_LOG_BASE + 9)

  Failed to dispose a participant discovery instance upon deletion.
- #define DPSE_LOG_DELETE_WRITER_EC (DPSE_LOG_BASE + 10)

  Failed to delete a participant discovery DataWriter.
- #define DPSE_LOG_DELETE_PUBLISHER_EC (DPSE_LOG_BASE + 11)

  Failed to delete a participant discovery Publisher.
- #define DPSE_LOG_DELETE_READER_EC (DPSE_LOG_BASE + 12)

  Failed to delete a participant discovery DataReader.
- #define DPSE_LOG_DELETE_TOPIC_EC (DPSE_LOG_BASE + 13)

  Failed to delete a participant discovery Topic.
- #define DPSE_LOG_DELETE_SUBSCRIBER_EC (DPSE_LOG_BASE + 14)

  Failed to delete a participant discovery Subscriber.
- #define DPSE_KEYLIST_OBJECT (1)

  The list keeping track of asserted participant keys.
- #define DPSE_PARTICIPANTMAMES_OBJECT (2)

  The list keeping track of asserted participant names.
- #define DPSE_PARTICIPANTNAMEINDEX_OBJECT (3)

  The index keeping track of asserted participant names.
- #define DPSE_PARTICIPANTANNOUCEMENT_OBJECT (4)

  The participant announcement timeout object.
- #define DPSE_DISCOVERYPLUGIN_OBJECT (5)

  A discovery plugin object.
- #define DPSE_PARTICIPANTBUILTINTOPICDATA_OBJECT (6)

  ParticipantBuiltinTopicData.
- #define DPSE_PARTICIPANTENTITYNAME_OBJECT (7)

  ParticipantBuiltinTopicData entity-name object.
- #define DPSE_ENABLEDPARTICIPANTINDEX_OBJECT (8)

  The index keeping track of enabled participant names.
- #define DPSE_PARTICIPANTPOOL_OBJECT (10)

  The pool of asserted participants.
- #define DPSE_LOG_OBJECT_ALLOCATE_EC (DPSE_LOG_BASE + 15)

  Failed to allocate an object of the specified kind.
- #define DPSE_LOG_OBJECT_INITIALIZE_EC (DPSE_LOG_BASE + 16)

  Failed to initialize an object of the specified kind.
- #define DPSE_LOG_OBJECT_FINALIZE_EC (DPSE_LOG_BASE + 17)

  Failed to finalize an object of the specified kind.
- #define DPSE_LOG_OBJECT_DELETE_EC (DPSE_LOG_BASE + 18)

Failed to delete an object of the specified kind.

- #define DPSE_LOG_OBJECT_INVALID_EC (DPSE_LOG_BASE + 19)

  The object was invalid in the context it was used.
- #define DPSE_LOG_CDR_SET_POSITION_EC (DPSE_LOG_BASE + 20)

  Failed to set the CDR stream position.
- #define DPSE_LOG_CDR_LONG_KIND 1

  CDR Long kind.
- #define DPSE_LOG_CDR_PID_KIND 2

  CDR PID kind, the pid argument specifies which pid.
- #define DPSE_LOG_CDR_SERIALIZE_EC (DPSE_LOG_BASE + 21)

  Failed to serialize the specified kind.
- #define DPSE_LOG_CDR_DESERIALIZE_EC (DPSE_LOG_BASE + 22)

  Failed to deserialize the specified kind.
- #define DPSE_LOG_SERIALIZE_GUID_EC (DPSE_LOG_BASE + 23)

  Failed to serialize a GUID key parameter.
- #define DPSE_LOG_SERIALIZE_BUILTIN_ENDPOINTS_EC (DPSE_LOG_BASE + 24)

  Failed to serialize a Builtin Endpoint Mask parameter.
- #define DPSE_LOG_SERIALIZE_PROTOCOL_VERSION_EC (DPSE_LOG_BASE + 25)

  Failed to serialize a Protocol Version parameter.
- #define DPSE_LOG_SERIALIZE_VENDOR_ID_EC (DPSE_LOG_BASE + 26)

  Failed to serialize a Vendor ID parameter.
- #define DPSE_LOG_SERIALIZE_DEFAULT_UNICAST_EC (DPSE_LOG_BASE + 27)

  Failed to serialize a Default Unicast Locator parameter.
- #define DPSE_LOG_SERIALIZE_META_UNICAST_EC (DPSE_LOG_BASE + 28)

  Failed to serialize a Meta Unicast Locator parameter.
- #define DPSE_LOG_SERIALIZE_META_MULTICAST_EC (DPSE_LOG_BASE + 29)

  Failed to serialize a Meta Multicast Locator parameter.
- #define DPSE_LOG_SERIALIZE_LEASE_DURATION_EC (DPSE_LOG_BASE + 30)

  Failed to serialize a Lease Duration parameter.
- #define DPSE_LOG_SERIALIZE_PRODUCT_VERSION_EC (DPSE_LOG_BASE + 31)

  Failed to serialize a Product Version parameter.
- #define DPSE_LOG_DESERIALIZE_PRODUCT_VERSION_EC (DPSE_LOG_BASE + 32)

  Failed to serialize a Product Version parameter.
- #define DPSE_LOG_SERIALIZE_PARTICIPANT_NAME_EC (DPSE_LOG_BASE + 33)

  Failed to serialize a Participant Name parameter.
- #define DPSE_LOG_DESERIALIZE_GUID_EC (DPSE_LOG_BASE + 34)

  Failed to deserialize a GUID key parameter.
- #define DPSE_LOG_DESERIALIZE_BUILTIN_ENDPOINTS_EC (DPSE_LOG_BASE + 35)

  Failed to deserialize a Builtin Endpoint Mask parameter.
- #define DPSE_LOG_DESERIALIZE_PROTOCOL_VERSION_EC (DPSE_LOG_BASE + 36)

  Failed to deserialize a Protocol Version parameter.
- #define DPSE_LOG_DESERIALIZE_VENDOR_ID_EC (DPSE_LOG_BASE + 37)

  Failed to deserialize a Vendor ID parameter.
- #define DPSE_LOG_DESERIALIZE_TOO_MANY_LOCATORS_EC (DPSE_LOG_BASE + 38)

  Cannot deserialize another locator parameter, having reached maximum of 4 unicast or 4 multicast locators.
- #define DPSE_LOG_DESERIALIZE_PARTICIPANT_NAME_EC (DPSE_LOG_BASE + 39)

  Failed to deserialize a Participant Name parameter.
- #define DPSE_LOG_DESERIALIZE_UNKNOWN_PID_EC (DPSE_LOG_BASE + 40)

  Failed to deserialize a parameter with an unknown ID.
- #define DPSE_LOG_ANNOUNCEMENT_EC (DPSE_LOG_BASE + 41)

  Failed to send a participant discovery announcement.

- #define DPSE_LOG_UPDATE_PARTICIPANT_ASSERT_PERIOD_EC (DPSE_LOG_BASE + 42)

  Failed to schedule an event to send the next participant discovery announcement.
- #define DPSE_LOG_ADVANCE_SN_EC (DPSE_LOG_BASE + 43)

  Failed to advance the sequence number of a participant discovery announcement.
- #define DPSE_LOG_ANNOUNCE_WRITE_EC (DPSE_LOG_BASE + 44)

  Failed to write a participant discovery announcement message.
- #define DPSE_LOG_SCHEDULE_FAST_ASSERTION_EC (DPSE_LOG_BASE + 45)

  Failed to schedule an event to send the next participant discovery announcment.
- #define DPSE_LOG_PARTICIPANT_TAKE_EC (DPSE_LOG_BASE + 46)

  Failed to take a sample from a participant discovery DataReader.
- #define DPSE_LOG_ASSERT_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 47)

  Failed to assert remote participant.
- #define DPSE_LOG_ON_ASSERT_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 48)

  Failed to assert and complete discovery of a remote participant.
- #define DPSE_LOG_ADD_ANONYMOUS_ROUTE_EC (DPSE_LOG_BASE + 49)

  Failed to add an anonymous route.
- #define DPSE_LOG_DELETE_ANONYMOUS_ROUTE_EC (DPSE_LOG_BASE + 50)

  Failed to delete an anonymous route.
- #define DPSE_LOG_MAX_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 51)

  Exceeded resource limit, remote_participant_allocation.
- #define DPSE_LOG_REFRESH_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 52)

  Failed to refresh liveliness for a remote participant.
- #define DPSE_LOG_RESET_REMOTE_PARTICIPANT_EC (DPSE_LOG_BASE + 53)

  Failed to reset liveliness for a remote participant.
- #define DPSE_LOG_INVALID_DISCOVERY_SAMPLE_EC (DPSE_LOG_BASE + 54)

  Received a participant discovery announcement with invalid state.
- #define DPSE_LOG_RETURN_DISCOVERY_SAMPLE_EC (DPSE_LOG_BASE + 55)

  Failed to return a loan on a participant discovery announcement.
- #define DPSE_LOG_SERIALIZE_MULTICAST_EC (DPSE_LOG_BASE + 56)

  Failed to serialize a Multicast Locator parameter.
- #define DPSE_LOG_GET_DDS_PROPERTIES_EC (DPSE_LOG_BASE + 57)

  Failed to serialize a Multicast Locator parameter.
- #define DPSE_TOPIC_ENTITY 1

  A DDS Topic entity.
- #define DPSE_PUBLISHER_ENTITY 2

  A DDS Publisher entity.
- #define DPSE_SUBSCRIBER_ENTITY 3

  A DDS Subscriber entity.
- #define DPSE_LOG_ENTITY_ENABLE_EC (DPSE_LOG_BASE + 58)

  Failed to enable the specified entity kind.
- #define DPSE_LOG_DEFAULT_UNICAST_LOCATOR_SEQUENCE 1

  The default unicast locator sequence.
- #define DPSE_LOG_DEFAULT_MULTICAST_LOCATOR_SEQUENCE 2

  The default multicast locator sequence.
- #define DPSE_LOG_META_UNICAST_LOCATOR_SEQUENCE 3

  The meta unicast locator sequence.
- #define DPSE_LOG_META_MULTICAST_LOCATOR_SEQUENCE 4

  The meta multicast locator sequence.
- #define DPSE_LOG_SEQUENCE_SETMAX_EC (DPSE_LOG_BASE + 59)

  Failed to set the maximum length of a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_SETLENGTH_EC (DPSE_LOG_BASE + 60)

Failed to set the length of a sequence of the specified kind.

- #define DPSE_LOG_SEQUENCE_GETREF_EC (DPSE_LOG_BASE + 61)

  Failed to get a reference at the specified index for a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_INITIALIZE_EC (DPSE_LOG_BASE + 62)

  Failed to initialize a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_FINALIZE_EC (DPSE_LOG_BASE + 63)

  Failed to finalize a sequence of the specified kind.
- #define DPSE_LOG_SEQUENCE_COPY_EC (DPSE_LOG_BASE + 64)

  Failed to copy a sequence of the specified kind.
- #define DPSE_LOG_GET_TIMER_EC (DPSE_LOG_BASE + 65)

  Failed to get timer.
- #define DPSE_LOG_UNSUPPORTED_LOCATOR_EC (DPSE_LOG_BASE + 66)

  The locator kind is not supported by any transport registered with the domain participant and is dropped.
- #define DPSE_LOG_CHECKSUM_INCOMPATIBLE_PARTICIPANT_IGNORED_EC (DPDE_LOG_BASE + 67)

  A participant with incompatible checksum was discovered and ignored.

7.23.1 Detailed Description

7.23.2 Macro Definition Documentation

7.23.2.1 #define DPSE_KEYLIST_OBJECT (1)

The list keeping track of asserted participant keys.

7.23.2.2 #define DPSE_PARTICIPANTMAMES_OBJECT (2)

The list keeping track of asserted participant names.

7.23.2.3 #define DPSE_PARTICIPANTNAMEINDEX_OBJECT (3)

The index keeping track of asserted participant names.

7.23.2.4 #define DPSE_PARTICIPANTANNOUCEMENT_OBJECT (4)

The participant announcement timeout object.

7.23.2.5 #define DPSE_DISCOVERYPLUGIN_OBJECT (5)

A discovery plugin object.

7.23.2.6 #define DPSE_PARTICIPANTBUILTINTOPICDATA_OBJECT (6)

ParticipantBuiltinTopicData.

7.23.2.7 #define DPSE_PARTICIPANTENTITYNAME_OBJECT (7)

ParticipantBuiltinTopicData entity-name object.

7.23.2.8 #define DPSE_ENABLEDPARTICIPANTINDEX_OBJECT (8)

The index keeping track of enabled participant names.

7.23.2.9 #define DPSE_PARTICIPANTPOOL_OBJECT (10)

The pool of asserted participants.

7.23.2.10 #define DPSE_LOG_CDR_LONG_KIND 1

CDR Long kind.

7.23.2.11 #define DPSE_LOG_CDR_PID_KIND 2

CDR PID kind, the pid argument specifies which pid.

7.23.2.12 #define DPSE_TOPIC_ENTITY 1

A DDS Topic entity.

7.23.2.13 #define DPSE_PUBLISHER_ENTITY 2

A DDS Publisher entity.

7.23.2.14 #define DPSE_SUBSCRIBER_ENTITY 3

A DDS Subscriber entity.

7.23.2.15 #define DPSE_LOG_DEFAULT_UNICAST_LOCATOR_SEQUENCE 1

The default unicast locator sequence.

7.23.2.16 #define DPSE_LOG_DEFAULT_MULTICAST_LOCATOR_SEQUENCE 2

The default multicast locator sequence.

7.23.2.17 #define DPSE_LOG_META_UNICAST_LOCATOR_SEQUENCE 3

The meta unicast locator sequence.

7.23.2.18 #define DPSE_LOG_META_MULTICAST_LOCATOR_SEQUENCE 4

The meta multicast locator sequence.

7.24 netio_common.h File Reference

7.24.1 Detailed Description

7.25 netio_config.h File Reference

7.25.1 Detailed Description

7.26 netio_dll.h File Reference

7.26.1 Detailed Description

7.27 netio_interface.h File Reference

Generic NETIO_Interface functions.

Data Structures

- struct NETIO_Packet

  Implementation of the abstract NETIO_Packet type.
- struct NETIO_Interface

  Base-class definition for all NETIO interfaces.

- struct NETIO_InterfaceI

Macros

- #define NETIO_Interface_receive(self_, src_, dst_, p_)

  <<eXtension>> <<cert>> Call receive on a NETIO_Interface.

Typedefs

- typedef struct NETIO_Packet NETIO_Packet_T

  A NETIO Packet.
- typedef struct NETIO_Interface NETIO_Interface_T
- typedef RTI_BOOL(* NETIO_Interface_sendFunc)(NETIO_Interface_T *self, struct NETIO_Interface *source, struct NETIO_Address *destination, NETIO_Packet_T *packet)

  Definition of the NETIO_InterfaceI send method.
- typedef RTI_BOOL(* NETIO_Interface_receiveFunc)(NETIO_Interface_T *netio_intf, struct NETIO_Address *src_addr, struct NETIO_Address *dst_addr, NETIO_Packet_T *packet)

  Definition of the NETIO_InterfaceI receive method.
- typedef RTI_BOOL(* NETIO_Interface_reserve_addressFunc)(NETIO_Interface_T *self, struct NETIO_AddressSeq *req_address, struct NETIO_AddressSeq *rsvd_address, struct NETIOBindProperty *property)

  <<eXtension>> <<cert>> The definition of the NETIO_InterfaceI reserve_address method.
- typedef RTI_BOOL(* NETIO_Interface_release_addressFunc)(NETIO_Interface_T *self, struct NETIO_Address *src_addr)

  <<eXtension>> <<cert>> The definition of the NETIO_InterfaceI release_address method.
- typedef RTI_BOOL(* NETIO_Interface_resolve_addressFunc)(NETIO_Interface_T *netio_intf, const char *address_string, struct NETIO_Address *address_value, RTI_BOOL *invalid)

  <<eXtension>> <<cert>> Resolve an IPv4 address string to NETIO_Address if possible.
- typedef RTI_BOOL(* NETIO_Interface_get_route_tableFunc)(NETIO_Interface_T *netio_intf, struct NETIO_AddressSeq *address, struct NETIO_NetmaskSeq *netmask)

  <<eXtension>> <<cert>> The definition of the::NETIO_InterfaceI get_route_table method.

Functions

- void * NETIO_Packet_get_head (const NETIO_Packet_T *const packet)

  Get a pointer to the current head.
- RTI_BOOL NETIO_Packet_set_payload (NETIO_Packet_T *const packet, void *buffer, RTI_INT32 buffer_length)

  Initialize a packet with a payload.
- RTI_SIZE_T NETIO_Packet_get_payload_length (const NETIO_Packet_T *const packet)

  Get the length the packet payload.

Variables

- const char *const NETIO_DEFAULT_UDP_NAME

  Default name for the automatically registered UDP transport.
- const char *const NETIO_DEFAULT_INTRA_NAME

  Default name for the automatically registered INTRA transport.
- const char *const NETIO_DEFAULT_RTPS_NAME

  Default name for the automatically registered RTPS interface.

7.27.1 Detailed Description

Generic NETIO_Interface functions. This file contains generic NETIO interface functions. A NETIO_Interface is a base-class and is typically not instantiated on its own. Derived classes call methods here to initialize/finalize a base-class. This file also defines the NETIO interface structure as well as the NETIO_Packet structure and protocol types.

7.27.2 Typedef Documentation

7.27.2.1 typedef RTI_BOOL(* NETIO_Interface_reserve_addressFunc)(NETIO_Interface_T *self, struct NETIO_AddressSeq *req_address, struct NETIO_AddressSeq *rsvd_address, struct NETIOBindProperty *property)

<<eXtension>> <<cert>> The definition of the NETIO_InterfaceI reserve_address method.

Parameters

self	*<<in>>* The NETIO interface to reserve addresses on.
req_address	*<<in>>* The list of requested addresses.
rsvd_address	*<<in>> <<out>>* The downstream interface.
property	*<<in>>* Properties to use to listen on the reserved addresses.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

7.27.2.2 typedef RTI_BOOL(* NETIO_Interface_release_addressFunc)(NETIO_Interface_T *self, struct NETIO_Address *src_addr)

<<eXtension>> <<cert>> The definition of the NETIO_InterfaceI release_address method.

Parameters

self	*<<in>>* The NETIO interface to release addresses on.
src_addr	*<<in>>* The address to release.

Returns

RTI_TRUE on success, RTI_FALSE on failure.

7.27.3 Variable Documentation

7.27.3.1 const char* const NETIO_DEFAULT_UDP_NAME

Default name for the automatically registered UDP transport.

7.27.3.2 const char* const NETIO_DEFAULT_INTRA_NAME

Default name for the automatically registered INTRA transport.

7.27.3.3 const char* const NETIO_DEFAULT_RTPS_NAME

Default name for the automatically registered RTPS interface.

7.28 netio_log.h File Reference

NETIO module log codes.

Macros

- #define NETIO_LOG_GETHOST_BYNAME_EC (NETIO_LOG_BASE + 1)

  Failed to get host address from a string representation.
- #define NETIO_LOG_BIND_NEW_EC (NETIO_LOG_BASE + 2)

  Failed to allocate memory for a bind-resolver.
- #define NETIO_LOG_BIND_NEW_RTABLE_EC (NETIO_LOG_BASE + 3)

  Failed to create a database table for routes of a bind-resolver.
- #define NETIO_LOG_BIND_CREATE_RTABLE_EC (NETIO_LOG_BASE + 4)

  Failed to create route record.
- #define NETIO_LOG_BIND_DELETE_RTABLE_EC (NETIO_LOG_BASE + 5)

  Failed to delete a database table for routes of a bind_resolver.
- #define NETIO_LOG_BIND_SET_LENGTH_EC (NETIO_LOG_BASE + 6)

  Failed to set the length for a sequence of resolved addresses.
- #define NETIO_LOG_BIND_SET_MAX_EC (NETIO_LOG_BASE + 7)

  Failed to set the maximum length of an internal sequence of addresses.
- #define NETIO_LOG_BIND_EXTERNAL_FAILED_EC (NETIO_LOG_BASE + 8)

  An interface failed on bind_external.
- #define NETIO_LOG_UNBIND_EXTERNAL_FAILED_EC (NETIO_LOG_BASE + 10)

  An interface failed on unbind_external.
- #define NETIO_LOG_ROUTE_RTABLE_ALLOC_EC (NETIO_LOG_BASE + 11)

  Failed to allocate memory for a route resolver.
- #define NETIO_LOG_ROUTE_RTABLE_CREATE_EC (NETIO_LOG_BASE + 12)

  Failed to create a database table for routes of a route-resolver.
- #define NETIO_LOG_ROUTE_RTABLE_DELETE_EC (NETIO_LOG_BASE + 13)

  Failed to delete a database table for routes of a route-resolver.
- #define NETIO_LOG_ROUTE_RTABLE_ADD_EC (NETIO_LOG_BASE + 14)

  Failed to create route resolver record.
- #define NETIO_LOG_ROUTE_RTABLE_UPDATE_EC (NETIO_LOG_BASE + 15)

  Failed to find routes to update for a route-resolver.
- #define NETIO_LOG_AR_ALLOC_FAILED_EC (NETIO_LOG_BASE + 16)

  Failed to allocate memory for address resolver.
- #define NETIO_LOG_AR_ADD_INVALID_INTERFACE_EC (NETIO_LOG_BASE + 17)

  An attempt was made to add an existing interface with a different port resolver to the address resolver.
- #define NETIO_LOG_AR_ADDRESS_RESOLVE_FAILED_EC (NETIO_LOG_BASE + 18)

  An interface failed to resolve an address.
- #define NETIO_LOG_AR_PORT_RESOLVE_FAILED_EC (NETIO_LOG_BASE + 19)

  An interface failed to resolve a port.
- #define NETIO_LOG_AR_CONTEXT_UNINITIALIZED_EC (NETIO_LOG_BASE + 21)

  NETIO_AddressResolver_resolve/get_next was called with an uninitialized context.
- #define NETIO_LOG_AR_INVALID_TOKEN_EC (NETIO_LOG_BASE + 22)

  An invalid token was encountered in the address string.
- #define NETIO_LOG_AR_ADDRESS_STRING_EXCEEDED_EC (NETIO_LOG_BASE + 23)

  The length of the address exceeded the address buffer passed in.
- #define NETIO_LOG_INVALID_ADDRESS_INDEX_EC (NETIO_LOG_BASE + 24)

  An address index was not parsed correctly.
- #define NETIO_LOG_ILLEGAL_ROUTE_OPERATION_EC (NETIO_LOG_BASE + 25)

  Illegal route operation attempted.
- #define NETIO_LOG_SET_NAME_EC (UDP_LOG_BASE + 26)

  Failed to set the name of a runtime component interface.
- #define NETIO_LOG_PACKET_SET_HEAD_EC (UDP_LOG_BASE + 27)

Failed to set packet's head.

- #define NETIO_LOG_PACKET_SET_TAIL_EC (UDP_LOG_BASE + 28)

  Failed to set packet's tail.
- #define NETIO_LOG_PACKET_INITIALIZE_EC (UDP_LOG_BASE + 29)

  Failed to initialize packet.
- #define NETIO_LOG_LOOP_DELETE_TABLE_EC (NETIO_LOG_BASE + 100)

  Failed to delete a database table for a NETIO loopback interface.
- #define NETIO_LOG_LOOP_CREATE_TABLE_EC (NETIO_LOG_BASE + 101)

  Failed to create a database table for a NETIO loopback interface.
- #define NETIO_LOG_LOOP_SELECT_TABLE_EC (NETIO_LOG_BASE + 102)

  Failed to select a valid record when sending with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_FWD_EC (NETIO_LOG_BASE + 103)

  Failed to send forward with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_BINDX_DB_EC (NETIO_LOG_BASE + 104)

  Failed to bind an external interface with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_UNBINDX_DB_EC (NETIO_LOG_BASE + 105)

  Failed to unbind an external interface with the NETIO loopback interface.
- #define NETIO_LOG_LOOP_SET_LENGTH_EC (NETIO_LOG_BASE + 106)

  Failed to set the length of an address sequence.
- #define NETIO_LOG_LOOP_INVALID_PROPERTY_EC (NETIO_LOG_BASE + 107)

  Invalid property passed in when creating a loopback interface.
- #define NETIO_LOG_LOOP_INITIALIZE_FAILED_EC (NETIO_LOG_BASE + 108)

  Failed to initialize the loopback interface.
- #define NETIO_LOG_LOOP_CURSOR_ERROR_EC (NETIO_LOG_BASE + 109)

  An error occurred with a cursor while iterating over a table.
- #define NETIO_LOG_LOOP_INVALID_FACTORY_EC (NETIO_LOG_BASE + 110)

  An invalid factory was passed in to create a loopback interface.
- #define NETIO_LOG_LOOP_INVALID_COMPONENT_EC (NETIO_LOG_BASE + 111)

  An invalid component was passed in to delete a loopback interface.
- #define UDP_LOG_SOCKET_INIT_EC (UDP_LOG_BASE + 200)

  Failed to initialize use of sockets.
- #define UDP_LOG_GETHOSTNAME_EC (UDP_LOG_BASE + 201)

  Failed to get the host address given the host name.
- #define UDP_LOG_SOCKET_SET_MCAST_EC (UDP_LOG_BASE + 202)

  Failed to set a socket for multicast loopback.
- #define UDP_LOG_SOCKET_SET_MCASTIF_EC (UDP_LOG_BASE + 203)

  Failed to set a socket for multicast bound to a specific interface.
- #define UDP_LOG_SOCKET_CREATE_EC (UDP_LOG_BASE + 204)

  Failed to create a new socket, the sysrc code is the OS return code and reason for failure.
- #define UDP_LOG_SOCKET_SETFD_EC (UDP_LOG_BASE + 205)

  Failed to set the close-on-exec flag for a socket.
- #define UDP_LOG_SOCKET_SNDBUF_EC (UDP_LOG_BASE + 206)

  Failed to set the socket send buffer size.
- #define UDP_LOG_SOCKET_TTL_EC (UDP_LOG_BASE + 207)

  Failed to set multicast TTL for a socket.
- #define UDP_LOG_PACKET_INIT_EC (UDP_LOG_BASE + 208)

  Failed to initialize a receive packet.
- #define UDP_LOG_PACKET_HEAD_EC (UDP_LOG_BASE + 209)

  Failed to set the head of a receive packet.
- #define UDP_LOG_PACKET_FWD_EC (UDP_LOG_BASE + 210)

  Failed reception upstream for a received packet.

- #define UDP_LOG_NAT_DELETE_EC (UDP_LOG_BASE + 211)

  Failed to delete a database record or index from an internal NAT.
- #define UDP_LOG_FINALIZE_EC (UDP_LOG_BASE + 212)

  Failed to finalize the parent NETIO interface.
- #define UDP_LOG_SOCKET_REUSE_PORT_EC (UDP_LOG_BASE + 213)

  Failed to set socket to reuse a port or address.
- #define UDP_LOG_SOCKET_BIND_EC (UDP_LOG_BASE + 214)

  Failed a socket bind.
- #define UDP_LOG_SOCKET_ADDGRP_EC (UDP_LOG_BASE + 215)

  Failed to add membership to a multicast group for a socket.
- #define UDP_LOG_PORT_ENTRY_EC (UDP_LOG_BASE + 216)

  Failed to create a bind entry for a receive port.
- #define UDP_LOG_PORT_ALLOC_EC (UDP_LOG_BASE + 217)

  Failed to allocate memory for a receive buffer of a specific port.
- #define UDP_LOG_SOCKET_RXBUF_EC (UDP_LOG_BASE + 218)

  Failed to set the receive buffer size of a socket.
- #define UDP_LOG_PORT_ADD_EC (UDP_LOG_BASE + 219)

  Failed to add a port entry record to a database table.
- #define UDP_LOG_PORT_THREAD_EC (UDP_LOG_BASE + 220)

  Failed to create a receive thread for a receive port.
- #define UDP_LOG_SOCKET_IFLIST_EC (UDP_LOG_BASE + 221)

  Failed to get a list of interface addresses.
- #define UDP_LOG_SOCKET_IFFLAGS_EC (UDP_LOG_BASE + 222)

  Failed to get the network interface flags or mask of a socket.
- #define UDP_LOG_SOCKET_CLOSE_EC (UDP_LOG_BASE + 223)

  Failed to close a socket.
- #define UDP_LOG_LOADLIB_EC (UDP_LOG_BASE + 224)

  Failed to load a dynamic library.
- #define UDP_LOG_GET_BUF_SIZE_EC (UDP_LOG_BASE + 225)

  Failed to get interface buffer size.
- #define UDP_LOG_GET_BUF_ALLOC_EC (UDP_LOG_BASE + 226)

  Out of memory to allocate interface buffer.
- #define UDP_LOG_GET_IFLIST_EC (UDP_LOG_BASE + 227)

  Failed to get an interface list.
- #define UDP_LOG_GET_NAMEINFO_EC (UDP_LOG_BASE + 228)

  Failed getnameinfo()
- #define UDP_LOG_MULTICAST_ENABLE_EC (UDP_LOG_BASE + 229)

  Failed to enable multicast loopback.
- #define UDP_LOG_UDP_CREATE_TABLE_EC (UDP_LOG_BASE + 230)

  Failed to create a database table for a UDP interface.
- #define UDP_LOG_UDP_CREATE_INDEX_EC (UDP_LOG_BASE + 231)

  Failed to create an index for a UDP interface.
- #define UDP_LOG_RECORD_EC (UDP_LOG_BASE + 232)

  Failed to create or insert a database record for a UDP interface.
- #define UDP_LOG_PROPERTY_FINALIZE_EC (UDP_LOG_BASE + 233)

  Failed to finalize UDP interface factory property.
- #define UDP_LOG_ALLOC_EC (UDP_LOG_BASE + 234)

  Failed to allocate memory for a UDP interface.
- #define UDP_LOG_SEND_PING_EC (UDP_LOG_BASE + 235)

  Failed sending a ping message, upon adding a route or waking up a receive thread.
- #define UDP_LOG_DROPGRP_EC (UDP_LOG_BASE + 236)

Failed to drop membership from a multicast group.

- #define UDP_LOG_GET_LENGTH_EC (UDP_LOG_BASE + 237)

  Length of address sequence exceeds its maximum.
- #define UDP_LOG_SET_LENGTH_EC (UDP_LOG_BASE + 240)

  Failed to set the length of an address sequence.
- #define UDP_LOG_WSA_STARTUP_EC (UDP_LOG_BASE + 241)

  Failed WSAStartup()
- #define UDP_LOG_WINSOCK_INCOMPATIBLE_EC (UDP_LOG_BASE + 242)

  Incompatible version of Winsock, must not be older than 2.0.
- #define UDP_LOG_INCONSISTENT_MAX_MESSSAGE_SIZE_EC (UDP_LOG_BASE + 243)

  Inconsistent max_message_size.
- #define UDP_LOG_INITIALIZE_FAILED_EC (UDP_LOG_BASE + 244)

  Failed to initialized the loopback interface.
- #define UDP_LOG_PORT_NOT_FOUND_EC (UDP_LOG_BASE + 245)

  Did not find the specified thread port.
- #define UDP_LOG_CURSOR_ERROR_EC (UDP_LOG_BASE + 246)

  An error occurred while iterating over a cursor.
- #define UDP_LOG_SEND_ERROR_EC (UDP_LOG_BASE + 247)

  Failed to send message.
- #define UDP_LOG_MULTICAST_NOT_ENABLED_EC (UDP_LOG_BASE + 248)

  Multicast address ignored because multicast supported is not compiled in.
- #define UDP_LOG_TRANSPORT_NO_INTERFACES_EC (UDP_LOG_BASE + 249)

  Transport does not have any interfaces.
- #define UDP_LOG_RECV_ERROR_EC (UDP_LOG_BASE + 250)

  Failed to receive message.

7.28.1 Detailed Description

NETIO module log codes.

7.29 netio_rtps.h File Reference

RTPS API.

7.29.1 Detailed Description

RTPS API.

7.30 netio_zcopy.h File Reference

Zero Copy transport API.

Functions

- RTI_BOOL NDDS_Transport_ZeroCopy_initialize (RT_Registry_T *registry, const char *transport_name, struct ZCOPY_NotifInterfaceFactoryProperty *property)

  Initialize the NDDS transport with ZeroCopy support.

Variables

- const char *const NETIO_DEFAULT_NOTIF_NAME

  Default name for notification interface.

7.30.1 Detailed Description

Zero Copy transport API.

7.31 netio_zcopy_data.h File Reference

ZeroCopy extensions for DDS typed interface.

Functions

- DDS_ReturnCode_t FooDataWriter_get_loan (FooDataWriter *self, Foo **sample)

  Request a sample from a managed sample pool.
- DDS_ReturnCode_t FooDataWriter_discard_loan (FooDataWriter *self, Foo *sample)

  Return a Sample to a managed Sample pool.
- DDS_ReturnCode_t FooDataReader_is_data_consistent (DDS_DataReader *self, DDS_Boolean *is_data_consistent, const Foo *sample, const struct DDS_SampleInfo *sample_info)

  Checks if the sample has been overwritten by the DataWriter.

7.31.1 Detailed Description

ZeroCopy extensions for DDS typed interface.

7.32 netio_zcopy_dll.h File Reference

ZeroCopy DLL external linkage.

7.32.1 Detailed Description

ZeroCopy DLL external linkage.

7.33 netio_zcopy_log.h File Reference

ZeroCopy module log codes.

7.33.1 Detailed Description

ZeroCopy module log codes.

7.34 netio_zcopy_notif_interface.h File Reference

Data Structures

- struct ZCOPY_NotifInterfaceFactoryProperty

  Notification interface factory specific property.

Macros

- #define ZCOPY_NotifInterfaceFactoryProperty_INITIALIZER

  Constant to initialize a ZCOPY_NotifInterfaceFactoryProperty.

Functions

- RTI_BOOL ZCOPY_NotifInterface_receive (NETIO_Interface_T *netio_intf, const struct NETIO_Address *port, RTI_BOOL *has_more_data_out)

  Definition of the NETIO_InterfaceI receive method.
- RTI_BOOL ZCOPY_NotifInterfaceFactory_register (RT_Registry_T *registry, const char *name, struct ZCOPY_NotifInterfaceFactoryProperty *property)

  Register the a Notification interface factory with a factory property.
- RTI_BOOL ZCOPY_NotifInterfaceFactory_unregister (RT_Registry_T *registry, const char *name)

  Unregister a Notification interface factory.

7.34.1 Detailed Description

7.35 netio_zcopy_notif_mechanism.h File Reference

Data Structures

- struct ZCOPY_NotifUserInterfaceI

  Notification mechanism user interface.

Typedefs

- typedef NETIO_Interface_T *(* ZCOPY_NotifUserInterface_createFunc)(NETIO_Interface_T *upstream, void *user_property)

  Create an instance of a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_reserve_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, void **port_entry_out)

  Reserve an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_release_addressFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, void *port_entry)

  Release an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_add_routeFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source, struct NETIO_Address *destination, void **route_entry_out)

  Add a route on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_delete_routeFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *source, struct NETIO_Address *destination, void *route_entry)

  Delete a route on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_sendFunc)(NETIO_Interface_T *self, struct NETIO_Address *source, struct NETIO_Address *destination, void *route_entry)

  Send a notification on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_bindFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, struct NETIO_Address *dst_addr, void *port_entry, void **bind_entry_out)

  Bind an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_unbindFunc)(NETIO_Interface_T *user_intf, struct NETIO_Address *src_addr, struct NETIO_Address *dst_addr, void *port_entry, void *bind_entry)

  Unbind an address on a user interface.
- typedef RTI_BOOL(* ZCOPY_NotifUserInterface_notify_portFunc)(NETIO_Interface_T *user_intf, void *port_entry)

  Notify a specific receive port.

7.35.1 Detailed Description

7.36 netio_zcopy_publication.h File Reference

ZeroCopy extensions for DDS publication module.

7.36.1 Detailed Description

ZeroCopy extensions for DDS publication module.

7.37 netio_zcopy_subscription.h File Reference

ZeroCopy extensions for DDS subscription module.

7.37.1 Detailed Description

ZeroCopy extensions for DDS subscription module.

7.38 netio_zcopy_tdatareader_gen.h File Reference

ZeroCopy extensions for DDS DataReader.

7.38.1 Detailed Description

ZeroCopy extensions for DDS DataReader.

7.39 netio_zcopy_tdatawriter_gen.h File Reference

ZeroCopy extensions for DDS DataWriter.

7.39.1 Detailed Description

ZeroCopy extensions for DDS DataWriter.

7.40 netio_zcopy_whsq.h File Reference

Shared Queue Writer History includes.

Functions

- struct RT_ComponentFactoryI * WHSQ_HistoryFactory_get_interface (void)

  Function to retrieve the concrete implementation of the WHSQ interface factory.

7.40.1 Detailed Description

Shared Queue Writer History includes.

7.40.2 Function Documentation

7.40.2.1 struct RT_ComponentFactoryI* WHSQ_HistoryFactory_get_interface (void) [read]

Function to retrieve the concrete implementation of the WHSQ interface factory.

Returns

Pointer to RT Component factory implementation

7.41 netio_zcopy_whsq_log.h File Reference

WHSQ module log codes.

Macros

- #define WHSQ_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (WHSQ_LOG_BASE + 1)
  KEEP_ALL History kind is not supported.
- #define WHSQ_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC (WHSQ_LOG_BASE + 2)
  Unlimited length resource limits unsupported.
- #define WHSQ_LOG_MAX_SAMPLES_TOO_SMALL_EC (WHSQ_LOG_BASE + 3)
  DataWriterQos.resource_limits.max_samples set too small.
- #define WHSQ_LOG_HISTORY_OBJECT 1
  A history object.
- #define WHSQ_LOG_OBJECT_ALLOCATE_EC (WHSQ_LOG_BASE + 4)
  Failed to allocate object of the specified kind.
- #define WHSQ_LOG_PURGE_FAILED_EC (WHSQ_LOG_BASE + 5)
  Failed to purge a sample from the shared queue.
- #define WHSQ_LOG_WH_CREATE_FAILED_EC (WHSQ_LOG_BASE + 6)
  Failed to create the underlying writer history.
- #define WHSQ_LOG_OBJECT_DELETE_EC (WHSQ_LOG_BASE + 7)
  Failed to delete object of the specified kind.
- #define WHSQ_LOG_CONFIGURATION_EC (WHSQ_LOG_BASE + 8)
  Listener or Property not configured when creating the component.
- #define WHSQ_LOG_NO_DDS_FACTORY_EC (WHSQ_LOG_BASE + 9)
  No factory for forwarding when creating a writer history instance.
- #define WHSQ_LOG_COMMIT_FAILED_EC (WHSQ_LOG_BASE + 10)
  Committing a Sample failed.

7.41.1 Detailed Description

WHSQ module log codes.

7.41.2 Macro Definition Documentation

7.41.2.1 #define WHSQ_LOG_HISTORY_OBJECT 1

A history object.

7.42 osapi_cc.h File Reference

OS API Configuration.

Macros

- #define RTI_PRIVATE static

  If the compiler does not support static override it here.

7.42.1 Detailed Description

OS API Configuration.

7.42.2 Macro Definition Documentation

7.42.2.1 #define RTI_PRIVATE static

If the compiler does not support static override it here.

7.43 osapi_cc_gcc.h File Reference

OS API Configuration.

7.43.1 Detailed Description

OS API Configuration.

7.44 osapi_cc_qcc.h File Reference

OS API Configuration.

7.44.1 Detailed Description

OS API Configuration.

7.45 osapi_cc_stdc.h File Reference

OS API Configuration.

7.45.1 Detailed Description

OS API Configuration.

7.46 osapi_config.h File Reference

OS API Configuration.

7.46.1 Detailed Description

OS API Configuration.

7.47 osapi_heap_impl.h File Reference

Macro implementing heap interface.

7.47.1 Detailed Description

Macro implementing heap interface.

7.48 osapi_log_impl.h File Reference

Implementation of log functionality.

7.48.1 Detailed Description

Implementation of log functionality.

7.49 osapi_os.h File Reference

OS API Configuration.

7.49.1 Detailed Description

OS API Configuration.

7.50 osapi_os_def.h File Reference

OS API Configuration, pre-split.

7.50.1 Detailed Description

OS API Configuration, pre-split.

7.51 osapi_os_posix.h File Reference

OS API Configuration.

7.51.1 Detailed Description

OS API Configuration.

7.52 osapi_process.h File Reference

Process interface definition.

Typedefs

- typedef RTI_UINT64 OSAPI_ProcessId

  OSAPI_ProcessId type.

Functions

- OSAPI_ProcessId OSAPI_Process_getpid (void)

  Return the process ID, which is unique for the application.

7.52.1 Detailed Description

Process interface definition.

7.53 osapi_system.h File Reference

System API definition.

Data Structures

- struct OSAPI_SystemI

  <<*cert*>> *The System API.*
- struct OSAPI_SystemProperty

  <<*cert*>> *The system properties.*

Macros

- #define OSAPI_SystemI_INITIALIZER

  <<*cert*>> *The initializer for OSAPI_SystemI.*

Typedefs

- typedef RTI_BOOL(* OSAPI_System_start_timer_T)(OSAPI_Timer_T self, OSAPI_TimerTickHandlerFunction tick_handler)

  Start an OSAPI timer.
- typedef RTI_INT32(* OSAPI_System_get_timer_resolution_T)(void)

  Get the resolution of the clock driving the timer in nanoseconds.
- typedef RTI_BOOL(* OSAPI_System_get_time_T)(OSAPI_SystemTime *now)

  Get the current system time.
- typedef RTI_BOOL(* OSAPI_System_initialize_T)(void)

  Initialize the system.
- typedef RTI_BOOL(* OSAPI_System_get_hostname_T)(char *const hostname)

  Get the hostname.
- typedef RTI_BOOL(* OSAPI_System_generate_uuid_T)(struct OSAPI_SystemUUID *uuid_out)

  Generate a unique universal identifier (UUID).
- typedef RTI_BOOL(* OSAPI_System_get_ticktime_T)(RTI_INT32 *sec, RTI_UINT32 *nanosec)

  Get the current tick time.
- typedef struct OSAPI_SystemI OSAPI_SystemI

  <<*cert*>> *The System API.*

Functions

- RTI_BOOL OSAPI_System_set_interface (struct OSAPI_SystemI *intf)
  Set the system interface.
- RTI_BOOL OSAPI_System_get_property (struct OSAPI_SystemProperty *property)
  Get the system properties.
- RTI_BOOL OSAPI_System_set_property (struct OSAPI_SystemProperty *property)
  Set the system properties.
- void OSAPI_System_get_native_interface (struct OSAPI_SystemI *intf)
  Get the native system interface.
- RTI_BOOL OSAPI_System_clock_tick (RTI_UINT32 ticks)
  Advance the internal RTI Connext Micro timers with the specified number of clock ticks.
- RTI_BOOL OSAPI_System_clock_tick_from_time (void)
  Advance the internal RTI Connext Micro timers using the OSAPI_SystemI::get_time API.

7.53.1 Detailed Description

System API definition.

7.54 osapi_time.h File Reference

Time API definition.

Data Structures

- struct OSAPI_SystemTime
  System time.

Typedefs

- typedef struct OSAPI_SystemTime OSAPI_SystemTime
  System time.

7.54.1 Detailed Description

Time API definition. This file implements functions to convert between time formats.

7.54.2 Typedef Documentation

7.54.2.1 typedef struct OSAPI_SystemTime OSAPI_SystemTime

System time.

Expresses time in seconds and nanoseconds. Assumes that Nanoseconds is less than 1000000000.

7.55 reda_dll.h File Reference

7.55.1 Detailed Description

7.56 reda_epoch.h File Reference

7.56.1 Detailed Description

7.57 reda_log.h File Reference

REDA module log codes.

Macros

- #define REDA_LOG_BUFFERPOOL_OUT_OF_RESOURCES_EC (REDA_LOG_BASE + 1)

  Not sufficient memory to allocate buffer-pool.
- #define REDA_LOG_BUFFERPOOL_BUFFER_INITIALIZATION_FAILED_EC (REDA_LOG_BASE + 2)

  The buffer initialization method failed.
- #define REDA_LOG_BUFFERPOOL_NOT_EMPTY_EC (REDA_LOG_BASE + 3)

  Cannot delete buffer-pool due to buffer(s) not returned to pool.
- #define REDA_LOG_BUFFERPOOL_DOUBLE_FREE_EC (REDA_LOG_BASE + 4)

  A buffer was freed more than once.
- #define REDA_LOG_SEQUENCE_COPY_FAILED_EC (REDA_LOG_BASE + 100)

  An error occurred while copying a sequence.
- #define REDA_LOG_SEQUENCE_ALLOC_FAILED_EC (REDA_LOG_BASE + 101)

  An error occured while allocating space for a sequence.
- #define REDA_LOG_SEQUENCE_SET_MAX_FAILED_EC (REDA_LOG_BASE + 102)

  An error occured while setting the maximum sequence size.
- #define REDA_LOG_SEQUENCE_REALLOCATION_EC (REDA_LOG_BASE + 103)

  An attempt was made to resize an already allocated sequence.
- #define REDA_LOG_SEQUENCE_INVALID_OPERATION_EC (REDA_LOG_BASE + 104)

  An error occured while operating on two sequences (copy/compare)
- #define REDA_LOG_SEQUENCE_INVALID_LENGTH_EC (REDA_LOG_BASE + 105)

  An attempt was made to set an illegal length (length $<$ 0 or length $>$ max_length)
- #define REDA_LOG_SEQUENCE_INDEX_OUT_OF_BOUNDS_EC (REDA_LOG_BASE + 106)

  An illegal sequence index was specified (index $<$ 0 or index $>$ length)
- #define REDA_LOG_STRING_ALLOC_FAILED_EC (REDA_LOG_BASE + 200)

  An error occured while allocating memory for a string.
- #define REDA_LOG_INDEX_FULL_EC (REDA_LOG_BASE + 300)

  An attempt was made to add an element to a full index.
- #define REDA_LOG_INDEX_ENTRY_EXISTS_EC (REDA_LOG_BASE + 301)

  An attempt was made to add an existing element to an index.
- #define REDA_LOG_INVALID_LIST_NODE_EC (REDA_LOG_BASE + 302)

  An attempt was made to add an existing element to an index.

7.57.1 Detailed Description

REDA module log codes.

7.58 reda_sequence_defn.h File Reference

7.58.1 Detailed Description

7.59 rh_sm_dll.h File Reference

ReaderHistory DLL external linkage.

7.59.1 Detailed Description

ReaderHistory DLL external linkage.

7.60 rh_sm_history.h File Reference

Reader History API.

Functions

- struct RT_ComponentFactoryI * RHSM_HistoryFactory_get_interface (void)
  Gets the singleton instance of the RHSM reader history factory.

7.60.1 Detailed Description

Reader History API.

7.61 rh_sm_log.h File Reference

RH module log codes.

Macros

- #define RHSM_LOG_OUTSTANDING_SAMPLE_EC (RHSM_LOG_BASE + 1)
  Failed to delete Reader History due to outstanding, unremoved sample.
- #define RHSM_LOG_NO_PROPERTY_QOS_EC (RHSM_LOG_BASE + 2)
  Failed to create Reader History due to unspecified property Qos.
- #define RHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 3)
  KEEP_ALL History kind is not supported.
- #define RHSM_LOG_MAX_SAMPLE_TOO_SMALL_EC (RHSM_LOG_BASE + 4)
  DataReaderQos.resource_limits.max_samples set too small.
- #define RHSM_LOG_TBF_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 5)
  Time-based filtering is not supported.
- #define RHSM_LOG_SAMPLE_INFO_POOL_EC (RHSM_LOG_BASE + 6)
  Failed to allocate a buffer pool for sample info.
- #define RHSM_LOG_SAMPLE_POOL_EC (RHSM_LOG_BASE + 7)
  Failed to allocate a buffer pool for sample pointers.
- #define RHSM_LOG_SAMPLE_PTR_ARRAY_EC (RHSM_LOG_BASE + 8)
  Failed to get sample buffer.
- #define RHSM_LOG_INFO_ARRAY_EC (RHSM_LOG_BASE + 9)
  Failed to get sample info buffer.
- #define RHSM_LOG_GET_RECEPTION_TIMESTAMP_EC (RHSM_LOG_BASE + 10)
  Failed to get time for reception timestamp.
- #define RHSM_LOG_INVALID_INSTANCE_REPLACEMENT_EC (RHSM_LOG_BASE + 11)
  An invalid instance replacement policy was specified.
- #define RHSM_LOG_FAILED_TO_REMOVE_OLDEST_EC (RHSM_LOG_BASE + 12)
  Failed to remove the oldest sample.
- #define RHSM_LOG_SAMPLE_POOL_EMPTY_EC (RHSM_LOG_BASE + 13)
  Sample pool empty, may have exceeded.
- #define RHSM_LOG_RW_PRUNE_FAILED_EC (RHSM_LOG_BASE + 14)

Failed to prune remote writer entry.

- #define RHSM_LOG_ENTRY_RESERVATION_FAILED_EC (RHSM_LOG_BASE + 15)

  Failed to reserve an entry.
- #define RHSM_LOG_RETURN_SAMPLE_EC (RHSM_LOG_BASE + 16)

  Failed to return a sample.
- #define RHSM_LOG_HISTORY_UPDATE_EC (RHSM_LOG_BASE + 17)

  Failed to update the history queue.
- #define RHSM_LOG_GET_TIME_EC (RHSM_LOG_BASE + 18)

  Failed to get the system time.
- #define RHSM_LOG_HISTORY_OBJECT 1

  A history object.
- #define RHSM_LOG_REMOVELIST_OBJECT 2

  A removal list object.
- #define RHSM_LOG_RWPOOL_OBJECT 3

  A remote writer pool object.
- #define RHSM_LOG_RWINDEX_OBJECT 4

  A remote writer index object.
- #define RHSM_LOG_KEY_OBJECT 5

  A Key object.
- #define RHSM_LOG_RW_OBJECT 6

  A Key object.
- #define RHSM_LOG_KEYPOOL_OBJECT 7

  A Key pool object.
- #define RHSM_LOG_OWNER_OBJECT 8

  A key owner object.
- #define RHSM_LOG_SAMPLEPOOL_OBJECT 9

  A sample pool object.
- #define RHSM_LOG_KEYINDEXPOOL_OBJECT 10

  A key index pool.
- #define RHSM_LOG_SAMPLEPTPOOL_OBJECT 11

  A sample ptr pool object.
- #define RHSM_LOG_SAMPLEINFO_OBJECT 12

  A sample info pool.
- #define RHSM_LOG_INSTANCEHANDLE_OBJECT 13

  A sample info pool.
- #define RHSM_LOG_OBJECT_ALLOCATE_EC (RHSM_LOG_BASE + 19)

  Failed to allocate object of the specified kind.
- #define RHSM_LOG_OBJECT_DELETE_EC (RHSM_LOG_BASE + 20)

  Failed to delete object of the specified kind.
- #define RHSM_LOG_OBJECT_INDEX_EC (RHSM_LOG_BASE + 21)

  Failed to index an object of the specified kind.
- #define RHSM_LOG_OBJECT_INVALID_EC (RHSM_LOG_BASE + 22)

  Invalid object specified.
- #define RHSM_LOG_NO_PROPERTY_EC (RHSM_LOG_BASE + 23)

  No property when creating a reader history instance.
- #define RHSM_LOG_DESTINATION_BY_SOURCE_NOT_SUPPORTED_EC (RHSM_LOG_BASE + 24)

  Destination ordering by SOURCE_TIMESTAMP ordering is not supported.
- #define RHSM_LOG_RECOMMIT_FAILURE_EC (RHSM_LOG_BASE + 25)

  Failed to recommit sample.

7.61.1 Detailed Description

RH module log codes.

7.61.2 Macro Definition Documentation

7.61.2.1 #define RHSM_LOG_HISTORY_OBJECT 1

A history object.

7.61.2.2 #define RHSM_LOG_REMOVELIST_OBJECT 2

A removal list object.

7.61.2.3 #define RHSM_LOG_RWPOOL_OBJECT 3

A remote writer pool object.

7.61.2.4 #define RHSM_LOG_RWINDEX_OBJECT 4

A remote writer index object.

7.61.2.5 #define RHSM_LOG_KEY_OBJECT 5

A Key object.

7.61.2.6 #define RHSM_LOG_RW_OBJECT 6

A Key object.

7.61.2.7 #define RHSM_LOG_KEYPOOL_OBJECT 7

A Key pool object.

7.61.2.8 #define RHSM_LOG_OWNER_OBJECT 8

A key owner object.

7.61.2.9 #define RHSM_LOG_SAMPLEPOOL_OBJECT 9

A sample pool object.

7.61.2.10 #define RHSM_LOG_KEYINDEXPOOL_OBJECT 10

A key index pool.

7.61.2.11 #define RHSM_LOG_SAMPLEPTPOOL_OBJECT 11

A sample ptr pool object.

7.61.2.12 #define RHSM_LOG_SAMPLEINFO_OBJECT 12

A sample info pool.

7.61.2.13 #define RHSM_LOG_INSTANCEHANDLE_OBJECT 13

A sample info pool.

7.62 rt_dll.h File Reference

7.62.1 Detailed Description

7.63 rt_log.h File Reference

Macros

- #define RT_LOG_SET_IMMUTABLE_PROPERTY_EC (RT_LOG_BASE + 1)

  Failed to set registry property due to registry already being enabled.
- #define RT_LOG_REGISTRY_INIT_FAILURE_EC (RT_LOG_BASE + 2)

  Failed to initialize registry due to failed table creation.
- #define RT_LOG_REGISTRY_FINALIZE_EC (RT_LOG_BASE + 3)

  Failed to finalize registry due to failed table deletion.
- #define RT_LOG_REGISTRY_EXISTS_EC (RT_LOG_BASE + 4)

  An attempt was made to register a factory that already existed.
- #define RT_LOG_REGISTRY_REGISTER_EC (RT_LOG_BASE + 5)

  Error registering a component factory. May have exceeded RT_RegistryProperty.max_factories.
- #define RT_LOG_REGISTRY_INIT_FACTORY_EC (RT_LOG_BASE + 7)

  A registered factory failed to initialize.
- #define RT_LOG_REGISTRY_NAME_TOO_LONG_EC (RT_LOG_BASE + 8)

  Factory name longer than maximum length of 8.
- #define RT_LOG_REGISTRY_INCONSISTENT_CID_EC (RT_LOG_BASE + 9)

  Factory name exists, but the class ID is not of the requested type.
- #define RT_LOG_REGISTRY_NOT_INITIALIZED_EC (RT_LOG_BASE + 10)

  An attempt was made to operate on an uninitialized registry.

7.63.1 Detailed Description

7.64 rtps_config.h File Reference

RTPS Configuration.

7.64.1 Detailed Description

RTPS Configuration. Configurable build flags for RTPS functionality

7.65 rtps_dll.h File Reference

RTPS Configuration.

7.65.1 Detailed Description

RTPS Configuration. Configurable build flags for RTPS functionality

7.66 rtps_log.h File Reference

RTPS module log codes.

Macros

- #define RTPS_LOG_INITIALIZE_INTERFACE_EC (RTPS_LOG_BASE + 1)

  Failed to initialize an RTPS interface.
- #define RTPS_LOG_ALLOCATE_EC (RTPS_LOG_BASE + 2)

  Failed to allocate heap memory for internal resources.
- #define RTPS_LOG_CREATE_ROUTE_TABLE_EC (RTPS_LOG_BASE + 3)

  Failed to create a database table for storing RTPS route info.
- #define RTPS_LOG_DB_CREATE_ROUTE_PEER_INDEX_EC (RTPS_LOG_BASE + 4)

  Failed to create database index for route-peer info.
- #define RTPS_LOG_DB_CREATE_ROUTE_XPORT_INDEX_EC (RTPS_LOG_BASE + 5)

  Failed to create database index for route-transport info.
- #define RTPS_LOG_DB_CREATE_BIND_PEER_INDEX_EC (RTPS_LOG_BASE + 6)

  Failed to create database index for bind-peer info.
- #define RTPS_LOG_INDEX_ADD_ENTRY_EC (RTPS_LOG_BASE + 8)

  Failed to add an entry to a database index.
- #define RTPS_LOG_DB_SELECT_ALL_EC (RTPS_LOG_BASE + 9)

  Failed to select all entries of a database table.
- #define RTPS_LOG_DB_SELECT_MATCH_EC (RTPS_LOG_BASE + 10)

  Matching entry not found in a database table.
- #define RTPS_LOG_DB_SELECT_RANGE_EC (RTPS_LOG_BASE + 11)

  No matching entries found in a database table.
- #define RTPS_LOG_DB_CREATE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 12)

  Failed to create database entry for route.
- #define RTPS_LOG_DB_INSERT_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 13)

  Failed to insert entry into route table.
- #define RTPS_LOG_DB_REMOVE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 14)

  Failed to remove entry from route table.
- #define RTPS_LOG_DB_DELETE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 15)

  Failed to delete entry from route table.
- #define RTPS_LOG_DB_CREATE_BIND_ENTRY_EC (RTPS_LOG_BASE + 16)

  Failed to create database entry for bind.
- #define RTPS_LOG_DB_INSERT_BIND_ENTRY_EC (RTPS_LOG_BASE + 17)

  Failed to insert an entry into the bind table.
- #define RTPS_LOG_DB_REMOVE_BIND_ENTRY_EC (RTPS_LOG_BASE + 18)

  Failed to remove an entry from the bind table.
- #define RTPS_LOG_DB_REMOVE_EXTERNAL_BIND_ENTRY_EC (RTPS_LOG_BASE + 19)

  Failed to remove an entry from the external bind table.
- #define RTPS_LOG_DB_DELETE_EXTERNAL_BIND_ENTRY_EC (RTPS_LOG_BASE + 20)

  Failed to delete entry from external bind table.
- #define RTPS_LOG_SEND_EC (RTPS_LOG_BASE + 21)

  Failed to send a packet due to a lower module failing to send the packet.
- #define RTPS_LOG_RECEIVE_EC (RTPS_LOG_BASE + 22)

  Failed to receive a packet due to a higher module failing to receive the packet.
- #define RTPS_LOG_ROUTE_PACKET_EC (RTPS_LOG_BASE + 23)

  Failed to send a packet, when routing to a lower module or peer.
- #define RTPS_LOG_FORWARD_UPSTREAM_EC (RTPS_LOG_BASE + 24)

  Failed to receive a packet, when forwarding to a higher upstream module.
- #define RTPS_LOG_BAD_PARAMETER_EC (RTPS_LOG_BASE + 25)

  Bad parameter to an RTPS interface function.
- #define RTPS_LOG_NOT_ENABLED_EC (RTPS_LOG_BASE + 26)

Failed because interface is not enabled.

- #define RTPS_LOG_READER_UNSUPPORTED_EC (RTPS_LOG_BASE + 27)

  RTPS reader does not support send.
- #define RTPS_LOG_INTERFACE_MISMATCH_EC (RTPS_LOG_BASE + 28)

  Upstream interface does not match source or destination of packet being sent or received.
- #define RTPS_LOG_FULL_SEND_WINDOW_EC (RTPS_LOG_BASE + 29)

  Failed to send due to reaching maximum send window size.
- #define RTPS_LOG_NETIO_PACKET_SET_TAIL_EC (RTPS_LOG_BASE + 31)

  Failed to set tail of packet to send.
- #define RTPS_LOG_FUNC_UNSUPPORTED_EC (RTPS_LOG_BASE + 32)

  RTPS interface does not support this function.
- #define RTPS_LOG_EXCEEDED_LIMIT_TRANSPORTS_EC (RTPS_LOG_BASE + 33)

  Out of transport entries.
- #define RTPS_LOG_EXCEEDED_LIMIT_PEERS_EC (RTPS_LOG_BASE + 34)

  Out of peer entries.
- #define RTPS_LOG_ASSERT_PEER_EC (RTPS_LOG_BASE + 35)

  Failed to assert a remote writer or reader.
- #define RTPS_LOG_ASSERT_TRANSPORT_EC (RTPS_LOG_BASE + 36)

  Failed to assert a downstream transport.
- #define RTPS_LOG_FIND_EXTERNAL_INTERFACE_EC (RTPS_LOG_BASE + 37)

  Failed to lookup an external interface.
- #define RTPS_LOG_NONEXISTENT_ROUTE_EC (RTPS_LOG_BASE + 38)

  Failed to delete a nonexistent route.
- #define RTPS_LOG_NONEXISTENT_BIND_EC (RTPS_LOG_BASE + 39)

  Failed to remove a nonexistent bind entry.
- #define RTPS_LOG_NONEXISTENT_EXTERNAL_BIND_EC (RTPS_LOG_BASE + 40)

  Failed to remove a nonexistent external bind entry.
- #define RTPS_LOG_STALE_ACK_EPOCH__EC (RTPS_LOG_BASE + 41)

  Dropped an ACKNACK submessage with an old epoch.
- #define RTPS_LOG_STALE_HB_EPOCH_EC (RTPS_LOG_BASE + 42)

  Dropped a HEARTBEAT submessage with an old epoch.
- #define RTPS_LOG_ACK_EC (RTPS_LOG_BASE + 43)

  Failed an acknack() upstream.
- #define RTPS_LOG_REQUEST_EC (RTPS_LOG_BASE + 45)

  Failed a request() upstream.
- #define RTPS_LOG_RETURN_LOAN_EC (RTPS_LOG_BASE + 46)

  Failed a return_loan() upstream.
- #define RTPS_LOG_NETIO_PACKET_SET_HEAD_EC (RTPS_LOG_BASE + 47)

  Failed to set the head of a packet.
- #define RTPS_LOG_SEND_HEARTBEAT_EC (RTPS_LOG_BASE + 48)

  Failed to send a HEARTBEAT submessage.
- #define RTPS_LOG_SHIFT_BITMAP_EC (RTPS_LOG_BASE + 49)

  Failed to shift a bitmap.
- #define RTPS_LOG_FULLY_ACKED_READER_EC (RTPS_LOG_BASE + 50)

  A Reader is fully acknowledged and does not need to respond to a final HEARTBEAT.
- #define RTPS_LOG_DATA_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 51)

  Dropping a DATA submessage whose sequence number is outside of a Reader's receive window.
- #define RTPS_LOG_DATA_ALREADY_RECEIVED_EC (RTPS_LOG_BASE + 52)

  Dropping a DATA submessage whose sequence number was previously received.
- #define RTPS_LOG_INTERFACE_NOT_ENABLED_EC (RTPS_LOG_BASE + 53)

  Failed to receive a message because interface is not enabled.

- #define RTPS_LOG_INVALID_PACKET_EC (RTPS_LOG_BASE + 54)

  Dropped a message with an invalid packet header.
- #define RTPS_LOG_UNSUPPORTED_INTERFACE_EC (RTPS_LOG_BASE + 55)

  Received a message on an unsupported interface.
- #define RTPS_LOG_UNKNOWN_SUBMESSAGE_EC (RTPS_LOG_BASE + 56)

  Received a submessage with an unknown ID.
- #define RTPS_LOG_PROCESS_ACKNACK_EC (RTPS_LOG_BASE + 57)

  Failed to process received ACKNACK submessage.
- #define RTPS_LOG_PROCESS_DATA_EC (RTPS_LOG_BASE + 58)

  Failed to process received DATA submessage.
- #define RTPS_LOG_PROCESS_GAP_EC (RTPS_LOG_BASE + 59)

  Failed to process received GAP submessage.
- #define RTPS_LOG_PROCESS_HEARTBEAT_EC (RTPS_LOG_BASE + 60)

  Failed to process received HEARTBEAT submessage.
- #define RTPS_LOG_CREATE_HB_EVENT_EC (RTPS_LOG_BASE + 61)

  Failed to create periodic HEARTBEAT event.
- #define RTPS_LOG_CREATE_EVENT_TIMER_EC (RTPS_LOG_BASE + 62)

  Failed to create periodic HEARTBEAT event timer.
- #define RTPS_LOG_SEQ_NUM_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 63)

  Failed to set bitmap bit due to out-of-range sequence number.
- #define RTPS_LOG_FIRST_SN_GREATER_LAST_SN_EC (RTPS_LOG_BASE + 65)

  Invalid range of sequence numbers to fill in bitmap.
- #define RTPS_LOG_BITCOUNT_OUT_OF_BOUNDS_EC (RTPS_LOG_BASE + 66)

  Deserialized an invalid out-of-bounds bitcount for a bitmap.
- #define RTPS_LOG_SHIFT_SN_OUT_OF_RANGE_EC (RTPS_LOG_BASE + 67)

  Failed to shift bitmap due to invalid starting sequence number.
- #define RTPS_LOG_SERIALIZE_HOST_ID_EC (RTPS_LOG_BASE + 68)

  Failed to serialize host ID of GUID.
- #define RTPS_LOG_SERIALIZE_APP_ID_EC (RTPS_LOG_BASE + 69)

  Failed to serialize app ID of GUID.
- #define RTPS_LOG_SERIALIZE_INSTANCE_ID_EC (RTPS_LOG_BASE + 70)

  Failed to serialized instance ID of GUID.
- #define RTPS_LOG_SERIALIZE_OBJECT_ID_EC (RTPS_LOG_BASE + 71)

  Failed to serialize object ID of GUID.
- #define RTPS_LOG_DESERIALIZE_HOST_ID_EC (RTPS_LOG_BASE + 72)

  Failed to deserialize host ID of GUID.
- #define RTPS_LOG_DESERIALIZE_APP_ID_EC (RTPS_LOG_BASE + 73)

  Faild to deserialize app ID of GUID.
- #define RTPS_LOG_DESERIALIZE_INSTANCE_ID_EC (RTPS_LOG_BASE + 74)

  Failed to deserialize instance ID of GUID.
- #define RTPS_LOG_DESERIALIZE_OBJECT_ID_EC (RTPS_LOG_BASE + 75)

  Failed to deserialize object ID of GUID.
- #define RTPS_LOG_DB_CREATE_EXT_BIND_ENTRY_EC (RTPS_LOG_BASE + 76)

  Failed to create database entry for external bind table.
- #define RTPS_LOG_BITMAP_SET_BIT_EC (RTPS_LOG_BASE + 77)

  Failed to set bit in bitmap.
- #define RTPS_LOG_UNSUPPORTED_INFO_REPLY_EC (RTPS_LOG_BASE + 78)

  Received and dropped currently unsupported INFO_REPLY submessage.
- #define RTPS_LOG_UNSUPPORTED_INFO_REPLY_IP4_EC (RTPS_LOG_BASE + 79)

  Received and dropped currently unsupported INFO_REPLY_IP4 submessage.
- #define RTPS_LOG_UNSUPPORTED_INFO_SRC_EC (RTPS_LOG_BASE + 80)

Received and dropped currently unsupported INFO_SRC submessage.

- #define RTPS_LOG_DELETE_INDEX_EC (RTPS_LOG_BASE + 81)

  Failed to delete index of a database table.
- #define RTPS_LOG_DELETE_TABLE_EC (RTPS_LOG_BASE + 82)

  Failed to delete a database table.
- #define RTPS_LOG_DB_LOCK_EC (RTPS_LOG_BASE + 83)

  Failed to take database lock.
- #define RTPS_LOG_DB_UNLOCK_EC (RTPS_LOG_BASE + 84)

  Failed to give database lock.
- #define RTPS_LOG_CREATE_BIND_TABLE_EC (RTPS_LOG_BASE + 85)

  Failed to create a database table for storing RTPS bind info.
- #define RTPS_LOG_STATUS_CHANGE_EC (RTPS_LOG_BASE + 86)

  Failed to update status for reliable reader activity changed.
- #define RTPS_LOG_SET_GAP_EC (RTPS_LOG_BASE + 87)

  Failed to update status for reliable reader activity changed.
- #define RTPS_LOG_WINDOW_POOL_ALLOC_EC (RTPS_LOG_BASE + 88)

  Out of memory to allocate send window pool.
- #define RTPS_LOG_GET_WINDOW_BUFFER_EC (RTPS_LOG_BASE + 89)

  Failed to get buffer from send window pool.
- #define RTPS_LOG_INSERT_WINDOW_FULL_EC (RTPS_LOG_BASE + 90)

  Cannot insert sample into full window.
- #define RTPS_LOG_WINDOW_INSERT_EC (RTPS_LOG_BASE + 91)

  Failed to insert sample into send window.
- #define RTPS_LOG_TIMER_DELETE_TIMEOUT_EC (RTPS_LOG_BASE + 92)

  Failed to delete a timeout event.
- #define RTPS_LOG_BUFFERPOOL_DELETE_EC (RTPS_LOG_BASE + 93)

  Failed to delete a buffer pool.
- #define RTPS_LOG_DIRECT_SEND_EC (RTPS_LOG_BASE + 94)

  Failed to send a message to a specific peer.
- #define RTPS_LOG_PACKET_INITIALIZE_EC (RTPS_LOG_BASE + 95)

  RTPS writer failed to initialize a packet.
- #define RTPS_LOG_WRITER_REQUEST_SENT_HISTORY_EC (RTPS_LOG_BASE + 96)

  RTPS reliable writer failed to request and resend history.
- #define RTPS_LOG_BITMAP_SHIFT_EC (RTPS_LOG_BASE + 97)

  Failed to shift bitmap to new lead sequence number.
- #define RTPS_LOG_WINDOW_ADVANCE_EC (RTPS_LOG_BASE + 98)

  Send window of a remote reader failed to advance.
- #define RTPS_LOG_WINDOW_ADVANCE_UNACKED_EC (RTPS_LOG_BASE + 99)

  Failed when advancing window ahead of unacknowledged sequence number.
- #define RTPS_LOG_LOOKUP_PEER_EC (RTPS_LOG_BASE + 100)

  Failed to find peer that should exist.
- #define RTPS_LOG_READER_SEND_ACKNACK_EC (RTPS_LOG_BASE + 101)

  RTPS reader failed to send an ACKNACK message.
- #define RTPS_LOG_FINALIZE_INTERFACE_EC (RTPS_LOG_BASE + 102)

  Failed to finalize an RTPS interface.
- #define RTPS_LOG_INDEXER_REMOVE_ENTRY_EC (RTPS_LOG_BASE + 103)

  Failed to remove index entry for external interface upon deleting RTPS interface.
- #define RTPS_LOG_LOOKUP_EXT_BIND_ENTRY_EC (RTPS_LOG_BASE + 104)

  Failed when looking up external bind entry from database.
- #define RTPS_LOG_SEQ_INIT_EC (RTPS_LOG_BASE + 105)

  Failed to initialize a local sequence.

- #define RTPS_LOG_SEQ_MAX_EC (RTPS_LOG_BASE + 106)

  Failed to set maximum of a local sequence.
- #define RTPS_LOG_SEQ_LEN_EC (RTPS_LOG_BASE + 107)

  Failed to set length of a local sequence.
- #define RTPS_LOG_INTF_MODE_UNDEF_EC (RTPS_LOG_BASE + 108)

  Initializing an RTPS interface with undefined mode.
- #define RTPS_LOG_DELETE_BUFFER_POOL_EC (RTPS_LOG_BASE + 109)

  Failed to delete buffer pool.
- #define RTPS_LOG_CREATE_BUFFER_POOL_EC (RTPS_LOG_BASE + 110)

  Failed to create buffer pool.
- #define RTPS_LOG_CREATE_PEERS_INDEX_EC (RTPS_LOG_BASE + 111)

  Failed to create index of peers.
- #define RTPS_LOG_DELETE_INDEXER_EC (RTPS_LOG_BASE + 112)

  Failed to delete indexer.
- #define RTPS_LOG_GET_PEER_BUFFER_EC (RTPS_LOG_BASE + 113)

  Failed to get buffer from peer buffer pool.
- #define RTPS_LOG_DB_CREATE_DIRECT_INDEX_EC (RTPS_LOG_BASE + 114)

  Failed to create database index for direct sends.
- #define RTPS_LOG_DB_CREATE_GROUP_INDEX_EC (RTPS_LOG_BASE + 115)

  Failed to create database index for group sends.
- #define RTPS_LOG_NETIO_PACKET_INSUFFICIENT_BUFFER_EC (RTPS_LOG_BASE + 116)

  Insufficient buffer in packet.
- #define RTPS_LOG_PROCESS_DATA_BATCH_EC (RTPS_LOG_BASE + 117)

  Failed to process received DATA BATCH submessage.
- #define RTPS_LOG_PROCESS_HEARTBEAT_BATCH_EC (RTPS_LOG_BASE + 118)

  Failed to process received HEARTBEAT_BATCH submessage.
- #define RTPS_LOG_NEGATIVE_RESERVATION_COUNT_EC (RTPS_LOG_BASE + 119)

  The reservation count it zero.
- #define RTPS_LOG_DB_CREATE_SELECTED_GROUP_INDEX_EC (RTPS_LOG_BASE + 148)

  Failed to create database index for the selected route group.
- #define RTPS_LOG_CHECKSUM_MISSING_CHECKSUM_LENGTH_EC (RTPS_LOG_BASE + 149)

  A header extension was received, but the checksum length is not present.
- #define RTPS_LOG_CHECKSUM_UNSUPPORTED_EC (RTPS_LOG_BASE + 150)

  Unsupported checksum bits received, message dropped.
- #define RTPS_LOG_CHECKSUM_LENGTH_DESERIALIZE_ERROR_EC (RTPS_LOG_BASE + 151)

  Checksum length could not be deserialized.
- #define RTPS_LOG_CHECKSUM_INCONSISTENT_LENGTH_EC (RTPS_LOG_BASE + 152)

  The payload length is less than the deserialized checksum length.
- #define RTPS_LOG_CHECKSUM_CHECKSUM_DESERIALIZE_ERROR_EC (RTPS_LOG_BASE + 153)

  Checksum length could not be deserialized.
- #define RTPS_LOG_CHECKSUM_CHECKSUM_ERROR_EC (RTPS_LOG_BASE + 154)

  The checksum is invalid.
- #define RTPS_LOG_CHECKSUM_INVALID_CRC32_FLAGS_EC (RTPS_LOG_BASE + 155)

  The CRC32 submsg flags are invalid.
- #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM128_EC (RTPS_LOG_BASE + 200)

  The property tries to override the BUILTIN128 checksum without setting allow_builtin_override to TRUE.
- #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM32_EC (RTPS_LOG_BASE + 201)

  The property tries to override the BUILTIN32 checksum without setting allow_builtin_override to TRUE.
- #define RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM64_EC (RTPS_LOG_BASE + 202)

  The property tries to override the BUILTIN64 checksum without setting allow_builtin_override to TRUE.

- #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN128_EC (RTPS_LOG_BASE + 203)

  The property override of the BUILTIN128 checksum is invalid. It is not legal to reuse any of the built-in functions.
- #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN32_EC (RTPS_LOG_BASE + 204)

  The property override of the BUILTIN32 checksum is invalid. It is not legal to reuse any of the built-in functions functions.
- #define RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN64_EC (RTPS_LOG_BASE + 205)

  The property override of the BUILTIN64 checksum is invalid. It is not legal to reuse any of the built-in functions functions.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM256_EC (RTPS_LOG_BASE + 206)

  The property specifies an invalid custom256 checksum function.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM32_EC (RTPS_LOG_BASE + 207)

  The property specifies an invalid custom32 checksum function.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM64_EC (RTPS_LOG_BASE + 208)

  The property specifies an invalid custom64 checksum function.
- #define RTPS_LOG_CHECKSUM_INVALID_CUSTOM128_EC (RTPS_LOG_BASE + 209)

  The property specifies an invalid custom128 checksum function.
- #define RTPS_LOG_CHECKSUM_CHECKSUM_CALCULATION_FAILED_EC (RTPS_LOG_BASE + 210)

  RTPS_Interface_calculate_checksum() failed to calculate the checksum.
- #define RTPS_LOG_UNSUPPORTED_MANDTORY_HDR_EXT_PID_EC (RTPS_LOG_BASE + 211)

  The HeaderExtension contained a mandatory PID that was not understood.
- #define RTPS_LOG_INVALID_HDR_EXT_PID_LIST_EC (RTPS_LOG_BASE + 212)

  An invalid HeaderExtension PID was detected.
- #define RTPS_LOG_INVALID_HDR_EXT_PID_ERROR_EC (RTPS_LOG_BASE + 213)

  Failed to decode the HeaderExtension PID list.
- #define RTPS_LOG_INCONSISTENT_HDR_EXT_LENGTH_ERROR_EC (RTPS_LOG_BASE + 214)

  The HeaderExtension length is isconsistent with the decoded encoded HeaderExtension length.
- #define RTPS_LOG_MESSAGE_EXCEED_LENGTH_ERROR_EC (DDSC_LOG_BASE + 215)

  A RTPS message which was longer than the indicated message length in the RTPS header was received. This causes the rest of the RTPS message to be invalidated and the remaining submessages to be ignored.

7.66.1 Detailed Description

RTPS module log codes. Log codes of the RTPS module

7.66.2 Macro Definition Documentation

7.66.2.1 #define RTPS_LOG_DB_REMOVE_ROUTE_ENTRY_EC (RTPS_LOG_BASE + 14)

Failed to remove entry from route table.

Failure reason given by database return code (dbrc)

7.67 rtps_rtps_impl.h File Reference

Implementation of RTPS interface functions and types.

7.67.1 Detailed Description

Implementation of RTPS interface functions and types. RTPS protocol defined types, implemented in C.

7.68 wh_sm_dll.h File Reference

WriterHistory DLL external linkage.

7.68.1 Detailed Description

WriterHistory DLL external linkage.

7.69 wh_sm_history.h File Reference

Writer History API.

Functions

- struct RT_ComponentFactoryI * WHSM_HistoryFactory_get_interface (void)
  Gets the singleton instance of the WHSM writer history factory.

7.69.1 Detailed Description

Writer History API.

7.70 wh_sm_log.h File Reference

WH module log codes.

Macros

- #define WHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC (WHSM_LOG_BASE + 1)
  KEEP_ALL History kind is not supported.
- #define WHSM_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC (WHSM_LOG_BASE + 2)
  Unlimited length resource limits unsupported.
- #define WHSM_LOG_MAX_SAMPLES_TOO_SMALL_EC (WHSM_LOG_BASE + 3)
  DataWriterQos.resource_limits.max_samples set too small.
- #define WHSM_LOG_HISTORY_OBJECT 1
  A history object.
- #define WHSM_LOG_KEY_OBJECT 2
  A key object.
- #define WHSM_LOG_SAMPLE_OBJECT 2
  A sample object.
- #define WHSM_LOG_KEYPOOL_OBJECT 3
  A key-pool object.
- #define WHSM_LOG_SAMPLEPOOL_OBJECT 4
  A sample-pool object.
- #define WHSM_LOG_KEYINDEX_OBJECT 5
  A key index pool.
- #define WHSM_LOG_HISTORYINDEX_OBJECT 6
  A key index pool.
- #define WHSM_LOG_SAMPLEINDEX_OBJECT 5
  A key index pool.
- #define WHSM_LOG_OBJECT_ALLOCATE_EC (WHSM_LOG_BASE + 4)

Failed to allocate object of the specified kind.

- #define WHSM_LOG_OBJECT_DELETE_EC (WHSM_LOG_BASE + 5)

  Failed to delete object of the specified kind.
- #define WHSM_LOG_OBJECT_INDEX_EC (WHSM_LOG_BASE + 6)

  Failed to index an object of the specified kind.
- #define WHSM_LOG_NO_PROPERTY_EC (WHSM_LOG_BASE + 7)

  No property when creating a writer history instance.
- #define WHSM_LOG_INVALID_INDEX_OBJECT_EC (WHSM_LOG_BASE + 8)

  The sample removed from an index did not match sample's key.

7.70.1 Detailed Description

WH module log codes.

7.70.2 Macro Definition Documentation

7.70.2.1 #define WHSM_LOG_HISTORY_OBJECT 1

A history object.

7.70.2.2 #define WHSM_LOG_KEY_OBJECT 2

A key object.

7.70.2.3 #define WHSM_LOG_SAMPLE_OBJECT 2

A sample object.

7.70.2.4 #define WHSM_LOG_KEYPOOL_OBJECT 3

A key-pool object.

7.70.2.5 #define WHSM_LOG_SAMPLEPOOL_OBJECT 4

A sample-pool object.

7.70.2.6 #define WHSM_LOG_KEYINDEX_OBJECT 5

A key index pool.

7.70.2.7 #define WHSM_LOG_HISTORYINDEX_OBJECT 6

A key index pool.

7.70.2.8 #define WHSM_LOG_SAMPLEINDEX_OBJECT 5

A key index pool.

Index

_contiguous_buffer
  FooSeq, 445
_element_size
  FooSeq, 445
_flags
  FooSeq, 445
_length
  FooSeq, 445
_maximum
  FooSeq, 445
_token1
  FooSeq, 445
_token2
  FooSeq, 445

accept_unknown_peers
  DDS_DiscoveryQosPolicy, 377
access_scope
  DDS_PresentationQosPolicy, 406
active_count
  DDS_ReliableReaderActivityChangedStatus, 414
active_count_change
  DDS_ReliableReaderActivityChangedStatus, 414
add_route
  ZCOPY_NotifUserInterfaceI, 464
address
  DDS_Locator_t, 396
  NETIO_AddressValue, 449
  NETIO_DGRAM_InterfaceRouteEntry, 452
  NETIO_DGRAM_InterfaceTableEntry, 452
  UDP_InterfaceTableEntry, 462
address_high
  NETIO_DGRAM_InterfaceMultiCastGroup, 451
address_low
  NETIO_DGRAM_InterfaceMultiCastGroup, 451
alive_count
  DDS_LivelinessChangedStatus, 392
alive_count_change
  DDS_LivelinessChangedStatus, 392
allow_builtin_override
  RTPS Transport API, 136
allow_interface
  UDP_InterfaceFactoryProperty, 461
allowed_crc_mask
  DDS_ChecksumProperty, 359
  DDS_ChecksumProperty_t, 359
  DDS_WireProtocolQosPolicy, 441
as_datareaderlistener
  DDS_SubscriberListener, 430
as_datawriterlistener
  DDS_PublisherListener, 412
as_listener
  DDS_DataReaderListener, 362
  DDS_DataWriterListener, 368
  DDS_TopicListener, 436
as_publisherlistener
  DDS_DomainParticipantListener, 379
as_subscriberlistener
  DDS_DomainParticipantListener, 379
as_topiclistener
  DDS_DomainParticipantListener, 379
as_uint32
  NETIO_AddressValue, 449
autoenable_created_entities
  DDS_EntityFactoryQosPolicy, 385

bind
  ZCOPY_NotifUserInterfaceI, 464
bind_address
  NETIO_DGRAM_InterfaceI, 450
bits
  NETIO_Netmask, 454
builtin_checksum128_class
  RTPS Transport API, 135
builtin_checksum32_class
  RTPS Transport API, 135
builtin_checksum64_class
  RTPS Transport API, 135
builtin_multicast_port_offset
  DDS_RtpsWellKnownPorts_t, 424
builtin_unicast_port_offset
  DDS_RtpsWellKnownPorts_t, 424

CDR, 187
  CDR_LOG_INCR_OFFSET_EC, 187
  CDR_LOG_SET_OFFSET_EC, 187
  CDR_LOG_STREAM_ALLOC_EC, 187
  CDR_LOG_UNSUPPORTED_OPTIONS_EC, 187
CDR Stream API, 125
  CDR_Stream_check_size, 126
  CDR_Stream_get_current_position_offset, 125
  CDR_Stream_get_current_position_ptr, 126
  CDR_Stream_increment_current_position, 125
  CDR_Stream_is_byte_swapped, 126
  CDR_Stream_set_current_position_offset, 125
CDR_LOG_INCR_OFFSET_EC
  CDR, 187
CDR_LOG_SET_OFFSET_EC
  CDR, 187
CDR_LOG_STREAM_ALLOC_EC
  CDR, 187
CDR_LOG_UNSUPPORTED_OPTIONS_EC
  CDR, 187
CDR_Stream_check_size
  CDR Stream API, 126
CDR_Stream_get_current_position_offset
  CDR Stream API, 125
CDR_Stream_get_current_position_ptr
  CDR Stream API, 126
CDR_Stream_increment_current_position
  CDR Stream API, 125

CDR_Stream_is_byte_swapped
CDR Stream API, 126
CDR_Stream_set_current_position_offset
CDR Stream API, 125
cdr_log.h, 465
check_crc
DDS_WireProtocolQosPolicy, 441
checksum
DDS_ParticipantBuiltinTopicData, 403
RTPS Transport API, 136
checksum128
RTPS Transport API, 135
checksum32
RTPS Transport API, 135
checksum64
RTPS Transport API, 135
checksum_calculate
RTPS Transport API, 135
checksum_tx_mode
RTPS Transport API, 136
class_id
RTPS Transport API, 135
coherent_access
DDS_PresentationQosPolicy, 406
compute_crc
DDS_WireProtocolQosPolicy, 441
computed_crc_kind
DDS_ChecksumProperty, 359
DDS_ChecksumProperty_t, 359
DDS_WireProtocolQosPolicy, 441
Conditions and WaitSets, 274
DDS_Condition, 275
DDS_Condition_get_trigger_value, 276
DDS_GuardCondition, 275
DDS_GuardCondition_as_condition, 277
DDS_GuardCondition_new, 277
DDS_GuardCondition_set_trigger_value, 277
DDS_StatusCondition, 275
DDS_StatusCondition_as_condition, 277
DDS_StatusCondition_get_enabled_statuses, 278
DDS_StatusCondition_get_entity, 278
DDS_StatusCondition_set_enabled_statuses, 278
DDS_WaitSet, 275
DDS_WaitSet_attach_condition, 280
DDS_WaitSet_get_conditions, 280
DDS_WaitSet_new, 279
DDS_WaitSet_wait, 279
context
RTPS Transport API, 135
copy_sample
NDDS_Type_Plugin, 447
create_instance
NETIO_DGRAM_InterfaceI, 450
ZCOPY_NotifUserInterfaceI, 464
create_sample
NDDS_Type_Plugin, 447
current_count
DDS_PublicationMatchedStatus, 411
DDS_SubscriptionMatchedStatus, 434
current_count_change
DDS_PublicationMatchedStatus, 411
DDS_SubscriptionMatchedStatus, 434

DATA_READER_PROTOCOL, 270
DDS_DATAREADERPROTOCOL_QOS_POLICY-_NAME, 270
DATA_WRITER_PROTOCOL, 271
DDS_DATAWRITERPROTOCOL_QOS_POLICY-_NAME, 271
DB, 183
DB_LOG_ALLOC_CURSOR_POOL_EC, 185
DB_LOG_ALLOC_DATABASE_EC, 185
DB_LOG_ALLOC_INDEX_POOL_EC, 185
DB_LOG_ALLOC_RECORD_POOL_EC, 185
DB_LOG_ALLOC_TABLE_POOL_EC, 184
DB_LOG_CURSOR_INVALIDATED_EC, 185
DB_LOG_CURSORS_INUSE_EC, 185
DB_LOG_ILLEGAL_LOCK_MODE_EC, 184
DB_LOG_ILLEGAL_RECORD_COUNT_EC, 184
DB_LOG_ILLEGAL_TABLE_SIZE_EC, 184
DB_LOG_INDEX_INUSE_EC, 185
DB_LOG_LOCK_FAILURE_EC, 185
DB_LOG_MUTEX_ALLOC_EC, 184
DB_LOG_NAME_TOO_LONG_EC, 184
DB_LOG_OUT_OF_CURSORS_EC, 185
DB_LOG_OUT_OF_INDICES_EC, 185
DB_LOG_OUT_OF_RECORDS_EC, 184
DB_LOG_OUT_OF_TABLES_EC, 184
DB_LOG_RECORD_ALREADY_EXISTS_EC, 185
DB_LOG_RECORD_DOES_NOT_EXIST_EC, 185
DB_LOG_RECORDS_INUSE_EC, 185
DB_LOG_SORTED_ALLOC_EC, 184
DB_LOG_TABLE_EXISTS_EC, 184
DB_LOG_TABLE_NAME_TOO_LONG_EC, 184
DB_LOG_TABLE_NOT_INUSE_EC, 185
DB_LOG_TABLES_INUSE_EC, 184
DB_LOG_UNLOCK_FAILURE_EC, 185
DB_LOG_ALLOC_CURSOR_POOL_EC
DB, 185
DB_LOG_ALLOC_DATABASE_EC
DB, 185
DB_LOG_ALLOC_INDEX_POOL_EC
DB, 185
DB_LOG_ALLOC_RECORD_POOL_EC
DB, 185
DB_LOG_ALLOC_TABLE_POOL_EC
DB, 184
DB_LOG_CURSOR_INVALIDATED_EC
DB, 185
DB_LOG_CURSORS_INUSE_EC
DB, 185
DB_LOG_ILLEGAL_LOCK_MODE_EC
DB, 184
DB_LOG_ILLEGAL_RECORD_COUNT_EC
DB, 184
DB_LOG_ILLEGAL_TABLE_SIZE_EC

DB, 184
DB_LOG_INDEX_INUSE_EC
DB, 185
DB_LOG_LOCK_FAILURE_EC
DB, 185
DB_LOG_MUTEX_ALLOC_EC
DB, 184
DB_LOG_NAME_TOO_LONG_EC
DB, 184
DB_LOG_OUT_OF_CURSORS_EC
DB, 185
DB_LOG_OUT_OF_INDICES_EC
DB, 185
DB_LOG_OUT_OF_RECORDS_EC
DB, 184
DB_LOG_OUT_OF_TABLES_EC
DB, 184
DB_LOG_RECORD_ALREADY_EXISTS_EC
DB, 185
DB_LOG_RECORD_DOES_NOT_EXIST_EC
DB, 185
DB_LOG_RECORDS_INUSE_EC
DB, 185
DB_LOG_SORTED_ALLOC_EC
DB, 184
DB_LOG_TABLE_EXISTS_EC
DB, 184
DB_LOG_TABLE_NAME_TOO_LONG_EC
DB, 184
DB_LOG_TABLE_NOT_INUSE_EC
DB, 185
DB_LOG_TABLES_INUSE_EC
DB, 184
DB_LOG_UNLOCK_FAILURE_EC
DB, 185
DDS API, 115
DDS-Specific Primitive Types, 222
DDS_BOOLEAN_FALSE, 223
DDS_BOOLEAN_TRUE, 223
DDS_Boolean, 224
DDS_Char, 223
DDS_Double, 224
DDS_Enum, 224
DDS_Float, 224
DDS_Long, 223
DDS_LongDouble, 224
DDS_LongLong, 223
DDS_Octet, 223
DDS_Short, 223
DDS_String, 224
DDS_UnsignedLong, 223
DDS_UnsignedLongLong, 223
DDS_UnsignedShort, 223
DDS_Wchar, 223
DDS_Wstring, 224
DDS_ALIVE_INSTANCE_STATE
Instance States, 107
DDS_ANY_INSTANCE_STATE
Instance States, 107
DDS_ANY_SAMPLE_STATE
Sample States, 103
DDS_ANY_VIEW_STATE
View States, 105
DDS_AUTO_NAME_ENTITY
ENTITY_NAME, 287
DDS_AUTOMATIC_LIVELINESS_QOS
LIVELINESS, 253
DDS_BEST_EFFORT_RELIABILITY_QOS
RELIABILITY, 255
DDS_BOOLEAN_FALSE
DDS-Specific Primitive Types, 223
DDS_BOOLEAN_TRUE
DDS-Specific Primitive Types, 223
DDS_BUILTINTOPICKEY_AUTO
dds_c_infrastructure.h, 480
DDS_BUILTINTOPICKEY_UNKNOWN
dds_c_infrastructure.h, 480
DDS_BY_RECEPTION_TIMESTAMP_DESTINATION-ORDER_QOS
DESTINATION_ORDER, 260
DDS_BY_SOURCE_TIMESTAMP_DESTINATIONOR-DER_QOS
DESTINATION_ORDER, 260
DDS_Boolean
DDS-Specific Primitive Types, 224
DDS_BuiltinTopicKey_t, 358
Discovery, 34
value, 358
DDS_C, 288
DDSC_LOG_ADDRESSRESOLVER_OBJECT, 306
DDSC_LOG_BIND_RECORD, 304
DDSC_LOG_BINDRESOLVER_OBJECT, 306
DDSC_LOG_CDR_BUFFER_SET_EC, 315
DDSC_LOG_CDR_DESERIALIZE_DATA_EC, 316
DDSC_LOG_CDR_DESERIALIZE_HEADER_EC, 316
DDSC_LOG_CDR_DESERIALIZE_KEY_EC, 316
DDSC_LOG_CDR_DESERIALIZE_KEYHASH_E-C, 316
DDSC_LOG_CDR_DESERIALIZE_PID_EC, 316
DDSC_LOG_CDR_DESERIALIZE_PID_LENGTH-_EC, 316
DDSC_LOG_CDR_FINALIZE_SAMPLE_EC, 316
DDSC_LOG_CDR_INCREMENT_POS_EC, 316
DDSC_LOG_CDR_INITIALIZE_SAMPLE_EC, 316
DDSC_LOG_CDR_POOL_ALLOC_EC, 315
DDSC_LOG_CDR_POOL_DELETE_EC, 315
DDSC_LOG_CDR_SERIALIZE_DATA_EC, 316
DDSC_LOG_CDR_SERIALIZE_KEYHASH_EC, 316
DDSC_LOG_CDR_SERIALIZE_PID_EC, 315
DDSC_LOG_CDR_SERIALIZE_PID_LENGTH_E-C, 315

DDSC_LOG_CDR_SERIALIZE_STATUS_INFO_-EC, 316
DDSC_LOG_CDR_SET_POS_EC, 316
DDSC_LOG_CHECKSUM_INCOMPATIBLE_EC, 321
DDSC_LOG_CHECKSUM_INCONSISTENT_ALLOW_EC, 321
DDSC_LOG_CHECKSUM_INCONSISTENT_CLASS_EC, 321
DDSC_LOG_CHECKSUM_INCONSISTENT_COMPUTE_EC, 321
DDSC_LOG_CHECKSUM_REQUIRED_EC, 321
DDSC_LOG_COMPONENT_CREATE_EC, 310
DDSC_LOG_COMPONENT_DELETE_EC, 310
DDSC_LOG_COMPONENT_LOOKUP_EC, 310
DDSC_LOG_CONDITION_OBJECT, 307
DDSC_LOG_CONDREF_OBJECT, 306
DDSC_LOG_CREATE_INDEX_EC, 303
DDSC_LOG_DATABASE_CREATE_EC, 303
DDSC_LOG_DATABASE_DELETE_EC, 303
DDSC_LOG_DATAREADER_CDR, 315
DDSC_LOG_DATAREADER_ENTITY, 314
DDSC_LOG_DATAREADER_LISTENER, 313
DDSC_LOG_DATAREADER_NETIO_KIND, 317
DDSC_LOG_DATAREADER_QOS, 311
DDSC_LOG_DATAREADER_RECORD, 303
DDSC_LOG_DATAREADER_RESOURCE_QOS-_POLICY, 312
DDSC_LOG_DATAREADERIO_COMPONENT, 310
DDSC_LOG_DATAREADERIO_OBJECT, 306
DDSC_LOG_DATAREADERQOS_OBJECT, 305
DDSC_LOG_DATAWRITER_CDR, 315
DDSC_LOG_DATAWRITER_ENTITY, 314
DDSC_LOG_DATAWRITER_LISTENER, 314
DDSC_LOG_DATAWRITER_NETIO_KIND, 317
DDSC_LOG_DATAWRITER_QOS, 311
DDSC_LOG_DATAWRITER_RECORD, 303
DDSC_LOG_DATAWRITER_RESOURCE_QOS-_POLICY, 312
DDSC_LOG_DATAWRITERIO_COMPONENT, 310
DDSC_LOG_DATAWRITERIO_OBJECT, 306
DDSC_LOG_DATAWRITERQOS_OBJECT, 305
DDSC_LOG_DB_CURSOR_INVALIDATED_EC, 303
DDSC_LOG_DEADLINE_QOS_POLICY, 311
DDSC_LOG_DEFAULTDATAREADER_QOS, 311
DDSC_LOG_DEFAULTDATAWRITER_QOS, 311
DDSC_LOG_DEFAULTPARTICIPANT_QOS, 311
DDSC_LOG_DEFAULTPUBLISHER_QOS, 311
DDSC_LOG_DEFAULTSUBSCRIBER_QOS, 311
DDSC_LOG_DEFAULTTOPIC_QOS, 311
DDSC_LOG_DELETE_INDEX_EC, 303
DDSC_LOG_DESERIALIZE_BAD_PID_LENGTH-_EC, 316
DDSC_LOG_DESTINATION_ORDER_POLICY, 312
DDSC_LOG_DESTINATION_SEQUENCE, 308
DDSC_LOG_DISC_ADD_PEER_EC, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_DELETED_EC, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_ENABLED_EC, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_DELETED_EC, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_ENABLED_EC, 321
DDSC_LOG_DISC_AFTER_LOCAL_PARTICIPANT_CREATED_EC, 321
DDSC_LOG_DISC_BEFORE_LOCAL_PARTICIPANT_CREATED_EC, 321
DDSC_LOG_DISC_LOCAL_PARTICIPANT_ENABLED_EC, 321
DDSC_LOG_DISC_REMOTE_PARTICIPANT_EXPIRED_EC, 321
DDSC_LOG_DISCOVERY_COMPONENT, 309
DDSC_LOG_DISCOVERY_TRAFFIC_QOS_POLICY, 313
DDSC_LOG_DR_COMMIT_ENTRY_EC, 319
DDSC_LOG_DR_COMMIT_SAMPLE_EC, 319
DDSC_LOG_DR_COPY_DATA_SAMPLE_EC, 319
DDSC_LOG_DR_CREATE_TYPED_READER_EC, 319
DDSC_LOG_DR_DESERIALIZE_KEYHASH_EC, 319
DDSC_LOG_DR_DISPOSE_KEY_EC, 319
DDSC_LOG_DR_FILTER_ERROR_EC, 319
DDSC_LOG_DR_GET_ENTRY_FAILED_EC, 319
DDSC_LOG_DR_INSTANCE_TO_KEYHASH_EC, 319
DDSC_LOG_DR_READ_TAKE_FAILURE_EC, 319
DDSC_LOG_DR_UNREGISTER_KEY_EC, 319
DDSC_LOG_DURABILITY_QOS_POLICY, 312
DDSC_LOG_DW_ACKNACK_FAILED_EC, 318
DDSC_LOG_DW_COMMIT_EC, 318
DDSC_LOG_DW_CREATE_TYPED_WRITER_EC, 319
DDSC_LOG_DW_HISTORY_REGISTER_KEY_EC, 319
DDSC_LOG_DW_ILLEGAL_KEY_KIND_EC, 319
DDSC_LOG_DW_KEYHASH_CREATE_EC, 318
DDSC_LOG_ENABLED_DISCVOERY_TRANSPORT_SEQUENCE, 308
DDSC_LOG_ENABLED_TRANSPORT_SEQUENCE, 308
DDSC_LOG_ENABLED_USER_TRANSPORT_SEQUENCE, 308
DDSC_LOG_ENDPOINT_NOT_CHILD_OF_PARTICIPANT_EC, 301
DDSC_LOG_ENTITY_DIFFERENT_FACTORY_-EC, 315
DDSC_LOG_ENTITY_ENABLE_EC, 315
DDSC_LOG_ENTITY_FINALIZE_EC, 315

DDSC_LOG_ENTITY_INITIALIZE_EC, 315
DDSC_LOG_ENTITY_NOT_EMPTY_EC, 315
DDSC_LOG_ENTITY_NOT_ENABLED_EC, 315
DDSC_LOG_ENTITY_OBJECT, 306
DDSC_LOG_FAILED_UPDATE_STATUS_CONDITION_EC, 302
DDSC_LOG_FIND_PUBLICATION_PARENT_EC, 301
DDSC_LOG_FIND_SUBSCRIPTION_PARENT_EC, 301
DDSC_LOG_GET_NEXT_OBJECT_ID_EC, 300
DDSC_LOG_HEARTBEATS_QOS_POLICY, 312
DDSC_LOG_HISTORY_QOS_POLICY, 312
DDSC_LOG_HISTORY_RESOURCE, 310
DDSC_LOG_ILLEGAL_OBJECTID_EC, 302
DDSC_LOG_INITIAL_PEER_SEQUENCE, 308
DDSC_LOG_INTRA_NETIO_KIND, 317
DDSC_LOG_INVALID_DOMAINID_EC, 302
DDSC_LOG_INVALID_DURATION_EC, 300
DDSC_LOG_INVALID_ENDPOINT_GUID_EC, 301
DDSC_LOG_INVALID_PARTICIPANT_GUID_PREFIX_EC, 300
DDSC_LOG_INVALID_PARTICIPANT_NAME_EC, 300
DDSC_LOG_IO_SNPRINTF_FAILED_EC, 301
DDSC_LOG_LISTENER_GET_EC, 314
DDSC_LOG_LISTENER_INCONSISTENT_EC, 314
DDSC_LOG_LISTENER_SET_EC, 314
DDSC_LOG_LISTENER_SET_ILLEGAL_NULL_EC, 314
DDSC_LOG_LIVELINESS_QOS_POLICY, 311
DDSC_LOG_LOG_OBJECT, 306
DDSC_LOG_LOOKUP_TYPE_PLUGIN_EC, 320
DDSC_LOG_MAX_PARTICIPANT_ID_REACHED_EC, 302
DDSC_LOG_MD5STREAM_OBJECT, 306
DDSC_LOG_METAMULTICAST_SEQUENCE, 309
DDSC_LOG_METAUNICAST_SEQUENCE, 308
DDSC_LOG_MUTEX_OBJECT, 306
DDSC_LOG_NETIO_ADD_ANON_TOPIC_ROUTE_EC, 316
DDSC_LOG_NETIO_ADD_ROUTE_EC, 317
DDSC_LOG_NETIO_ADD_TOPIC_ROUTE_EC, 317
DDSC_LOG_NETIO_BIND_EC, 317
DDSC_LOG_NETIO_BIND_EXTERNAL_EC, 317
DDSC_LOG_NETIO_COMPONENT, 310
DDSC_LOG_NETIO_DELETE_ROUTE_EC, 317
DDSC_LOG_NETIO_DELETE_TOPIC_ROUTE_EC, 317
DDSC_LOG_NETIO_DISCOVERY_ENABLED_TRANSPORT_EC, 318
DDSC_LOG_NETIO_FORCED_REMOVE_EC, 318
DDSC_LOG_NETIO_FORWARD_TOPIC_EC, 317
DDSC_LOG_NETIO_GET_EXTERNAL_INTF_EC, 317
DDSC_LOG_NETIO_GET_ROUTE_TABLE_FAILED_EC, 318
DDSC_LOG_NETIO_NETIO_KIND, 317
DDSC_LOG_NETIO_NO_ROUTE_EC, 318
DDSC_LOG_NETIO_PEER_LOOKUP_EC, 318
DDSC_LOG_NETIO_ROUTE_LOOKUP_FAILED_EC, 318
DDSC_LOG_NETIO_SEND_FAILED_EC, 318
DDSC_LOG_NETIO_SET_STATE_EC, 318
DDSC_LOG_NETIO_UNBIND_EC, 317
DDSC_LOG_NETIO_UNBIND_EXTERNAL_EC, 317
DDSC_LOG_NETMASK_SEQUENCE, 308
DDSC_LOG_OBJECT_ALLOCATE_EC, 307
DDSC_LOG_OBJECT_COPY_EC, 307
DDSC_LOG_OBJECT_CREATE_EC, 308
DDSC_LOG_OBJECT_DELETE_EC, 307
DDSC_LOG_OBJECT_EMPTY_EC, 308
DDSC_LOG_OBJECT_FINALIZE_EC, 307
DDSC_LOG_OBJECT_GET_PROPERTY_EC, 308
DDSC_LOG_OBJECT_INITIALIZE_EC, 307
DDSC_LOG_OBJECT_INUSECOUNT_EC, 308
DDSC_LOG_OBJECT_REFCOUNT_EC, 308
DDSC_LOG_OBJECT_SET_PROPERTY_EC, 308
DDSC_LOG_OWNERSHIP_QOS_POLICY, 312
DDSC_LOG_OWNERSHIP_STRENGTH_QOS_POLICY, 312
DDSC_LOG_PACKET_INIT_EC, 318
DDSC_LOG_PACKET_SET_HEAD_EC, 318
DDSC_LOG_PACKET_SET_TAIL_EC, 318
DDSC_LOG_PARTICIPANT_DOES_NOT_EXIST_EC, 301
DDSC_LOG_PARTICIPANT_ENTITY, 314
DDSC_LOG_PARTICIPANT_ID_QOS_POLICY, 312
DDSC_LOG_PARTICIPANT_LISTENER, 314
DDSC_LOG_PARTICIPANT_LOOKUP_EC, 301
DDSC_LOG_PARTICIPANT_POOL_OBJECT, 307
DDSC_LOG_PARTICIPANT_QOS, 311
DDSC_LOG_PARTICIPANT_RECORD, 303
DDSC_LOG_PARTICIPANT_RESOURCES, 310
DDSC_LOG_PARTICIPANTDATA_OBJECT, 305
DDSC_LOG_PARTICIPANTFACTORY_LISTENER, 314
DDSC_LOG_PARTICIPANTFACTORY_QOS, 311
DDSC_LOG_PARTICIPANTFACTORYQOS_OBJECT, 306
DDSC_LOG_PARTICIPANTQOS_OBJECT, 306
DDSC_LOG_PROTOCOL_QOS_POLICY, 312
DDSC_LOG_PUBLICATION_ENTITY, 315
DDSC_LOG_PUBLICATION_RECORD, 303
DDSC_LOG_PUBLICATIONDATA_OBJECT, 305

DDSC_LOG_PUBLISHER_ENTITY, 314
DDSC_LOG_PUBLISHER_LISTENER, 314
DDSC_LOG_PUBLISHER_QOS, 311
DDSC_LOG_PUBLISHER_RECORD, 304
DDSC_LOG_PUBLISHERQOS_OBJECT, 305
DDSC_LOG_QOS_CHANGED_EC, 313
DDSC_LOG_QOS_COPY_EC, 313
DDSC_LOG_QOS_FINALIZE_EC, 313
DDSC_LOG_QOS_GET_EC, 313
DDSC_LOG_QOS_IMMUTABLE_EC, 313
DDSC_LOG_QOS_INCONSISTENT_EC, 313
DDSC_LOG_QOS_INCONSISTENT_POLICIES_-EC, 313
DDSC_LOG_QOS_INCONSISTENT_POLICY_E-C, 313
DDSC_LOG_QOS_INITIALIZE_EC, 313
DDSC_LOG_QOS_SET_EC, 313
DDSC_LOG_QOS_SET_ON_ENABLED_EC, 313
DDSC_LOG_READERHISTORY_COMPONENT, 309
DDSC_LOG_READTAKE_SEQUENCE, 309
DDSC_LOG_RECORD_CREATE_EC, 304
DDSC_LOG_RECORD_DELETE_EC, 304
DDSC_LOG_RECORD_ERROR_EC, 304
DDSC_LOG_RECORD_EXISTS_EC, 305
DDSC_LOG_RECORD_FINALIZE_EC, 305
DDSC_LOG_RECORD_INITIALIZE_EC, 305
DDSC_LOG_RECORD_INSERT_EC, 304
DDSC_LOG_RECORD_LOOKUP_EC, 305
DDSC_LOG_RECORD_NOT_EXISTS_EC, 305
DDSC_LOG_RECORD_REMOVE_EC, 305
DDSC_LOG_RECORD_SELECT_EC, 305
DDSC_LOG_REFRESH_REM_PARTICIPANT_E-C, 301
DDSC_LOG_REFRESH_REM_PARTICIPANT_T-IMEOUT_EC, 301
DDSC_LOG_RELEASE_META_MC_EC, 302
DDSC_LOG_RELEASE_META_UC_EC, 302
DDSC_LOG_RELEASE_USER_MC_EC, 302
DDSC_LOG_RELEASE_USER_UC_EC, 302
DDSC_LOG_RELIABILITY_QOS_POLICY, 312
DDSC_LOG_REMOTE_PARTICIPANT_KEY_NO-T_EQUAL_EC, 301
DDSC_LOG_REMOVE_PUBLICATION_EC, 301
DDSC_LOG_REMOVE_SUBSCRIPTION_EC, 301
DDSC_LOG_RESERVE_LOCATORS_EC, 302
DDSC_LOG_RESERVED_SEQUENCE, 308
DDSC_LOG_RESOLVE_LOCATORS_PRIORITY-_EC, 302
DDSC_LOG_RESOURCE_EXCEEDED_EC, 310
DDSC_LOG_RESOURCE_LIMIT_QOS_POLICY, 312
DDSC_LOG_ROUTE_RECORD, 304
DDSC_LOG_ROUTE_SEQUENCE, 308
DDSC_LOG_ROUTERESOLVER_OBJECT, 306
DDSC_LOG_RT_OBJECT, 307
DDSC_LOG_RTPS_COMPONENT, 309
DDSC_LOG_RTPS_NETIO_KIND, 317
DDSC_LOG_SAMPLE_RESOURCES, 310
DDSC_LOG_SEQUENCE_COPY_EC, 309
DDSC_LOG_SEQUENCE_FINALIZE_EC, 309
DDSC_LOG_SEQUENCE_GETREF_EC, 309
DDSC_LOG_SEQUENCE_INITIALIZE_EC, 309
DDSC_LOG_SEQUENCE_INVALID_EC, 309
DDSC_LOG_SEQUENCE_SETLENGTH_EC, 309
DDSC_LOG_SEQUENCE_SETMAX_EC, 309
DDSC_LOG_SET_ENTITY_NAME_EC, 300
DDSC_LOG_STRING_OBJECT, 307
DDSC_LOG_STRING_RECORD, 304
DDSC_LOG_STRINGMANAGER_OBJECT, 307
DDSC_LOG_SUBSCRIBER_ENTITY, 314
DDSC_LOG_SUBSCRIBER_LISTENER, 314
DDSC_LOG_SUBSCRIBER_QOS, 311
DDSC_LOG_SUBSCRIBER_RECORD, 304
DDSC_LOG_SUBSCRIBERQOS_OBJECT, 305
DDSC_LOG_SUBSCRIPTION_ENTITY, 315
DDSC_LOG_SUBSCRIPTION_RECORD, 303
DDSC_LOG_SUBSCRIPTIONDATA_OBJECT, 305
DDSC_LOG_SYS_GET_HOSTNAME_EC, 300
DDSC_LOG_SYS_GETTIME_EC, 300
DDSC_LOG_SYSTEM_OBJECT, 306
DDSC_LOG_TABLE_CREATE_EC, 303
DDSC_LOG_TABLE_DELETE_EC, 303
DDSC_LOG_TABLE_INUSE_EC, 303
DDSC_LOG_TABLE_SELECT_EC, 303
DDSC_LOG_TIMER_CREATE_TIMEOUT_EC, 302
DDSC_LOG_TIMER_OBJECT, 307
DDSC_LOG_TIMEROUT_OBJECT, 307
DDSC_LOG_TOPIC_ENTITY, 314
DDSC_LOG_TOPIC_FIND_EC, 301
DDSC_LOG_TOPIC_LISTENER, 313
DDSC_LOG_TOPIC_NAME_CMP_EC, 320
DDSC_LOG_TOPIC_NARROW_EC, 302
DDSC_LOG_TOPIC_OBJECT, 307
DDSC_LOG_TOPIC_QOS, 311
DDSC_LOG_TOPIC_RECORD, 303
DDSC_LOG_TOPIC_TOO_LONG_EC, 320
DDSC_LOG_TOPICQOS_OBJECT, 305
DDSC_LOG_TRANSPORT_COMPONENT, 310
DDSC_LOG_TRANSPORT_QOS_POLICY, 312
DDSC_LOG_TYPE_FUNCTION_COPY_SAMPL-E, 320
DDSC_LOG_TYPE_FUNCTION_CREATE_SAM-PLE, 320
DDSC_LOG_TYPE_FUNCTION_DELETE_SAM-PLE, 320
DDSC_LOG_TYPE_FUNCTION_DESERIALIZE_-DATA, 320
DDSC_LOG_TYPE_FUNCTION_GET_KEY_KIN-D, 320
DDSC_LOG_TYPE_FUNCTION_GET_SERIALIZ-ED_SAMPLE_MAX_SIZE, 320

DDSC_LOG_TYPE_FUNCTION_INSTANCE_TO_KEYHASH, 320
DDSC_LOG_TYPE_FUNCTION_NULL_EC, 320
DDSC_LOG_TYPE_FUNCTION_SERIALIZE_DATA, 320
DDSC_LOG_TYPE_GET_SAMPLE_ID_EC, 321
DDSC_LOG_TYPE_KEY_TYPE_EC, 320
DDSC_LOG_TYPE_MANAGED_POOL_FAILURE_EC, 320
DDSC_LOG_TYPE_NAME_CMP_EC, 319
DDSC_LOG_TYPE_OBJECT, 306
DDSC_LOG_TYPE_RECORD, 304
DDSC_LOG_TYPE_SUPPORT_QOS_POLICY, 312
DDSC_LOG_TYPE_TOO_LONG_EC, 320
DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_KEY_EC, 301
DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_NAME_EC, 301
DDSC_LOG_USER_TRAFFIC_QOS_POLICY, 313
DDSC_LOG_USERMULTICAST_SEQUENCE, 309
DDSC_LOG_USERUNICAST_SEQUENCE, 309
DDSC_LOG_UUID_OBJECT, 307
DDSC_LOG_WAITSET_OBJECT, 307
DDSC_LOG_WRITERHISTORY_COMPONENT, 309
DDSC_LOG_WS_ADD_COND_REFERENCE_EC, 302
DDSC_LOG_WS_REMOVE_COND_REFERENCE_EC, 302
DDSC_LOG_WSREF_OBJECT, 306
DDS_CHECKSUM_AUTO
WIRE_PROTOCOL, 267
DDS_CHECKSUM_BUILTIN128
WIRE_PROTOCOL, 267
DDS_CHECKSUM_BUILTIN32
WIRE_PROTOCOL, 267
DDS_CHECKSUM_BUILTIN64
WIRE_PROTOCOL, 267
DDS_CHECKSUM_NONE
WIRE_PROTOCOL, 267
DDS_Char
DDS-Specific Primitive Types, 223
DDS_ChecksumKind_t
WIRE_PROTOCOL, 267
DDS_ChecksumKindMask_t
WIRE_PROTOCOL, 267
DDS_ChecksumProperty, 358
allowed_crc_mask, 359
computed_crc_kind, 359
require_crc, 359
DDS_ChecksumProperty_t, 359
allowed_crc_mask, 359
computed_crc_kind, 359
require_crc, 359
DDS_Condition
Conditions and WaitSets, 275
DDS_Condition_get_trigger_value
Conditions and WaitSets, 276
DDS_ConditionSeq, 359
DDS_DATA_AVAILABLE_STATUS
Status Kinds, 244
DDS_DATA_ON_READERS_STATUS
Status Kinds, 244
DDS_DATAREADER_ENTITY_KIND
dds_c_infrastructure.h, 480
DDS_DATAREADER_QOS_DEFAULT
Subscriber, 79
DDS_DATAREADERPROTOCOL_QOS_POLICY_NAME
DATA_READER_PROTOCOL, 270
DDS_DATAREADERRESOURCELIMITS_QOS_POLICY_NAME
DataReader Resource Limits, 261
DDS_DATAWRITER_ENTITY_KIND
dds_c_infrastructure.h, 480
DDS_DATAWRITER_QOS_DEFAULT
Publisher, 51
DDS_DATAWRITERPROTOCOL_QOS_POLICY_NAME
DATA_WRITER_PROTOCOL, 271
DDS_DATAWRITERRESOURCELIMITS_QOS_POLICY_NAME
DataWriter Resource Limits, 262
DDS_DEADLINE_QOS_POLICY_ID
QoS Policies, 249
DDS_DEADLINE_QOS_POLICY_NAME
DEADLINE, 250
DDS_DISCOVERY_QOS_POLICY_NAME
DISCOVERY, 231
DDS_DOMAINPARTICIPANTRESOURCELIMITS_QOS_POLICY_NAME
DOMAIN_PARTICIPANT_RESOURCE_LIMITS, 273
DDS_DURABILITY_QOS_POLICY_NAME
DURABILITY, 257
DDS_DURATION_INFINITE
Time Support, 233
DDS_DURATION_INFINITE_NSEC
Time Support, 233
DDS_DURATION_INFINITE_SEC
Time Support, 233
DDS_DURATION_ZERO
Time Support, 233
DDS_DURATION_ZERO_NSEC
Time Support, 233
DDS_DURATION_ZERO_SEC
Time Support, 233
DDS_DataReader
DataReader, 84
DDS_DataReader_as_entity
DataReader, 97
DDS_DataReader_get_instance_replaced_status
DataReader, 100

DDS_DataReader_get_liveliness_changed_status
DataReader, 98
DDS_DataReader_get_requested_deadline_missed_-status
DataReader, 98
DDS_DataReader_get_requested_incompatible_qos_-status
DataReader, 99
DDS_DataReader_get_sample_lost_status
DataReader, 99
DDS_DataReader_get_sample_rejected_status
DataReader, 98
DDS_DataReader_get_subscriber
DataReader, 97
DDS_DataReader_get_subscription_matched_status
DataReader, 99
DDS_DataReader_get_topicdescription
DataReader, 97
DDS_DataReaderInstanceReplacedStatus, 360
last_instance_handle, 360
last_replacement_instance, 360
lost_samples, 360
publication_handle, 360
total_count, 360
total_count_change, 360
DDS_DataReaderInstanceReplacedStatus_INITIALIZE-R
DataReader, 84
DDS_DataReaderListener, 361
as_listener, 362
on_before_sample_commit, 362
on_before_sample_deserialize, 362
on_data_available, 362
on_instance_replaced, 363
on_liveliness_changed, 362
on_requested_deadline_missed, 362
on_requested_incompatible_qos, 362
on_sample_lost, 362
on_sample_rejected, 362
on_subscription_matched, 362
DDS_DataReaderListener_BeforeSampleCommit-Callback
DataReader, 86
DDS_DataReaderListener_BeforeSampleDeserialize-Callback
DataReader, 85
DDS_DataReaderListener_DataAvailableCallback
DataReader, 85
DDS_DataReaderListener_INITIALIZER
DataReader, 84
DDS_DataReaderListener_InstanceReplacedCallback
DataReader, 85
DDS_DataReaderListener_LivelinessChangedCallback
DataReader, 85
DDS_DataReaderListener_RequestedDeadlineMissed-Callback
DataReader, 85
DDS_DataReaderListener_RequestedIncompatible-QosCallback
DataReader, 85
DDS_DataReaderListener_SampleLostCallback
DataReader, 85
DDS_DataReaderListener_SampleRejectedCallback
DataReader, 85
DDS_DataReaderListener_SubscriptionMatched-Callback
DataReader, 85
DDS_DataReaderProtocolQosPolicy, 363
rtps_object_id, 364
rtps_reliable_reader, 364
DDS_DataReaderQos, 364
deadline, 365
destination_order, 365
durability, 365
history, 365
liveliness, 365
ownership, 365
protocol, 365
reader_resource_limits, 365
reliability, 365
resource_limits, 365
transport, 365
DDS_DataReaderQos_INITIALIZER
DataReader, 84
DDS_DataReaderQos_copy
DataReader, 96
DDS_DataReaderQos_initialize
DataReader, 96
DDS_DataReaderQos_is_equal
DataReader, 97
DDS_DataReaderResourceLimitsInstanceReplacement-Kind
DataReader Resource Limits, 261
DDS_DataReaderResourceLimitsQosPolicy, 366
instance_replacement, 367
max_outstanding_reads, 367
max_remote_writers, 366
max_remote_writers_per_instance, 366
max_routes_per_writer, 367
max_samples_per_remote_writer, 367
DDS_DataWriter
DataWriter, 56
DDS_DataWriter_as_entity
DataWriter, 66
DDS_DataWriter_assert_liveliness
DataWriter, 66
DDS_DataWriter_get_liveliness_lost_status
DataWriter, 68
DDS_DataWriter_get_offered_deadline_missed_status
DataWriter, 68
DDS_DataWriter_get_offered_incompatible_qos_status
DataWriter, 68
DDS_DataWriter_get_publication_matched_status
DataWriter, 68
DDS_DataWriter_get_publisher

DataWriter, 67
DDS_DataWriter_get_reliable_reader_activity_changed_status
DataWriter, 69
DDS_DataWriter_get_topic
DataWriter, 67
DDS_DataWriterListener, 368
as_listener, 368
on_liveliness_lost, 369
on_offered_deadline_missed, 368
on_offered_incompatible_qos, 369
on_publication_matched, 369
on_reliable_reader_activity_changed, 369
on_reliable_sample_unacknowledged, 369
DDS_DataWriterListener_INITIALIZER
DataWriter, 55
DDS_DataWriterListener_LivelinessLostCallback
DataWriter, 56
DDS_DataWriterListener_OfferedDeadlineMissedCallback
DataWriter, 56
DDS_DataWriterListener_OfferedIncompatibleQosCallback
DataWriter, 56
DDS_DataWriterListener_PublicationMatchedCallback
DataWriter, 57
DDS_DataWriterListener_ReliableReaderActivityChangedCallback
DataWriter, 57
DDS_DataWriterListener_ReliableSampleUnacknowledgedCallback
DataWriter, 57
DDS_DataWriterProtocolQosPolicy, 370
rtps_object_id, 370
rtps_reliable_writer, 370
DDS_DataWriterQos, 370
deadline, 372
destination_order, 372
durability, 372
history, 372
liveliness, 372
ownership, 372
ownership_strength, 372
protocol, 372
reliability, 372
resource_limits, 372
transport, 372
writer_resource_limits, 372
DDS_DataWriterQos_INITIALIZER
DataWriter, 55
DDS_DataWriterQos_copy
DataWriter, 65
DDS_DataWriterQos_initialize
DataWriter, 65
DDS_DataWriterQos_is_equal
DataWriter, 66
DDS_DataWriterResourceLimitsQosPolicy, 372
max_remote_readers, 373
max_routes_per_reader, 373
DDS_DeadlineQosPolicy, 373
period, 374
DDS_DestinationOrderQosPolicy, 374
kind, 375
source_timestamp_tolerance, 375
DDS_DestinationOrderQosPolicyKind
DESTINATION_ORDER, 260
DDS_DiscoveryComponent, 375
name, 375
DDS_DiscoveryComponent_INITIALIZER
dds_c_domain.h, 472
DDS_DiscoveryQosPolicy, 376
accept_unknown_peers, 377
discovery, 377
enabled_transports, 376
initial_peers, 376
DDS_DomainEntity
Entity Support, 283
DDS_DomainId_t
Domain Module, 14
DDS_DomainParticipant
DomainParticipant, 24
DDS_DomainParticipant_add_peer
DomainParticipant, 28
DDS_DomainParticipant_as_entity
DomainParticipant, 25
DDS_DomainParticipant_create_publisher
DomainParticipant, 26
DDS_DomainParticipant_create_subscriber
DomainParticipant, 26
DDS_DomainParticipant_create_topic
DomainParticipant, 27
DDS_DomainParticipant_get_current_time
DomainParticipant, 29
DDS_DomainParticipant_get_domain_id
DomainParticipant, 29
DDS_DomainParticipant_lookup_topicdescription
DomainParticipant, 28
DDS_DomainParticipant_register_type
DomainParticipant, 30
DDS_DomainParticipantFactory
DomainParticipantFactory, 16
DDS_DomainParticipantFactory_create_participant
DomainParticipantFactory, 18
DDS_DomainParticipantFactory_get_instance
DomainParticipantFactory, 18
DDS_DomainParticipantFactory_get_qos
DomainParticipantFactory, 19
DDS_DomainParticipantFactory_get_registry
DomainParticipantFactory, 20
DDS_DomainParticipantFactory_lookup_participant
DomainParticipantFactory, 19
DDS_DomainParticipantFactory_set_qos
DomainParticipantFactory, 20
DDS_DomainParticipantFactoryQos, 377
entity_factory, 378
resource_limits, 378

DDS_DomainParticipantFactoryQos_INITIALIZER
DomainParticipantFactory, 16
DDS_DomainParticipantFactoryQos_copy
DomainParticipantFactory, 17
DDS_DomainParticipantFactoryQos_initialize
DomainParticipantFactory, 17
DDS_DomainParticipantFactoryQos_is_equal
DomainParticipantFactory, 17
DDS_DomainParticipantListener, 378
as_publisherlistener, 379
as_subscriberlistener, 379
as_topiclistener, 379
DDS_DomainParticipantListener_INITIALIZER
DomainParticipant, 23
DDS_DomainParticipantQos, 379
discovery, 379
entity_factory, 379
participant_name, 380
protocol, 380
resource_limits, 380
transports, 380
user_traffic, 380
DDS_DomainParticipantQos_INITIALIZER
DomainParticipant, 23
DDS_DomainParticipantQos_copy
DomainParticipant, 24
DDS_DomainParticipantQos_initialize
DomainParticipant, 24
DDS_DomainParticipantQos_is_equal
DomainParticipant, 25
DDS_DomainParticipantResourceLimitsQosPolicy, 380
local_publisher_allocation, 381
local_reader_allocation, 381
local_subscriber_allocation, 381
local_topic_allocation, 381
local_type_allocation, 381
local_writer_allocation, 381
matching_writer_reader_pair_allocation, 382
max_destination_ports, 383
max_receive_ports, 382
remote_participant_allocation, 382
remote_reader_allocation, 382
remote_writer_allocation, 382
DDS_Double
DDS-Specific Primitive Types, 224
DDS_DurabilityQosPolicy, 384
kind, 384
DDS_DurabilityQosPolicyKind
DURABILITY, 257
DDS_Duration_t, 384
nanosec, 384
sec, 384
DDS_ENTITYFACTORY_QOS_POLICY_ID
QoS Policies, 249
DDS_ENTITYFACTORY_QOS_POLICY_NAME
ENTITY_FACTORY, 263
DDS_ENTITYNAME_QOS_NAME_MAX
ENTITY_NAME, 286
DDS_ENTITYNAME_QOS_POLICY_NAME
ENTITY_NAME, 287
DDS_EXCLUSIVE_OWNERSHIP_QOS
OWNERSHIP, 251
DDS_Entity
Entity Support, 282
DDS_Entity_enable
Entity Support, 283
DDS_Entity_get_instance_handle
Entity Support, 284
DDS_Entity_get_status_changes
Entity Support, 285
DDS_Entity_get_statuscondition
Entity Support, 284
DDS_EntityFactoryQosPolicy, 385
autoenable_created_entities, 385
DDS_EntityKind_t
dds_c_infrastructure.h, 480
DDS_EntityNameQosPolicy, 385
name, 386
DDS_EntityNameQosPolicy_set_name
ENTITY_NAME, 286
DDS_Enum
DDS-Specific Primitive Types, 224
DDS_Float
DDS-Specific Primitive Types, 224
DDS_GROUP_PRESENTATION_QOS
PRESENTATION, 259
DDS_GUID_UNKNOWN
GUID Support, 234
DDS_GUID_t, 386
value, 386
DDS_GuardCondition
Conditions and WaitSets, 275
DDS_GuardCondition_as_condition
Conditions and WaitSets, 277
DDS_GuardCondition_new
Conditions and WaitSets, 277
DDS_GuardCondition_set_trigger_value
Conditions and WaitSets, 277
DDS_HANDLE_NIL
User Data Type Support, 45
DDS_HIGHEST_OFFERED_PRESENTATION_QOS
PRESENTATION, 259
DDS_HISTORY_QOS_POLICY_ID
QoS Policies, 249
DDS_HISTORY_QOS_POLICY_NAME
HISTORY, 256
DDS_HistoryQosPolicy, 387
depth, 388
kind, 388
DDS_HistoryQosPolicyKind
HISTORY, 256
DDS_INCONSISTENT_TOPIC_STATUS
Status Kinds, 242
DDS_INSTANCE_PRESENTATION_QOS
PRESENTATION, 259
DDS_INSTANCE_REPLACED_STATUS

Status Kinds, 245
DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS
WIRE_PROTOCOL, 268
DDS_INVALID_QOS_POLICY_ID
QoS Policies, 249
DDS_InconsistentTopicStatus, 388
DDS_InconsistentTopicStatus_INITIALIZER
Topic, 38
DDS_InstanceHandle_equals
User Data Type Support, 45
DDS_InstanceHandle_is_nil
User Data Type Support, 45
DDS_InstanceHandle_t
User Data Type Support, 43
DDS_InstanceId_t
User-Defined Types, 110
DDS_InstanceStateKind
Instance States, 106
DDS_InstanceStateMask
Instance States, 106
DDS_KEEP_LAST_HISTORY_QOS
HISTORY, 256
DDS_LENGTH_AUTO
RESOURCE_LIMITS, 258
DDS_LENGTH_UNLIMITED
RESOURCE_LIMITS, 258
DDS_LIVELINESS_CHANGED_STATUS
Status Kinds, 244
DDS_LIVELINESS_LOST_STATUS
Status Kinds, 244
DDS_LIVELINESS_QOS_POLICY_ID
QoS Policies, 249
DDS_LIVELINESS_QOS_POLICY_NAME
LIVELINESS, 254
DDS_LOCATOR_INVALID
Discovery, 33
DDS_LOCATOR_KIND_RESERVED
Discovery, 34
DDS_LOCATOR_KIND_SHMEM
Discovery, 34
DDS_LOCATOR_KIND_UDPv4
Discovery, 33
DDS_LOCATOR_KIND_UDPv6
Discovery, 33
DDS_Listener, 388
listener_data, 392
DDS_Listener_INITIALIZER
Entity Support, 281
DDS_LivelinessChangedStatus, 392
alive_count, 392
alive_count_change, 392
last_publication_handle, 392
not_alive_count, 392
not_alive_count_change, 392
DDS_LivelinessChangedStatus_INITIALIZER
DataReader, 83
DDS_LivelinessLostStatus, 392
total_count, 393
total_count_change, 393
DDS_LivelinessLostStatus_INITIALIZER
DataWriter, 54
DDS_LivelinessQosPolicy, 393
kind, 395
lease_duration, 395
DDS_LivelinessQosPolicyKind
LIVELINESS, 253
DDS_Locator_t, 395
address, 396
kind, 396
port, 396
DDS_LocatorSeq, 396
DDS_Long
DDS-Specific Primitive Types, 223
DDS_LongDouble
DDS-Specific Primitive Types, 224
DDS_LongLong
DDS-Specific Primitive Types, 223
DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS
LIVELINESS, 253
DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS
LIVELINESS, 253
DDS_NEW_VIEW_STATE
View States, 105
DDS_NO_INSTANCE_REPLACEMENT_QOS
DataReader Resource Limits, 261
DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE
Instance States, 107
DDS_NOT_ALIVE_INSTANCE_STATE
Instance States, 107
DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE
Instance States, 107
DDS_NOT_NEW_VIEW_STATE
View States, 105
DDS_NOT_READ_SAMPLE_STATE
Sample States, 102
DDS_NOT_REJECTED
DataReader, 86
DDS_OFFERED_DEADLINE_MISSED_STATUS
Status Kinds, 243
DDS_OFFERED_INCOMPATIBLE_QOS_STATUS
Status Kinds, 243
DDS_OWNERSHIP_QOS_POLICY_ID
QoS Policies, 249
DDS_OWNERSHIP_QOS_POLICY_NAME
OWNERSHIP, 251
DDS_OWNERSHIPSTRENGTH_QOS_POLICY_ID
QoS Policies, 249
DDS_OWNERSHIPSTRENGTH_QOS_POLICY_NAME
OWNERSHIP_STRENGTH, 252
DDS_Octet
DDS-Specific Primitive Types, 223
DDS_OfferedDeadlineMissedStatus, 396

last_instance_handle, 397
total_count, 397
total_count_change, 397
DDS_OfferedDeadlineMissedStatus_INITIALIZER
DataWriter, 54
DDS_OfferedIncompatibleQosStatus, 397
last_policy_id, 398
policies, 398
total_count, 398
total_count_change, 398
DDS_OfferedIncompatibleQosStatus_INITIALIZER
DataWriter, 55
DDS_OwnershipQosPolicy, 398
kind, 401
DDS_OwnershipQosPolicyKind
OWNERSHIP, 251
DDS_OwnershipStrengthQosPolicy, 401
value, 402
DDS_PARTICIPANT_ENTITY_KIND
dds_c_infrastructure.h, 480
DDS_PARTICIPANT_FACTORY_QOS_DEFAULT
DomainParticipantFactory, 21
DDS_PARTICIPANT_QOS_DEFAULT
DomainParticipantFactory, 21
DDS_PARTICIPANT_TOPIC_NAME
Participant Built-in Topic, 225
DDS_PRESENTATION_QOS_POLICY_ID
QoS Policies, 249
DDS_PRODUCTVERSION_UNKNOWN
dds_c_infrastructure.h, 479
DDS_PROTOCOLVERSION
Discovery, 34
DDS_PROTOCOLVERSION_1_0
Discovery, 34
DDS_PROTOCOLVERSION_1_1
Discovery, 34
DDS_PROTOCOLVERSION_1_2
Discovery, 34
DDS_PROTOCOLVERSION_2_0
Discovery, 34
DDS_PROTOCOLVERSION_2_1
Discovery, 34
DDS_PUBLICATION_MATCHED_STATUS
Status Kinds, 245
DDS_PUBLICATION_TOPIC_NAME
Publication Built-in Topic, 227
DDS_PUBLISHER_ENTITY_KIND
dds_c_infrastructure.h, 480
DDS_PUBLISHER_QOS_DEFAULT
DomainParticipant, 31
DDS_ParticipantBuiltinTopicData, 402
checksum, 403
dds_builtin_endpoints, 403
default_multicast_locators, 403
default_unicast_locators, 403
key, 403
liveliness_lease_duration, 403
metatraffic_multicast_locators, 403
metatraffic_unicast_locators, 403
participant_name, 403
product_version, 403
rtps_protocol_version, 403
rtps_vendor_id, 403
DDS_ParticipantBuiltinTopicData_copy
Participant Built-in Topic, 226
DDS_ParticipantBuiltinTopicData_initialize
Participant Built-in Topic, 225
DDS_ParticipantBuiltinTopicData_is_equal
Participant Built-in Topic, 226
DDS_PresentationQosPolicy, 403
access_scope, 406
coherent_access, 406
ordered_access, 406
DDS_PresentationQosPolicyAccessScopeKind
PRESENTATION, 259
DDS_ProductVersion, 406
major, 407
minor, 407
release, 407
revision, 407
DDS_ProductVersion_t, 407
major, 407
minor, 407
release, 407
revision, 407
DDS_ProtocolVersion, 408
major, 408
minor, 408
DDS_ProtocolVersion_t
Discovery, 34
DDS_PublicationBuiltinTopicData, 408
deadline, 410
destination_order, 410
durability, 410
key, 409
liveliness, 410
ownership, 410
ownership_strength, 410
participant_key, 409
reliability, 410
topic_name, 409
type_name, 409
unicast_locator, 410
DDS_PublicationBuiltinTopicData_copy
Publication Built-in Topic, 228
DDS_PublicationBuiltinTopicData_initialize
Publication Built-in Topic, 227
DDS_PublicationBuiltinTopicData_is_equal
Publication Built-in Topic, 228
DDS_PublicationMatchedStatus, 410
current_count, 411
current_count_change, 411
last_subscription_handle, 411
total_count, 411
total_count_change, 411
DDS_PublicationMatchedStatus_INITIALIZER

DataWriter, 55
DDS_Publisher
Publisher, 48
DDS_Publisher_as_entity
Publisher, 50
DDS_Publisher_create_datawriter
Publisher, 50
DDS_Publisher_get_participant
Publisher, 51
DDS_Publisher_lookup_datawriter
Publisher, 50
DDS_PublisherListener, 411
as_datawriterlistener, 412
DDS_PublisherListener_INITIALIZER
Publisher, 48
DDS_PublisherQos, 412
entity_factory, 412
DDS_PublisherQos_INITIALIZER
Publisher, 48
DDS_PublisherQos_copy
Publisher, 49
DDS_PublisherQos_initialize
Publisher, 48
DDS_PublisherQos_is_equal
Publisher, 49
DDS_QosPolicyId_t
QoS Policies, 249
DDS_READ_SAMPLE_STATE
Sample States, 102
DDS_REJECTED_BY_INSTANCES_LIMIT
DataReader, 86
DDS_REJECTED_BY_REMOTE_WRITERS_LIMIT
DataReader, 86
DDS_REJECTED_BY_SAMPLES_LIMIT
DataReader, 86
DDS_REJECTED_BY_SAMPLES_PER_INSTANCE_-
LIMIT
DataReader, 86
DDS_REJECTED_BY_SAMPLES_PER_REMOTE_W-
RITER_LIMIT
DataReader, 86
DDS_RELIABILITY_QOS_POLICY_ID
QoS Policies, 249
DDS_RELIABILITY_QOS_POLICY_NAME
RELIABILITY, 255
DDS_RELIABLE_READER_ACTIVITY_CHANGED_S-
TATUS
Status Kinds, 245
DDS_RELIABLE_RELIABILITY_QOS
RELIABILITY, 255
DDS_REPLACE_OLDEST_INSTANCE_REPLACEME-
NT_QOS
DataReader Resource Limits, 261
DDS_REQUESTED_DEADLINE_MISSED_STATUS
Status Kinds, 243
DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS
Status Kinds, 243
DDS_RESOURCELIMITS_QOS_POLICY_NAME
RESOURCE_LIMITS, 258
DDS_RETCODE_ALREADY_DELETED
Return Codes, 237
DDS_RETCODE_BAD_PARAMETER
Return Codes, 236
DDS_RETCODE_ERROR
Return Codes, 236
DDS_RETCODE_ILLEGAL_OPERATION
Return Codes, 237
DDS_RETCODE_IMMUTABLE_POLICY
Return Codes, 237
DDS_RETCODE_INCONSISTENT_POLICY
Return Codes, 237
DDS_RETCODE_NO_DATA
Return Codes, 237
DDS_RETCODE_NOT_ENABLED
Return Codes, 237
DDS_RETCODE_OK
Return Codes, 236
DDS_RETCODE_OUT_OF_RESOURCES
Return Codes, 237
DDS_RETCODE_PRECONDITION_NOT_MET
Return Codes, 237
DDS_RETCODE_TIMEOUT
Return Codes, 237
DDS_RETCODE_UNSUPPORTED
Return Codes, 236
DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WEL-
L_KNOWN_PORTS
WIRE_PROTOCOL, 268
DDS_RTPS_AUTO_ID
dds_c_infrastructure.h, 480
DDS_ReliabilityQosPolicy, 412
kind, 413
max_blocking_time, 413
DDS_ReliabilityQosPolicyKind
RELIABILITY, 255
DDS_ReliableReaderActivityChangedStatus, 413
active_count, 414
active_count_change, 414
inactive_count, 414
inactive_count_change, 414
last_instance_handle, 414
DDS_ReliableReaderActivityChangedStatus_INITIALI-
ZER
DataWriter, 55
DDS_ReliableSampleUnacknowledgedStatus, 414
instance_handle, 415
sequence_number, 415
unacknowledged_count, 415
DDS_RequestedDeadlineMissedStatus, 415
last_instance_handle, 415
total_count, 415
total_count_change, 415
DDS_RequestedDeadlineMissedStatus_INITIALIZER
DataReader, 83
DDS_RequestedIncompatibleQosStatus, 416
last_policy_id, 416

policies, 416
total_count, 416
total_count_change, 416
DDS_RequestedIncompatibleQosStatus_INITIALIZER
DataReader, 83
DDS_ResourceLimitsQosPolicy, 416
max_instances, 418
max_samples, 418
max_samples_per_instance, 419
DDS_ReturnCode_t
Return Codes, 236
DDS_RtpsReliableReaderProtocol_t, 420
nack_period, 420
DDS_RtpsReliableWriterProtocol_t, 420
heartbeat_period, 421
heartbeats_per_max_samples, 421
max_heartbeat_retries, 421
max_send_window, 421
DDS_RtpsWellKnownPorts_t, 422
builtin_multicast_port_offset, 424
builtin_unicast_port_offset, 424
domain_id_gain, 423
participant_id_gain, 424
port_base, 423
user_multicast_port_offset, 424
user_unicast_port_offset, 424
DDS_SAMPLE_LOST_BY_DATAWRITER
DataReader, 87
DDS_SAMPLE_LOST_BY_HISTORY_DEPTH_LIMIT
DataReader, 87
DDS_SAMPLE_LOST_BY_MANAGEDPOOL_OVERWRITE
DataReader, 87
DDS_SAMPLE_LOST_BY_MAX_SAMPLES_LIMIT
DataReader, 87
DDS_SAMPLE_LOST_BY_META_SAMPLE_LIMIT
DataReader, 87
DDS_SAMPLE_LOST_BY_NOT_READ_ON_CACHE_DELETION
DataReader, 87
DDS_SAMPLE_LOST_NOT_LOST
DataReader, 86
DDS_SAMPLE_LOST_STATUS
Status Kinds, 243
DDS_SAMPLE_REJECTED_STATUS
Status Kinds, 244
DDS_SEQUENCE_INITIALIZER
Sequence Support, 323
DDS_SHARED_OWNERSHIP_QOS
OWNERSHIP, 251
DDS_STATUS_MASK_ALL
Status Kinds, 242
DDS_STATUS_MASK_NONE
Status Kinds, 242
DDS_SUBSCRIBER_ENTITY_KIND
dds_c_infrastructure.h, 480
DDS_SUBSCRIBER_QOS_DEFAULT
DomainParticipant, 31
DDS_SUBSCRIPTION_MATCHED_STATUS
Status Kinds, 245
DDS_SUBSCRIPTION_TOPIC_NAME
Subscription Built-in Topic, 229
DDS_SYSTEMRESOURCELIMITS_QOS_POLICY_NAME
SYSTEM_RESOURCE_LIMITS, 265
DDS_SampleInfo, 425
instance_handle, 427
instance_state, 426
publication_handle, 427
publication_sequence_number, 427
reception_timestamp, 427
sample_state, 426
source_timestamp, 426
valid_data, 427
view_state, 426
DDS_SampleInfoSeq, 427
DDS_SampleLostStatus, 428
reason, 428
sample_info, 428
total_count, 428
total_count_change, 428
DDS_SampleLostStatus_INITIALIZER
DataReader, 84
DDS_SampleLostStatusKind
DataReader, 86
DDS_SampleRejectedStatus, 428
last_instance_handle, 429
last_reason, 429
total_count, 429
total_count_change, 429
DDS_SampleRejectedStatus_INITIALIZER
DataReader, 83
DDS_SampleRejectedStatusKind
DataReader, 86
DDS_SampleStateKind
Sample States, 102
DDS_SampleStateMask
Sample States, 102
DDS_SequenceNumber_t, 429
high, 430
low, 430
DDS_Short
DDS-Specific Primitive Types, 223
DDS_StatusCondition
Conditions and WaitSets, 275
DDS_StatusCondition_as_condition
Conditions and WaitSets, 277
DDS_StatusCondition_get_enabled_statuses
Conditions and WaitSets, 278
DDS_StatusCondition_get_entity
Conditions and WaitSets, 278
DDS_StatusCondition_set_enabled_statuses
Conditions and WaitSets, 278
DDS_StatusKind
Status Kinds, 242
DDS_StatusMask

Status Kinds, 242
DDS_String
DDS-Specific Primitive Types, 224
DDS_String_alloc
String Support, 330
DDS_String_cmp
String Support, 330
DDS_String_dup
String Support, 330
DDS_String_length
String Support, 331
DDS_String_ncmp
String Support, 331
DDS_Subscriber
Subscriber, 74
DDS_Subscriber_as_entity
Subscriber, 76
DDS_Subscriber_create_datareader
Subscriber, 77
DDS_Subscriber_get_default_datareader_qos
Subscriber, 76
DDS_Subscriber_get_participant
Subscriber, 79
DDS_Subscriber_lookup_datareader
Subscriber, 78
DDS_Subscriber_set_default_datareader_qos
Subscriber, 77
DDS_SubscriberListener, 430
as_datareaderlistener, 430
on_data_on_readers, 430
DDS_SubscriberListener_DataOnReadersCallback
Subscriber, 75
DDS_SubscriberListener_INITIALIZER
Subscriber, 74
DDS_SubscriberQos, 431
entity_factory, 431
DDS_SubscriberQos_INITIALIZER
Subscriber, 74
DDS_SubscriberQos_copy
Subscriber, 75
DDS_SubscriberQos_initialize
Subscriber, 75
DDS_SubscriberQos_is_equal
Subscriber, 76
DDS_SubscriptionBuiltinTopicData, 431
deadline, 432
destination_order, 433
durability, 432
key, 432
liveliness, 432
multicast_locator, 433
ownership, 432
participant_key, 432
reliability, 432
topic_name, 432
type_name, 432
unicast_locator, 433
DDS_SubscriptionBuiltinTopicData_copy
Subscription Built-in Topic, 230
DDS_SubscriptionBuiltinTopicData_initialize
Subscription Built-in Topic, 229
DDS_SubscriptionBuiltinTopicData_is_equal
Subscription Built-in Topic, 230
DDS_SubscriptionMatchedStatus, 433
current_count, 434
current_count_change, 434
last_publication_handle, 434
total_count, 433
total_count_change, 433
DDS_SubscriptionMatchedStatus_INITIALIZER
DataReader, 84
DDS_SystemResourceLimitsQosPolicy, 434
max_components, 435
max_participants, 434
DDS_TIME_INVALID
Time Support, 233
DDS_TIME_INVALID_NSEC
Time Support, 232
DDS_TIME_INVALID_SEC
Time Support, 232
DDS_TIME_ZERO
Time Support, 232
DDS_TOPIC_ENTITY_KIND
dds_c_infrastructure.h, 480
DDS_TOPIC_PRESENTATION_QOS
PRESENTATION, 259
DDS_TOPIC_QOS_DEFAULT
DomainParticipant, 30
DDS_TRANSIENT_LOCAL_DURABILITY_QOS
DURABILITY, 257
DDS_TRANSPORT_QOS_POLICY_NAME
TRANSPORT, 272
DDS_TYPESUPPORT_C
User Data Type Support, 43
DDS_TheParticipantFactory
DomainParticipantFactory, 16
DDS_Time_t, 435
nanosec, 435
sec, 435
DDS_Topic
Topic, 38
DDS_Topic_as_entity
Topic, 40
DDS_Topic_as_topicdescription
Topic, 40
DDS_Topic_get_inconsistent_topic_status
Topic, 40
DDS_Topic_narrow
Topic, 40
DDS_TopicDescription
Topic, 38
DDS_TopicDescription_get_name
Topic, 39
DDS_TopicDescription_get_participant
Topic, 40
DDS_TopicDescription_get_type_name

Topic, 39
DDS_TopicListener, 435
as_listener, 436
on_inconsistent_topic, 436
DDS_TopicListener_INITIALIZER
Topic, 38
DDS_TopicListener_InconsistentTopicCallback
Topic, 39
DDS_TopicQos, 436
DDS_TransportQosPolicy, 436
enabled_transports, 437
DDS_UNKNOWN_ENTITY_KIND
dds_c_infrastructure.h, 480
DDS_UnsignedLong
DDS-Specific Primitive Types, 223
DDS_UnsignedLongLong
DDS-Specific Primitive Types, 223
DDS_UnsignedShort
DDS-Specific Primitive Types, 223
DDS_UserTrafficQosPolicy, 437
enabled_transports, 437
DDS_VENDOR_ID_LENGTH_MAX
Discovery, 34
DDS_VOLATILE_DURABILITY_QOS
DURABILITY, 257
DDS_VendorId, 438
vendorId, 438
DDS_VendorId_t, 438
vendorId, 438
DDS_ViewStateKind
View States, 104
DDS_ViewStateMask
View States, 104
DDS_WIREPROTOCOL_QOS_POLICY_NAME
WIRE_PROTOCOL, 268
DDS_WaitSet
Conditions and WaitSets, 275
DDS_WaitSet_attach_condition
Conditions and WaitSets, 280
DDS_WaitSet_get_conditions
Conditions and WaitSets, 280
DDS_WaitSet_new
Conditions and WaitSets, 279
DDS_WaitSet_wait
Conditions and WaitSets, 279
DDS_Wchar
DDS-Specific Primitive Types, 223
DDS_WireProtocolQosPolicy, 438
allowed_crc_mask, 441
check_crc, 441
compute_crc, 441
computed_crc_kind, 441
participant_id, 439
require_crc, 441
rtps_app_id, 440
rtps_host_id, 440
rtps_instance_id, 440
rtps_well_known_ports, 440
DDS_Wstring
DDS-Specific Primitive Types, 224
DDS_Wstring_cmp
String Support, 331
DDSC_LOG_ADDRESSRESOLVER_OBJECT
DDS_C, 306
DDSC_LOG_BIND_RECORD
DDS_C, 304
DDSC_LOG_BINDRESOLVER_OBJECT
DDS_C, 306
DDSC_LOG_CDR_BUFFER_SET_EC
DDS_C, 315
DDSC_LOG_CDR_DESERIALIZE_DATA_EC
DDS_C, 316
DDSC_LOG_CDR_DESERIALIZE_HEADER_EC
DDS_C, 316
DDSC_LOG_CDR_DESERIALIZE_KEY_EC
DDS_C, 316
DDSC_LOG_CDR_DESERIALIZE_KEYHASH_EC
DDS_C, 316
DDSC_LOG_CDR_DESERIALIZE_PID_EC
DDS_C, 316
DDSC_LOG_CDR_DESERIALIZE_PID_LENGTH_EC
DDS_C, 316
DDSC_LOG_CDR_FINALIZE_SAMPLE_EC
DDS_C, 316
DDSC_LOG_CDR_INCREMENT_POS_EC
DDS_C, 316
DDSC_LOG_CDR_INITIALIZE_SAMPLE_EC
DDS_C, 316
DDSC_LOG_CDR_POOL_ALLOC_EC
DDS_C, 315
DDSC_LOG_CDR_POOL_DELETE_EC
DDS_C, 315
DDSC_LOG_CDR_SERIALIZE_DATA_EC
DDS_C, 316
DDSC_LOG_CDR_SERIALIZE_KEYHASH_EC
DDS_C, 316
DDSC_LOG_CDR_SERIALIZE_PID_EC
DDS_C, 315
DDSC_LOG_CDR_SERIALIZE_PID_LENGTH_EC
DDS_C, 315
DDSC_LOG_CDR_SERIALIZE_STATUS_INFO_EC
DDS_C, 316
DDSC_LOG_CDR_SET_POS_EC
DDS_C, 316
DDSC_LOG_CHECKSUM_INCOMPATIBLE_EC
DDS_C, 321
DDSC_LOG_CHECKSUM_INCONSISTENT_ALLOW_-
EC
DDS_C, 321
DDSC_LOG_CHECKSUM_INCONSISTENT_CLASS_-
EC
DDS_C, 321
DDSC_LOG_CHECKSUM_INCONSISTENT_COMPU-
TE_EC
DDS_C, 321
DDSC_LOG_CHECKSUM_REQUIRED_EC

DDS_C, 321
DDSC_LOG_COMPONENT_CREATE_EC
DDS_C, 310
DDSC_LOG_COMPONENT_DELETE_EC
DDS_C, 310
DDSC_LOG_COMPONENT_LOOKUP_EC
DDS_C, 310
DDSC_LOG_CONDITION_OBJECT
DDS_C, 307
DDSC_LOG_CONDREF_OBJECT
DDS_C, 306
DDSC_LOG_CREATE_INDEX_EC
DDS_C, 303
DDSC_LOG_DATABASE_CREATE_EC
DDS_C, 303
DDSC_LOG_DATABASE_DELETE_EC
DDS_C, 303
DDSC_LOG_DATAREADER_CDR
DDS_C, 315
DDSC_LOG_DATAREADER_ENTITY
DDS_C, 314
DDSC_LOG_DATAREADER_LISTENER
DDS_C, 313
DDSC_LOG_DATAREADER_NETIO_KIND
DDS_C, 317
DDSC_LOG_DATAREADER_QOS
DDS_C, 311
DDSC_LOG_DATAREADER_RECORD
DDS_C, 303
DDSC_LOG_DATAREADER_RESOURCE_QOS_POLICY
DDS_C, 312
DDSC_LOG_DATAREADERIO_COMPONENT
DDS_C, 310
DDSC_LOG_DATAREADERIO_OBJECT
DDS_C, 306
DDSC_LOG_DATAREADERQOS_OBJECT
DDS_C, 305
DDSC_LOG_DATAWRITER_CDR
DDS_C, 315
DDSC_LOG_DATAWRITER_ENTITY
DDS_C, 314
DDSC_LOG_DATAWRITER_LISTENER
DDS_C, 314
DDSC_LOG_DATAWRITER_NETIO_KIND
DDS_C, 317
DDSC_LOG_DATAWRITER_QOS
DDS_C, 311
DDSC_LOG_DATAWRITER_RECORD
DDS_C, 303
DDSC_LOG_DATAWRITER_RESOURCE_QOS_POLICY
DDS_C, 312
DDSC_LOG_DATAWRITERIO_COMPONENT
DDS_C, 310
DDSC_LOG_DATAWRITERIO_OBJECT
DDS_C, 306
DDSC_LOG_DATAWRITERQOS_OBJECT
DDS_C, 305
DDSC_LOG_DB_CURSOR_INVALIDATED_EC
DDS_C, 303
DDSC_LOG_DEADLINE_QOS_POLICY
DDS_C, 311
DDSC_LOG_DEFAULTDATAREADER_QOS
DDS_C, 311
DDSC_LOG_DEFAULTDATAWRITER_QOS
DDS_C, 311
DDSC_LOG_DEFAULTPARTICIPANT_QOS
DDS_C, 311
DDSC_LOG_DEFAULTPUBLISHER_QOS
DDS_C, 311
DDSC_LOG_DEFAULTSUBSCRIBER_QOS
DDS_C, 311
DDSC_LOG_DEFAULTTOPIC_QOS
DDS_C, 311
DDSC_LOG_DELETE_INDEX_EC
DDS_C, 303
DDSC_LOG_DESERIALIZE_BAD_PID_LENGTH_EC
DDS_C, 316
DDSC_LOG_DESTINATION_ORDER_POLICY
DDS_C, 312
DDSC_LOG_DESTINATION_SEQUENCE
DDS_C, 308
DDSC_LOG_DISC_ADD_PEER_EC
DDS_C, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_DELETED_EC
DDS_C, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAREADER_ENABLED_EC
DDS_C, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_DELETED_EC
DDS_C, 321
DDSC_LOG_DISC_AFTER_LOCAL_DATAWRITER_ENABLED_EC
DDS_C, 321
DDSC_LOG_DISC_AFTER_LOCAL_PARTICIPANT_CREATED_EC
DDS_C, 321
DDSC_LOG_DISC_BEFORE_LOCAL_PARTICIPANT_CREATED_EC
DDS_C, 321
DDSC_LOG_DISC_LOCAL_PARTICIPANT_ENABLED_EC
DDS_C, 321
DDSC_LOG_DISC_REMOTE_PARTICIPANT_EXPIRED_EC
DDS_C, 321
DDSC_LOG_DISCOVERY_COMPONENT
DDS_C, 309
DDSC_LOG_DISCOVERY_TRAFFIC_QOS_POLICY
DDS_C, 313
DDSC_LOG_DR_COMMIT_ENTRY_EC
DDS_C, 319
DDSC_LOG_DR_COMMIT_SAMPLE_EC

DDS_C, 319
DDSC_LOG_DR_COPY_DATA_SAMPLE_EC
DDS_C, 319
DDSC_LOG_DR_CREATE_TYPED_READER_EC
DDS_C, 319
DDSC_LOG_DR_DESERIALIZE_KEYHASH_EC
DDS_C, 319
DDSC_LOG_DR_DISPOSE_KEY_EC
DDS_C, 319
DDSC_LOG_DR_FILTER_ERROR_EC
DDS_C, 319
DDSC_LOG_DR_GET_ENTRY_FAILED_EC
DDS_C, 319
DDSC_LOG_DR_INSTANCE_TO_KEYHASH_EC
DDS_C, 319
DDSC_LOG_DR_READ_TAKE_FAILURE_EC
DDS_C, 319
DDSC_LOG_DR_UNREGISTER_KEY_EC
DDS_C, 319
DDSC_LOG_DURABILITY_QOS_POLICY
DDS_C, 312
DDSC_LOG_DW_ACKNACK_FAILED_EC
DDS_C, 318
DDSC_LOG_DW_COMMIT_EC
DDS_C, 318
DDSC_LOG_DW_CREATE_TYPED_WRITER_EC
DDS_C, 319
DDSC_LOG_DW_HISTORY_REGISTER_KEY_EC
DDS_C, 319
DDSC_LOG_DW_ILLEGAL_KEY_KIND_EC
DDS_C, 319
DDSC_LOG_DW_KEYHASH_CREATE_EC
DDS_C, 318
DDSC_LOG_ENABLED_DISCVOERY_TRANSPORT-
_SEQUENCE
DDS_C, 308
DDSC_LOG_ENABLED_TRANSPORT_SEQUENCE
DDS_C, 308
DDSC_LOG_ENABLED_USER_TRANSPORT_SEQU-
ENCE
DDS_C, 308
DDSC_LOG_ENDPOINT_NOT_CHILD_OF_PARTICI-
PANT_EC
DDS_C, 301
DDSC_LOG_ENTITY_DIFFERENT_FACTORY_EC
DDS_C, 315
DDSC_LOG_ENTITY_ENABLE_EC
DDS_C, 315
DDSC_LOG_ENTITY_FINALIZE_EC
DDS_C, 315
DDSC_LOG_ENTITY_INITIALIZE_EC
DDS_C, 315
DDSC_LOG_ENTITY_NOT_EMPTY_EC
DDS_C, 315
DDSC_LOG_ENTITY_NOT_ENABLED_EC
DDS_C, 315
DDSC_LOG_ENTITY_OBJECT
DDS_C, 306
DDSC_LOG_FAILED_UPDATE_STATUS_CONDITIO-
N_EC
DDS_C, 302
DDSC_LOG_FIND_PUBLICATION_PARENT_EC
DDS_C, 301
DDSC_LOG_FIND_SUBSCRIPTION_PARENT_EC
DDS_C, 301
DDSC_LOG_GET_NEXT_OBJECT_ID_EC
DDS_C, 300
DDSC_LOG_HEARTBEATS_QOS_POLICY
DDS_C, 312
DDSC_LOG_HISTORY_QOS_POLICY
DDS_C, 312
DDSC_LOG_HISTORY_RESOURCE
DDS_C, 310
DDSC_LOG_ILLEGAL_OBJECTID_EC
DDS_C, 302
DDSC_LOG_INITIAL_PEER_SEQUENCE
DDS_C, 308
DDSC_LOG_INTRA_NETIO_KIND
DDS_C, 317
DDSC_LOG_INVALID_DOMAINID_EC
DDS_C, 302
DDSC_LOG_INVALID_DURATION_EC
DDS_C, 300
DDSC_LOG_INVALID_ENDPOINT_GUID_EC
DDS_C, 301
DDSC_LOG_INVALID_PARTICIPANT_GUID_PREFIX-
_EC
DDS_C, 300
DDSC_LOG_INVALID_PARTICIPANT_NAME_EC
DDS_C, 300
DDSC_LOG_IO_SNPRINTF_FAILED_EC
DDS_C, 301
DDSC_LOG_LISTENER_GET_EC
DDS_C, 314
DDSC_LOG_LISTENER_INCONSISTENT_EC
DDS_C, 314
DDSC_LOG_LISTENER_SET_EC
DDS_C, 314
DDSC_LOG_LISTENER_SET_ILLEGAL_NULL_EC
DDS_C, 314
DDSC_LOG_LIVELINESS_QOS_POLICY
DDS_C, 311
DDSC_LOG_LOG_OBJECT
DDS_C, 306
DDSC_LOG_LOOKUP_TYPE_PLUGIN_EC
DDS_C, 320
DDSC_LOG_MAX_PARTICIPANT_ID_REACHED_EC
DDS_C, 302
DDSC_LOG_MD5STREAM_OBJECT
DDS_C, 306
DDSC_LOG_METAMULTICAST_SEQUENCE
DDS_C, 309
DDSC_LOG_METAUNICAST_SEQUENCE
DDS_C, 308
DDSC_LOG_MUTEX_OBJECT
DDS_C, 306

DDSC_LOG_NETIO_ADD_ANON_TOPIC_ROUTE_EC
DDS_C, 316
DDSC_LOG_NETIO_ADD_ROUTE_EC
DDS_C, 317
DDSC_LOG_NETIO_ADD_TOPIC_ROUTE_EC
DDS_C, 317
DDSC_LOG_NETIO_BIND_EC
DDS_C, 317
DDSC_LOG_NETIO_BIND_EXTERNAL_EC
DDS_C, 317
DDSC_LOG_NETIO_COMPONENT
DDS_C, 310
DDSC_LOG_NETIO_DELETE_ROUTE_EC
DDS_C, 317
DDSC_LOG_NETIO_DELETE_TOPIC_ROUTE_EC
DDS_C, 317
DDSC_LOG_NETIO_DISCOVERY_ENABLED_TRANSPORT_EC
DDS_C, 318
DDSC_LOG_NETIO_FORCED_REMOVE_EC
DDS_C, 318
DDSC_LOG_NETIO_FORWARD_TOPIC_EC
DDS_C, 317
DDSC_LOG_NETIO_GET_EXTERNAL_INTF_EC
DDS_C, 317
DDSC_LOG_NETIO_GET_ROUTE_TABLE_FAILED_EC
DDS_C, 318
DDSC_LOG_NETIO_NETIO_KIND
DDS_C, 317
DDSC_LOG_NETIO_NO_ROUTE_EC
DDS_C, 318
DDSC_LOG_NETIO_PEER_LOOKUP_EC
DDS_C, 318
DDSC_LOG_NETIO_ROUTE_LOOKUP_FAILED_EC
DDS_C, 318
DDSC_LOG_NETIO_SEND_FAILED_EC
DDS_C, 318
DDSC_LOG_NETIO_SET_STATE_EC
DDS_C, 318
DDSC_LOG_NETIO_UNBIND_EC
DDS_C, 317
DDSC_LOG_NETIO_UNBIND_EXTERNAL_EC
DDS_C, 317
DDSC_LOG_NETMASK_SEQUENCE
DDS_C, 308
DDSC_LOG_OBJECT_ALLOCATE_EC
DDS_C, 307
DDSC_LOG_OBJECT_COPY_EC
DDS_C, 307
DDSC_LOG_OBJECT_CREATE_EC
DDS_C, 308
DDSC_LOG_OBJECT_DELETE_EC
DDS_C, 307
DDSC_LOG_OBJECT_EMPTY_EC
DDS_C, 308
DDSC_LOG_OBJECT_FINALIZE_EC
DDS_C, 307
DDSC_LOG_OBJECT_GET_PROPERTY_EC
DDS_C, 308
DDSC_LOG_OBJECT_INITIALIZE_EC
DDS_C, 307
DDSC_LOG_OBJECT_INUSECOUNT_EC
DDS_C, 308
DDSC_LOG_OBJECT_REFCOUNT_EC
DDS_C, 308
DDSC_LOG_OBJECT_SET_PROPERTY_EC
DDS_C, 308
DDSC_LOG_OWNERSHIP_QOS_POLICY
DDS_C, 312
DDSC_LOG_OWNERSHIP_STRENGTH_QOS_POLICY
DDS_C, 312
DDSC_LOG_PACKET_INIT_EC
DDS_C, 318
DDSC_LOG_PACKET_SET_HEAD_EC
DDS_C, 318
DDSC_LOG_PACKET_SET_TAIL_EC
DDS_C, 318
DDSC_LOG_PARTICIPANT_DOES_NOT_EXIST_EC
DDS_C, 301
DDSC_LOG_PARTICIPANT_ENTITY
DDS_C, 314
DDSC_LOG_PARTICIPANT_ID_QOS_POLICY
DDS_C, 312
DDSC_LOG_PARTICIPANT_LISTENER
DDS_C, 314
DDSC_LOG_PARTICIPANT_LOOKUP_EC
DDS_C, 301
DDSC_LOG_PARTICIPANT_POOL_OBJECT
DDS_C, 307
DDSC_LOG_PARTICIPANT_QOS
DDS_C, 311
DDSC_LOG_PARTICIPANT_RECORD
DDS_C, 303
DDSC_LOG_PARTICIPANT_RESOURCES
DDS_C, 310
DDSC_LOG_PARTICIPANTDATA_OBJECT
DDS_C, 305
DDSC_LOG_PARTICIPANTFACTORY_LISTENER
DDS_C, 314
DDSC_LOG_PARTICIPANTFACTORY_QOS
DDS_C, 311
DDSC_LOG_PARTICIPANTFACTORYQOS_OBJECT
DDS_C, 306
DDSC_LOG_PARTICIPANTQOS_OBJECT
DDS_C, 306
DDSC_LOG_PROTOCOL_QOS_POLICY
DDS_C, 312
DDSC_LOG_PUBLICATION_ENTITY
DDS_C, 315
DDSC_LOG_PUBLICATION_RECORD
DDS_C, 303
DDSC_LOG_PUBLICATIONDATA_OBJECT
DDS_C, 305

DDSC_LOG_PUBLISHER_ENTITY
DDS_C, 314
DDSC_LOG_PUBLISHER_LISTENER
DDS_C, 314
DDSC_LOG_PUBLISHER_QOS
DDS_C, 311
DDSC_LOG_PUBLISHER_RECORD
DDS_C, 304
DDSC_LOG_PUBLISHERQOS_OBJECT
DDS_C, 305
DDSC_LOG_QOS_CHANGED_EC
DDS_C, 313
DDSC_LOG_QOS_COPY_EC
DDS_C, 313
DDSC_LOG_QOS_FINALIZE_EC
DDS_C, 313
DDSC_LOG_QOS_GET_EC
DDS_C, 313
DDSC_LOG_QOS_IMMUTABLE_EC
DDS_C, 313
DDSC_LOG_QOS_INCONSISTENT_EC
DDS_C, 313
DDSC_LOG_QOS_INCONSISTENT_POLICIES_EC
DDS_C, 313
DDSC_LOG_QOS_INCONSISTENT_POLICY_EC
DDS_C, 313
DDSC_LOG_QOS_INITIALIZE_EC
DDS_C, 313
DDSC_LOG_QOS_SET_EC
DDS_C, 313
DDSC_LOG_QOS_SET_ON_ENABLED_EC
DDS_C, 313
DDSC_LOG_READERHISTORY_COMPONENT
DDS_C, 309
DDSC_LOG_READTAKE_SEQUENCE
DDS_C, 309
DDSC_LOG_RECORD_CREATE_EC
DDS_C, 304
DDSC_LOG_RECORD_DELETE_EC
DDS_C, 304
DDSC_LOG_RECORD_ERROR_EC
DDS_C, 304
DDSC_LOG_RECORD_EXISTS_EC
DDS_C, 305
DDSC_LOG_RECORD_FINALIZE_EC
DDS_C, 305
DDSC_LOG_RECORD_INITIALIZE_EC
DDS_C, 305
DDSC_LOG_RECORD_INSERT_EC
DDS_C, 304
DDSC_LOG_RECORD_LOOKUP_EC
DDS_C, 305
DDSC_LOG_RECORD_NOT_EXISTS_EC
DDS_C, 305
DDSC_LOG_RECORD_REMOVE_EC
DDS_C, 305
DDSC_LOG_RECORD_SELECT_EC
DDS_C, 305
DDSC_LOG_REFRESH_REM_PARTICIPANT_EC
DDS_C, 301
DDSC_LOG_REFRESH_REM_PARTICIPANT_TIME-
OUT_EC
DDS_C, 301
DDSC_LOG_RELEASE_META_MC_EC
DDS_C, 302
DDSC_LOG_RELEASE_META_UC_EC
DDS_C, 302
DDSC_LOG_RELEASE_USER_MC_EC
DDS_C, 302
DDSC_LOG_RELEASE_USER_UC_EC
DDS_C, 302
DDSC_LOG_RELIABILITY_QOS_POLICY
DDS_C, 312
DDSC_LOG_REMOTE_PARTICIPANT_KEY_NOT_E-
QUAL_EC
DDS_C, 301
DDSC_LOG_REMOVE_PUBLICATION_EC
DDS_C, 301
DDSC_LOG_REMOVE_SUBSCRIPTION_EC
DDS_C, 301
DDSC_LOG_RESERVE_LOCATORS_EC
DDS_C, 302
DDSC_LOG_RESERVED_SEQUENCE
DDS_C, 308
DDSC_LOG_RESOLVE_LOCATORS_PRIORITY_EC
DDS_C, 302
DDSC_LOG_RESOURCE_EXCEEDED_EC
DDS_C, 310
DDSC_LOG_RESOURCE_LIMIT_QOS_POLICY
DDS_C, 312
DDSC_LOG_ROUTE_RECORD
DDS_C, 304
DDSC_LOG_ROUTE_SEQUENCE
DDS_C, 308
DDSC_LOG_ROUTERESOLVER_OBJECT
DDS_C, 306
DDSC_LOG_RT_OBJECT
DDS_C, 307
DDSC_LOG_RTPS_COMPONENT
DDS_C, 309
DDSC_LOG_RTPS_NETIO_KIND
DDS_C, 317
DDSC_LOG_SAMPLE_RESOURCES
DDS_C, 310
DDSC_LOG_SEQUENCE_COPY_EC
DDS_C, 309
DDSC_LOG_SEQUENCE_FINALIZE_EC
DDS_C, 309
DDSC_LOG_SEQUENCE_GETREF_EC
DDS_C, 309
DDSC_LOG_SEQUENCE_INITIALIZE_EC
DDS_C, 309
DDSC_LOG_SEQUENCE_INVALID_EC
DDS_C, 309
DDSC_LOG_SEQUENCE_SETLENGTH_EC
DDS_C, 309

DDSC_LOG_SEQUENCE_SETMAX_EC
DDS_C, 309
DDSC_LOG_SET_ENTITY_NAME_EC
DDS_C, 300
DDSC_LOG_STRING_OBJECT
DDS_C, 307
DDSC_LOG_STRING_RECORD
DDS_C, 304
DDSC_LOG_STRINGMANAGER_OBJECT
DDS_C, 307
DDSC_LOG_SUBSCRIBER_ENTITY
DDS_C, 314
DDSC_LOG_SUBSCRIBER_LISTENER
DDS_C, 314
DDSC_LOG_SUBSCRIBER_QOS
DDS_C, 311
DDSC_LOG_SUBSCRIBER_RECORD
DDS_C, 304
DDSC_LOG_SUBSCRIBERQOS_OBJECT
DDS_C, 305
DDSC_LOG_SUBSCRIPTION_ENTITY
DDS_C, 315
DDSC_LOG_SUBSCRIPTION_RECORD
DDS_C, 303
DDSC_LOG_SUBSCRIPTIONDATA_OBJECT
DDS_C, 305
DDSC_LOG_SYS_GET_HOSTNAME_EC
DDS_C, 300
DDSC_LOG_SYS_GETTIME_EC
DDS_C, 300
DDSC_LOG_SYSTEM_OBJECT
DDS_C, 306
DDSC_LOG_TABLE_CREATE_EC
DDS_C, 303
DDSC_LOG_TABLE_DELETE_EC
DDS_C, 303
DDSC_LOG_TABLE_INUSE_EC
DDS_C, 303
DDSC_LOG_TABLE_SELECT_EC
DDS_C, 303
DDSC_LOG_TIMER_CREATE_TIMEOUT_EC
DDS_C, 302
DDSC_LOG_TIMER_OBJECT
DDS_C, 307
DDSC_LOG_TIMEROUT_OBJECT
DDS_C, 307
DDSC_LOG_TOPIC_ENTITY
DDS_C, 314
DDSC_LOG_TOPIC_FIND_EC
DDS_C, 301
DDSC_LOG_TOPIC_LISTENER
DDS_C, 313
DDSC_LOG_TOPIC_NAME_CMP_EC
DDS_C, 320
DDSC_LOG_TOPIC_NARROW_EC
DDS_C, 302
DDSC_LOG_TOPIC_OBJECT
DDS_C, 307
DDSC_LOG_TOPIC_QOS
DDS_C, 311
DDSC_LOG_TOPIC_RECORD
DDS_C, 303
DDSC_LOG_TOPIC_TOO_LONG_EC
DDS_C, 320
DDSC_LOG_TOPICQOS_OBJECT
DDS_C, 305
DDSC_LOG_TRANSPORT_COMPONENT
DDS_C, 310
DDSC_LOG_TRANSPORT_QOS_POLICY
DDS_C, 312
DDSC_LOG_TYPE_FUNCTION_COPY_SAMPLE
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_CREATE_SAMPLE
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_DELETE_SAMPLE
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_DESERIALIZE_DATA
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_GET_KEY_KIND
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_GET_SERIALIZED_-
SAMPLE_MAX_SIZE
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_INSTANCE_TO_KE-
YHASH
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_NULL_EC
DDS_C, 320
DDSC_LOG_TYPE_FUNCTION_SERIALIZE_DATA
DDS_C, 320
DDSC_LOG_TYPE_GET_SAMPLE_ID_EC
DDS_C, 321
DDSC_LOG_TYPE_KEY_TYPE_EC
DDS_C, 320
DDSC_LOG_TYPE_MANAGED_POOL_FAILURE_EC
DDS_C, 320
DDSC_LOG_TYPE_NAME_CMP_EC
DDS_C, 319
DDSC_LOG_TYPE_OBJECT
DDS_C, 306
DDSC_LOG_TYPE_RECORD
DDS_C, 304
DDSC_LOG_TYPE_SUPPORT_QOS_POLICY
DDS_C, 312
DDSC_LOG_TYPE_TOO_LONG_EC
DDS_C, 320
DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_-
KEY_EC
DDS_C, 301
DDSC_LOG_UNKNOWN_REMOTE_PARTICIPANT_-
NAME_EC
DDS_C, 301
DDSC_LOG_USER_TRAFFIC_QOS_POLICY
DDS_C, 313
DDSC_LOG_USERMULTICAST_SEQUENCE
DDS_C, 309

DDSC_LOG_USERUNICAST_SEQUENCE
DDS_C, 309
DDSC_LOG_UUID_OBJECT
DDS_C, 307
DDSC_LOG_WAITSET_OBJECT
DDS_C, 307
DDSC_LOG_WRITERHISTORY_COMPONENT
DDS_C, 309
DDSC_LOG_WS_ADD_COND_REFERENCE_EC
DDS_C, 302
DDSC_LOG_WS_REMOVE_COND_REFERENCE_EC
DDS_C, 302
DDSC_LOG_WSREF_OBJECT
DDS_C, 306
DDSHST_READER_DEFAULT_HISTORY_NAME
dds_c_rh_plugin.h, 496
DDSHST_WRITER_DEFAULT_HISTORY_NAME
dds_c_wh_plugin.h, 504
DEADLINE, 250
DDS_DEADLINE_QOS_POLICY_NAME, 250
DESTINATION_ORDER, 260
DDS_BY_RECEPTION_TIMESTAMP_DESTINATIONORDER_QOS, 260
DDS_BY_SOURCE_TIMESTAMP_DESTINATIONORDER_QOS, 260
DDS_DestinationOrderQosPolicyKind, 260
DISCOVERY, 231
DDS_DISCOVERY_QOS_POLICY_NAME, 231
DOMAIN_PARTICIPANT_RESOURCE_LIMITS, 273
DDS_DOMAINPARTICIPANTRESOURCELIMITS_QOS_POLICY_NAME, 273
DPSE, 338
DPSE_LOG_ADD_ANONYMOUS_ROUTE_EC, 344
DPSE_LOG_ADVANCE_SN_EC, 343
DPSE_LOG_ANNOUNCE_LOCAL_PARTICIPANT_DELETION_EC, 341
DPSE_LOG_ANNOUNCE_WRITE_EC, 343
DPSE_LOG_ANNOUNCEMENT_EC, 343
DPSE_LOG_ASSERT_REMOTE_PARTICIPANT_EC, 343
DPSE_LOG_CDR_DESERIALIZE_EC, 342
DPSE_LOG_CDR_SERIALIZE_EC, 342
DPSE_LOG_CDR_SET_POSITION_EC, 342
DPSE_LOG_CHECKSUM_INCOMPATIBLE_PARTICIPANT_IGNORED_EC, 345
DPSE_LOG_CREATE_DISCOVERY_PUBLISHER_EC, 340
DPSE_LOG_CREATE_DISCOVERY_SUBSCRIBER_EC, 341
DPSE_LOG_CREATE_READER_EC, 341
DPSE_LOG_CREATE_TOPIC_EC, 341
DPSE_LOG_CREATE_WRITER_EC, 341
DPSE_LOG_DELETE_ANONYMOUS_ROUTE_EC, 344
DPSE_LOG_DELETE_PUBLISHER_EC, 341
DPSE_LOG_DELETE_READER_EC, 341
DPSE_LOG_DELETE_SUBSCRIBER_EC, 341
DPSE_LOG_DELETE_TOPIC_EC, 341
DPSE_LOG_DELETE_WRITER_EC, 341
DPSE_LOG_DESERIALIZE_BUILTIN_ENDPOINTS_EC, 343
DPSE_LOG_DESERIALIZE_GUID_EC, 343
DPSE_LOG_DESERIALIZE_PARTICIPANT_NAME_EC, 343
DPSE_LOG_DESERIALIZE_PRODUCT_VERSION_EC, 342
DPSE_LOG_DESERIALIZE_PROTOCOL_VERSION_EC, 343
DPSE_LOG_DESERIALIZE_TOO_MANY_LOCATORS_EC, 343
DPSE_LOG_DESERIALIZE_UNKNOWN_PID_EC, 343
DPSE_LOG_DESERIALIZE_VENDOR_ID_EC, 343
DPSE_LOG_DISPOSE_EC, 341
DPSE_LOG_ENTITY_ENABLE_EC, 344
DPSE_LOG_GET_DDS_PROPERTIES_EC, 344
DPSE_LOG_GET_TIMER_EC, 345
DPSE_LOG_INVALID_DISCOVERY_SAMPLE_EC, 344
DPSE_LOG_INVALID_PEER_ADDRESS_EC, 340
DPSE_LOG_MAX_REMOTE_PARTICIPANT_EC, 344
DPSE_LOG_OBJECT_ALLOCATE_EC, 341
DPSE_LOG_OBJECT_DELETE_EC, 342
DPSE_LOG_OBJECT_FINALIZE_EC, 341
DPSE_LOG_OBJECT_INITIALIZE_EC, 341
DPSE_LOG_OBJECT_INVALID_EC, 342
DPSE_LOG_ON_ASSERT_REMOTE_PARTICIPANT_EC, 344
DPSE_LOG_PARTICIPANT_TAKE_EC, 343
DPSE_LOG_REFRESH_REMOTE_PARTICIPANT_EC, 344
DPSE_LOG_REGISTER_TYPE_EC, 341
DPSE_LOG_RESET_REMOTE_PARTICIPANT_EC, 344
DPSE_LOG_RETURN_DISCOVERY_SAMPLE_EC, 344
DPSE_LOG_SCHEDULE_FAST_ASSERTION_EC, 343
DPSE_LOG_SEQUENCE_COPY_EC, 345
DPSE_LOG_SEQUENCE_FINALIZE_EC, 345
DPSE_LOG_SEQUENCE_GETREF_EC, 344
DPSE_LOG_SEQUENCE_INITIALIZE_EC, 344
DPSE_LOG_SEQUENCE_SETLENGTH_EC, 344
DPSE_LOG_SEQUENCE_SETMAX_EC, 344
DPSE_LOG_SERIALIZE_BUILTIN_ENDPOINTS_EC, 342
DPSE_LOG_SERIALIZE_DEFAULT_UNICAST_EC, 342
DPSE_LOG_SERIALIZE_GUID_EC, 342
DPSE_LOG_SERIALIZE_LEASE_DURATION_EC, 342
DPSE_LOG_SERIALIZE_META_MULTICAST_E-

C, 342
DPSE_LOG_SERIALIZE_META_UNICAST_EC, 342
DPSE_LOG_SERIALIZE_MULTICAST_EC, 344
DPSE_LOG_SERIALIZE_PARTICIPANT_NAME_EC, 343
DPSE_LOG_SERIALIZE_PRODUCT_VERSION_EC, 342
DPSE_LOG_SERIALIZE_PROTOCOL_VERSION_EC, 342
DPSE_LOG_SERIALIZE_VENDOR_ID_EC, 342
DPSE_LOG_UNSUPPORTED_LOCATOR_EC, 345
DPSE_LOG_UPDATE_PARTICIPANT_ASSERT_PERIOD_EC, 343
DPSE Static Discovery API, 121
DPSE_DiscoveryFactory_get_interface, 123
DPSE_DiscoveryPluginProperty_INITIALIZER, 121
DPSE_DiscoveryPluginProperty_initialize, 122
DPSE_RemoteParticipant_assert, 122
DPSE_RemotePublication_assert, 122
DPSE_RemoteSubscription_assert, 123
DPSE_DISCOVERYPLUGIN_OBJECT
disc_dpse_log.h, 509
DPSE_DiscoveryFactory_get_interface
DPSE Static Discovery API, 123
DPSE_DiscoveryPluginProperty, 442
initial_participant_announcement_period, 443
initial_participant_announcements, 443
max_locators_per_discovered_participant, 443
max_participant_locators, 443
participant_liveliness_assert_period, 442
participant_liveliness_lease_duration, 442
DPSE_DiscoveryPluginProperty_INITIALIZER
DPSE Static Discovery API, 121
DPSE_DiscoveryPluginProperty_initialize
DPSE Static Discovery API, 122
DPSE_ENABLEDPARTICIPANTINDEX_OBJECT
disc_dpse_log.h, 509
DPSE_KEYLIST_OBJECT
disc_dpse_log.h, 509
DPSE_LOG_ADD_ANONYMOUS_ROUTE_EC
DPSE, 344
DPSE_LOG_ADVANCE_SN_EC
DPSE, 343
DPSE_LOG_ANNOUNCE_LOCAL_PARTICIPANT_DELETION_EC
DPSE, 341
DPSE_LOG_ANNOUNCE_WRITE_EC
DPSE, 343
DPSE_LOG_ANNOUNCEMENT_EC
DPSE, 343
DPSE_LOG_ASSERT_REMOTE_PARTICIPANT_EC
DPSE, 343
DPSE_LOG_CDR_DESERIALIZE_EC
DPSE, 342
DPSE_LOG_CDR_LONG_KIND
disc_dpse_log.h, 509
DPSE_LOG_CDR_PID_KIND
disc_dpse_log.h, 510
DPSE_LOG_CDR_SERIALIZE_EC
DPSE, 342
DPSE_LOG_CDR_SET_POSITION_EC
DPSE, 342
DPSE_LOG_CHECKSUM_INCOMPATIBLE_PARTICIPANT_IGNORED_EC
DPSE, 345
DPSE_LOG_CREATE_DISCOVERY_PUBLISHER_EC
DPSE, 340
DPSE_LOG_CREATE_DISCOVERY_SUBSCRIBER_EC
DPSE, 341
DPSE_LOG_CREATE_READER_EC
DPSE, 341
DPSE_LOG_CREATE_TOPIC_EC
DPSE, 341
DPSE_LOG_CREATE_WRITER_EC
DPSE, 341
DPSE_LOG_DEFAULT_MULTICAST_LOCATOR_SEQUENCE
disc_dpse_log.h, 510
DPSE_LOG_DEFAULT_UNICAST_LOCATOR_SEQUENCE
disc_dpse_log.h, 510
DPSE_LOG_DELETE_ANONYMOUS_ROUTE_EC
DPSE, 344
DPSE_LOG_DELETE_PUBLISHER_EC
DPSE, 341
DPSE_LOG_DELETE_READER_EC
DPSE, 341
DPSE_LOG_DELETE_SUBSCRIBER_EC
DPSE, 341
DPSE_LOG_DELETE_TOPIC_EC
DPSE, 341
DPSE_LOG_DELETE_WRITER_EC
DPSE, 341
DPSE_LOG_DESERIALIZE_BUILTIN_ENDPOINTS_EC
DPSE, 343
DPSE_LOG_DESERIALIZE_GUID_EC
DPSE, 343
DPSE_LOG_DESERIALIZE_PARTICIPANT_NAME_EC
DPSE, 343
DPSE_LOG_DESERIALIZE_PRODUCT_VERSION_EC
DPSE, 342
DPSE_LOG_DESERIALIZE_PROTOCOL_VERSION_EC
DPSE, 343
DPSE_LOG_DESERIALIZE_TOO_MANY_LOCATORS_EC
DPSE, 343
DPSE_LOG_DESERIALIZE_UNKNOWN_PID_EC
DPSE, 343

DPSE_LOG_DESERIALIZE_VENDOR_ID_EC
DPSE, 343
DPSE_LOG_DISPOSE_EC
DPSE, 341
DPSE_LOG_ENTITY_ENABLE_EC
DPSE, 344
DPSE_LOG_GET_DDS_PROPERTIES_EC
DPSE, 344
DPSE_LOG_GET_TIMER_EC
DPSE, 345
DPSE_LOG_INVALID_DISCOVERY_SAMPLE_EC
DPSE, 344
DPSE_LOG_INVALID_PEER_ADDRESS_EC
DPSE, 340
DPSE_LOG_MAX_REMOTE_PARTICIPANT_EC
DPSE, 344
DPSE_LOG_META_MULTICAST_LOCATOR_SEQUENCE
disc_dpse_log.h, 510
DPSE_LOG_META_UNICAST_LOCATOR_SEQUENCE
disc_dpse_log.h, 510
DPSE_LOG_OBJECT_ALLOCATE_EC
DPSE, 341
DPSE_LOG_OBJECT_DELETE_EC
DPSE, 342
DPSE_LOG_OBJECT_FINALIZE_EC
DPSE, 341
DPSE_LOG_OBJECT_INITIALIZE_EC
DPSE, 341
DPSE_LOG_OBJECT_INVALID_EC
DPSE, 342
DPSE_LOG_ON_ASSERT_REMOTE_PARTICIPANT_EC
DPSE, 344
DPSE_LOG_PARTICIPANT_TAKE_EC
DPSE, 343
DPSE_LOG_REFRESH_REMOTE_PARTICIPANT_EC
DPSE, 344
DPSE_LOG_REGISTER_TYPE_EC
DPSE, 341
DPSE_LOG_RESET_REMOTE_PARTICIPANT_EC
DPSE, 344
DPSE_LOG_RETURN_DISCOVERY_SAMPLE_EC
DPSE, 344
DPSE_LOG_SCHEDULE_FAST_ASSERTION_EC
DPSE, 343
DPSE_LOG_SEQUENCE_COPY_EC
DPSE, 345
DPSE_LOG_SEQUENCE_FINALIZE_EC
DPSE, 345
DPSE_LOG_SEQUENCE_GETREF_EC
DPSE, 344
DPSE_LOG_SEQUENCE_INITIALIZE_EC
DPSE, 344
DPSE_LOG_SEQUENCE_SETLENGTH_EC
DPSE, 344
DPSE_LOG_SEQUENCE_SETMAX_EC
DPSE, 344
DPSE_LOG_SERIALIZE_BUILTIN_ENDPOINTS_EC
DPSE, 342
DPSE_LOG_SERIALIZE_DEFAULT_UNICAST_EC
DPSE, 342
DPSE_LOG_SERIALIZE_GUID_EC
DPSE, 342
DPSE_LOG_SERIALIZE_LEASE_DURATION_EC
DPSE, 342
DPSE_LOG_SERIALIZE_META_MULTICAST_EC
DPSE, 342
DPSE_LOG_SERIALIZE_META_UNICAST_EC
DPSE, 342
DPSE_LOG_SERIALIZE_MULTICAST_EC
DPSE, 344
DPSE_LOG_SERIALIZE_PARTICIPANT_NAME_EC
DPSE, 343
DPSE_LOG_SERIALIZE_PRODUCT_VERSION_EC
DPSE, 342
DPSE_LOG_SERIALIZE_PROTOCOL_VERSION_EC
DPSE, 342
DPSE_LOG_SERIALIZE_VENDOR_ID_EC
DPSE, 342
DPSE_LOG_UNSUPPORTED_LOCATOR_EC
DPSE, 345
DPSE_LOG_UPDATE_PARTICIPANT_ASSERT_PERIOD_EC
DPSE, 343
DPSE_PARTICIPANTANNOUCEMENT_OBJECT
disc_dpse_log.h, 509
DPSE_PARTICIPANTBUILTINTOPICDATA_OBJECT
disc_dpse_log.h, 509
DPSE_PARTICIPANTENTITYNAME_OBJECT
disc_dpse_log.h, 509
DPSE_PARTICIPANTMAMES_OBJECT
disc_dpse_log.h, 509
DPSE_PARTICIPANTNAMEINDEX_OBJECT
disc_dpse_log.h, 509
DPSE_PARTICIPANTPOOL_OBJECT
disc_dpse_log.h, 509
DPSE_PUBLISHER_ENTITY
disc_dpse_log.h, 510
DPSE_RemoteParticipant_assert
DPSE Static Discovery API, 122
DPSE_RemotePublication_assert
DPSE Static Discovery API, 122
DPSE_RemoteSubscription_assert
DPSE Static Discovery API, 123
DPSE_SUBSCRIBER_ENTITY
disc_dpse_log.h, 510
DPSE_TOPIC_ENTITY
disc_dpse_log.h, 510
DURABILITY, 257
DDS_DURABILITY_QOS_POLICY_NAME, 257
DDS_DurabilityQosPolicyKind, 257
DDS_TRANSIENT_LOCAL_DURABILITY_QOS, 257
DDS_VOLATILE_DURABILITY_QOS, 257

Data Sample, 101
DataReader, 80
DDS_DataReader, 84
DDS_DataReader_as_entity, 97
DDS_DataReader_get_instance_replaced_status, 100
DDS_DataReader_get_liveliness_changed_status, 98
DDS_DataReader_get_requested_deadline_missed_status, 98
DDS_DataReader_get_requested_incompatible_qos_status, 99
DDS_DataReader_get_sample_lost_status, 99
DDS_DataReader_get_sample_rejected_status, 98
DDS_DataReader_get_subscriber, 97
DDS_DataReader_get_subscription_matched_status, 99
DDS_DataReader_get_topicdescription, 97
DDS_DataReaderInstanceReplacedStatus_INITIALIZER, 84
DDS_DataReaderListener_BeforeSampleCommitCallback, 86
DDS_DataReaderListener_BeforeSampleDeserializeCallback, 85
DDS_DataReaderListener_DataAvailableCallback, 85
DDS_DataReaderListener_INITIALIZER, 84
DDS_DataReaderListener_InstanceReplacedCallback, 85
DDS_DataReaderListener_LivelinessChangedCallback, 85
DDS_DataReaderListener_RequestedDeadlineMissedCallback, 85
DDS_DataReaderListener_RequestedIncompatibleQosCallback, 85
DDS_DataReaderListener_SampleLostCallback, 85
DDS_DataReaderListener_SampleRejectedCallback, 85
DDS_DataReaderListener_SubscriptionMatchedCallback, 85
DDS_DataReaderQos_INITIALIZER, 84
DDS_DataReaderQos_copy, 96
DDS_DataReaderQos_initialize, 96
DDS_DataReaderQos_is_equal, 97
DDS_LivelinessChangedStatus_INITIALIZER, 83
DDS_NOT_REJECTED, 86
DDS_REJECTED_BY_INSTANCES_LIMIT, 86
DDS_REJECTED_BY_REMOTE_WRITERS_LIMIT, 86
DDS_REJECTED_BY_SAMPLES_LIMIT, 86
DDS_REJECTED_BY_SAMPLES_PER_INSTANCE_LIMIT, 86
DDS_REJECTED_BY_SAMPLES_PER_REMOTE_WRITER_LIMIT, 86
DDS_RequestedDeadlineMissedStatus_INITIALIZER, 83
DDS_RequestedIncompatibleQosStatus_INITIALIZER, 83
DDS_SAMPLE_LOST_BY_DATAWRITER, 87
DDS_SAMPLE_LOST_BY_HISTORY_DEPTH_LIMIT, 87
DDS_SAMPLE_LOST_BY_MANAGEDPOOL_OVERWRITE, 87
DDS_SAMPLE_LOST_BY_MAX_SAMPLES_LIMIT, 87
DDS_SAMPLE_LOST_BY_META_SAMPLE_LIMIT, 87
DDS_SAMPLE_LOST_BY_NOT_READ_ON_CACHE_DELETION, 87
DDS_SAMPLE_LOST_NOT_LOST, 86
DDS_SampleLostStatus_INITIALIZER, 84
DDS_SampleLostStatusKind, 86
DDS_SampleRejectedStatus_INITIALIZER, 83
DDS_SampleRejectedStatusKind, 86
DDS_SubscriptionMatchedStatus_INITIALIZER, 84
FooDataReader_as_datareader, 87
FooDataReader_is_data_consistent, 100
FooDataReader_lookup_instance, 95
FooDataReader_narrow, 87
FooDataReader_read, 87
FooDataReader_read_instance, 92
FooDataReader_read_next_sample, 91
FooDataReader_return_loan, 95
FooDataReader_take, 88
FooDataReader_take_instance, 94
FooDataReader_take_next_sample, 92
DataReader Resource Limits, 261
DDS_DATAREADERRESOURCELIMITS_QOS_POLICY_NAME, 261
DDS_DataReaderResourceLimitsInstanceReplacementKind, 261
DDS_NO_INSTANCE_REPLACEMENT_QOS, 261
DDS_REPLACE_OLDEST_INSTANCE_REPLACEMENT_QOS, 261
DataWriter, 52
DDS_DataWriter, 56
DDS_DataWriter_as_entity, 66
DDS_DataWriter_assert_liveliness, 66
DDS_DataWriter_get_liveliness_lost_status, 68
DDS_DataWriter_get_offered_deadline_missed_status, 68
DDS_DataWriter_get_offered_incompatible_qos_status, 68
DDS_DataWriter_get_publication_matched_status, 68
DDS_DataWriter_get_publisher, 67
DDS_DataWriter_get_reliable_reader_activity_changed_status, 69
DDS_DataWriter_get_topic, 67
DDS_DataWriterListener_INITIALIZER, 55
DDS_DataWriterListener_LivelinessLostCallback, 56

DDS_DataWriterListener_OfferedDeadlineMissed-Callback, 56
DDS_DataWriterListener_OfferedIncompatible-QosCallback, 56
DDS_DataWriterListener_PublicationMatched-Callback, 57
DDS_DataWriterListener_ReliableReaderActivity-ChangedCallback, 57
DDS_DataWriterListener_ReliableSampleUnacknowledged-Callback, 57
DDS_DataWriterQos_INITIALIZER, 55
DDS_DataWriterQos_copy, 65
DDS_DataWriterQos_initialize, 65
DDS_DataWriterQos_is_equal, 66
DDS_LivelinessLostStatus_INITIALIZER, 54
DDS_OfferedDeadlineMissedStatus_INITIALIZER, 54
DDS_OfferedIncompatibleQosStatus_INITIALIZE-R, 55
DDS_PublicationMatchedStatus_INITIALIZER, 55
DDS_ReliableReaderActivityChangedStatus_INIT-IALIZER, 55
FooDataWriter_as_datawriter, 57
FooDataWriter_discard_loan, 70
FooDataWriter_dispose, 61
FooDataWriter_dispose_w_timestamp, 62
FooDataWriter_get_loan, 69
FooDataWriter_narrow, 57
FooDataWriter_register_instance, 58
FooDataWriter_register_instance_w_timestamp, 58
FooDataWriter_unregister_instance, 59
FooDataWriter_unregister_instance_w_timestamp, 60
FooDataWriter_write, 63
FooDataWriter_write_w_timestamp, 64
DataWriter Resource Limits, 262
DDS_DATAWRITERRESOURCELIMITS_QOS_P-OLICY_NAME, 262
db_dll.h, 465
db_log.h, 465
dds_builtin_endpoints
DDS_ParticipantBuiltinTopicData, 403
dds_c_common.h, 467
dds_c_common_impl.h, 468
dds_c_config.h, 468
dds_c_discovery.h, 468
dds_c_discovery_plugin.h, 469
dds_c_dll.h, 469
dds_c_domain.h, 469
DDS_DiscoveryComponent_INITIALIZER, 472
dds_c_infrastructure.h, 472
DDS_BUILTINTOPICKEY_AUTO, 480
DDS_BUILTINTOPICKEY_UNKNOWN, 480
DDS_DATAREADER_ENTITY_KIND, 480
DDS_DATAWRITER_ENTITY_KIND, 480
DDS_EntityKind_t, 480
DDS_PARTICIPANT_ENTITY_KIND, 480
DDS_PRODUCTVERSION_UNKNOWN, 479
DDS_PUBLISHER_ENTITY_KIND, 480
DDS_RTPS_AUTO_ID, 480
DDS_SUBSCRIBER_ENTITY_KIND, 480
DDS_TOPIC_ENTITY_KIND, 480
DDS_UNKNOWN_ENTITY_KIND, 480
T, 479, 480
dds_c_log.h, 480
dds_c_publication.h, 493
dds_c_rh_plugin.h, 496
DDSHST_READER_DEFAULT_HISTORY_NAM-E, 496
dds_c_sequence.h, 496
dds_c_string.h, 497
dds_c_subscription.h, 497
dds_c_topic.h, 502
dds_c_type.h, 503
dds_c_wh_plugin.h, 504
DDSHST_WRITER_DEFAULT_HISTORY_NAME, 504
deadline
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_PublicationBuiltinTopicData, 410
DDS_SubscriptionBuiltinTopicData, 432
default_multicast_locators
DDS_ParticipantBuiltinTopicData, 403
default_unicast_locators
DDS_ParticipantBuiltinTopicData, 403
delete_route
ZCOPY_NotifUserInterfaceI, 464
delete_sample
NDDS_Type_Plugin, 447
deny_interface
UDP_InterfaceFactoryProperty, 461
depth
DDS_HistoryQosPolicy, 388
deserialize_data
NDDS_Type_Plugin, 446
destination_order
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_PublicationBuiltinTopicData, 410
DDS_SubscriptionBuiltinTopicData, 433
disc_dpse_dll.h, 504
disc_dpse_dpsediscovery.h, 505
disc_dpse_log.h, 505
DPSE_DISCOVERYPLUGIN_OBJECT, 509
DPSE_ENABLEDPARTICIPANTINDEX_OBJECT, 509
DPSE_KEYLIST_OBJECT, 509
DPSE_LOG_CDR_LONG_KIND, 509
DPSE_LOG_CDR_PID_KIND, 510
DPSE_LOG_DEFAULT_MULTICAST_LOCATOR-_SEQUENCE, 510
DPSE_LOG_DEFAULT_UNICAST_LOCATOR_S-EQUENCE, 510

DPSE_LOG_META_MULTICAST_LOCATOR_SEQUENCE, 510
DPSE_LOG_META_UNICAST_LOCATOR_SEQUENCE, 510
DPSE_PARTICIPANTANNOUCEMENT_OBJECT, 509
DPSE_PARTICIPANTBUILTINTOPICDATA_OBJECT, 509
DPSE_PARTICIPANTENTITYNAME_OBJECT, 509
DPSE_PARTICIPANTMAMES_OBJECT, 509
DPSE_PARTICIPANTNAMEINDEX_OBJECT, 509
DPSE_PARTICIPANTPOOL_OBJECT, 509
DPSE_PUBLISHER_ENTITY, 510
DPSE_SUBSCRIBER_ENTITY, 510
DPSE_TOPIC_ENTITY, 510
Discovery, 32
DDS_BuiltinTopicKey_t, 34
DDS_LOCATOR_INVALID, 33
DDS_LOCATOR_KIND_RESERVED, 34
DDS_LOCATOR_KIND_SHMEM, 34
DDS_LOCATOR_KIND_UDPv4, 33
DDS_LOCATOR_KIND_UDPv6, 33
DDS_PROTOCOLVERSION, 34
DDS_PROTOCOLVERSION_1_0, 34
DDS_PROTOCOLVERSION_1_1, 34
DDS_PROTOCOLVERSION_1_2, 34
DDS_PROTOCOLVERSION_2_0, 34
DDS_PROTOCOLVERSION_2_1, 34
DDS_ProtocolVersion_t, 34
DDS_VENDOR_ID_LENGTH_MAX, 34
discovery
DDS_DiscoveryQosPolicy, 377
DDS_DomainParticipantQos, 379
Documentation Guide, 113
Domain Module, 13
DDS_DomainId_t, 14
domain_id_gain
DDS_RtpsWellKnownPorts_t, 423
DomainParticipant, 22
DDS_DomainParticipant, 24
DDS_DomainParticipant_add_peer, 28
DDS_DomainParticipant_as_entity, 25
DDS_DomainParticipant_create_publisher, 26
DDS_DomainParticipant_create_subscriber, 26
DDS_DomainParticipant_create_topic, 27
DDS_DomainParticipant_get_current_time, 29
DDS_DomainParticipant_get_domain_id, 29
DDS_DomainParticipant_lookup_topicdescription, 28
DDS_DomainParticipant_register_type, 30
DDS_DomainParticipantListener_INITIALIZER, 23
DDS_DomainParticipantQos_INITIALIZER, 23
DDS_DomainParticipantQos_copy, 24
DDS_DomainParticipantQos_initialize, 24
DDS_DomainParticipantQos_is_equal, 25
DDS_PUBLISHER_QOS_DEFAULT, 31
DDS_SUBSCRIBER_QOS_DEFAULT, 31
DDS_TOPIC_QOS_DEFAULT, 30
DomainParticipantFactory, 15
DDS_DomainParticipantFactory, 16
DDS_DomainParticipantFactory_create_participant, 18
DDS_DomainParticipantFactory_get_instance, 18
DDS_DomainParticipantFactory_get_qos, 19
DDS_DomainParticipantFactory_get_registry, 20
DDS_DomainParticipantFactory_lookup_participant, 19
DDS_DomainParticipantFactory_set_qos, 20
DDS_DomainParticipantFactoryQos_INITIALIZER, 16
DDS_DomainParticipantFactoryQos_copy, 17
DDS_DomainParticipantFactoryQos_initialize, 17
DDS_DomainParticipantFactoryQos_is_equal, 17
DDS_PARTICIPANT_FACTORY_QOS_DEFAULT, 21
DDS_PARTICIPANT_QOS_DEFAULT, 21
DDS_TheParticipantFactory, 16
durability
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_PublicationBuiltinTopicData, 410
DDS_SubscriptionBuiltinTopicData, 432

ENTITY_FACTORY, 263
DDS_ENTITYFACTORY_QOS_POLICY_NAME, 263
ENTITY_NAME, 286
DDS_AUTO_NAME_ENTITY, 287
DDS_ENTITYNAME_QOS_NAME_MAX, 286
DDS_ENTITYNAME_QOS_POLICY_NAME, 287
DDS_EntityNameQosPolicy_set_name, 286
enabled_transports
DDS_DiscoveryQosPolicy, 376
DDS_TransportQosPolicy, 437
DDS_UserTrafficQosPolicy, 437
Entity Support, 281
DDS_DomainEntity, 283
DDS_Entity, 282
DDS_Entity_enable, 283
DDS_Entity_get_instance_handle, 284
DDS_Entity_get_status_changes, 285
DDS_Entity_get_statuscondition, 284
DDS_Listener_INITIALIZER, 281
entity_factory
DDS_DomainParticipantFactoryQos, 378
DDS_DomainParticipantQos, 379
DDS_PublisherQos, 412
DDS_SubscriberQos, 431
Extended Qos Support, 264

flags
NETIO_DGRAM_InterfaceTableEntry, 452
UDP_InterfaceTableEntry, 462
FooDataReader, 443
FooDataReader_as_datareader
DataReader, 87

FooDataReader_is_data_consistent
DataReader, 100
FooDataReader_lookup_instance
DataReader, 95
FooDataReader_narrow
DataReader, 87
FooDataReader_read
DataReader, 87
FooDataReader_read_instance
DataReader, 92
FooDataReader_read_next_sample
DataReader, 91
FooDataReader_return_loan
DataReader, 95
FooDataReader_take
DataReader, 88
FooDataReader_take_instance
DataReader, 94
FooDataReader_take_next_sample
DataReader, 92
FooDataWriter, 444
FooDataWriter_as_datawriter
DataWriter, 57
FooDataWriter_discard_loan
DataWriter, 70
FooDataWriter_dispose
DataWriter, 61
FooDataWriter_dispose_w_timestamp
DataWriter, 62
FooDataWriter_get_loan
DataWriter, 69
FooDataWriter_narrow
DataWriter, 57
FooDataWriter_register_instance
DataWriter, 58
FooDataWriter_register_instance_w_timestamp
DataWriter, 58
FooDataWriter_unregister_instance
DataWriter, 59
FooDataWriter_unregister_instance_w_timestamp
DataWriter, 60
FooDataWriter_write
DataWriter, 63
FooDataWriter_write_w_timestamp
DataWriter, 64
FooSeq, 444
_contiguous_buffer, 445
_element_size, 445
_flags, 445
_length, 445
_maximum, 445
_token1, 445
_token2, 445
max_alloc_size, 445
FooSeq_copy
Sequence Support, 327
FooSeq_copy_w_max
Sequence Support, 327
FooSeq_get_length
Sequence Support, 326
FooSeq_get_maximum
Sequence Support, 324
FooSeq_get_reference
Sequence Support, 327
FooSeq_initialize
Sequence Support, 324
FooSeq_set_length
Sequence Support, 326
FooSeq_set_maximum
Sequence Support, 324
FooSeq_set_maximum_w_max
Sequence Support, 325
FooTypeSupport, 445
FooTypeSupport_create_data
User Data Type Support, 44
FooTypeSupport_register_type
User Data Type Support, 44

GUID Support, 234
DDS_GUID_UNKNOWN, 234
generate_uuid
OSAPI_SystemI, 456
get_hostname
OSAPI_SystemI, 456
get_interface_list
NETIO_DGRAM_InterfaceI, 450
get_key_kind
NDDS_Type_Plugin, 447
get_route_table
NETIO Interface API, 194
ZCOPY_NotifUserInterfaceI, 464
get_serialized_key_max_size
NDDS_Type_Plugin, 447
get_serialized_sample_max_size
NDDS_Type_Plugin, 447
get_ticktime
OSAPI_SystemI, 457
get_time
OSAPI_SystemI, 456
get_timer_resolution
OSAPI_SystemI, 456

HISTORY, 256
DDS_HISTORY_QOS_POLICY_NAME, 256
DDS_HistoryQosPolicyKind, 256
DDS_KEEP_LAST_HISTORY_QOS, 256
heartbeat_period
DDS_RtpsReliableWriterProtocol_t, 421
heartbeats_per_max_samples
DDS_RtpsReliableWriterProtocol_t, 421
high
DDS_SequenceNumber_t, 430
history
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
hostname
OSAPI_SystemProperty, 457

if_table
UDP_InterfaceFactoryProperty, 462
ifname
NETIO_DGRAM_InterfaceTableEntry, 453
UDP_InterfaceTableEntry, 462
inactive_count
DDS_ReliableReaderActivityChangedStatus, 414
inactive_count_change
DDS_ReliableReaderActivityChangedStatus, 414
Infrastructure Module, 108
initial_participant_announcement_period
DPSE_DiscoveryPluginProperty, 443
initial_participant_announcements
DPSE_DiscoveryPluginProperty, 443
initial_peers
DDS_DiscoveryQosPolicy, 376
Instance States, 106
DDS_ALIVE_INSTANCE_STATE, 107
DDS_ANY_INSTANCE_STATE, 107
DDS_InstanceStateKind, 106
DDS_InstanceStateMask, 106
DDS_NOT_ALIVE_DISPOSED_INSTANCE_STATE, 107
DDS_NOT_ALIVE_INSTANCE_STATE, 107
DDS_NOT_ALIVE_NO_WRITERS_INSTANCE_STATE, 107
instance_handle
DDS_ReliableSampleUnacknowledgedStatus, 415
DDS_SampleInfo, 427
instance_replacement
DDS_DataReaderResourceLimitsQosPolicy, 367
instance_state
DDS_SampleInfo, 426
instance_to_keyhash
NDDS_Type_Plugin, 447
is_anonymous
RTI_ManagementQosPolicy, 459
is_default_interface
UDP_InterfaceFactoryProperty, 462

key
DDS_ParticipantBuiltinTopicData, 403
DDS_PublicationBuiltinTopicData, 409
DDS_SubscriptionBuiltinTopicData, 432
kind
DDS_DestinationOrderQosPolicy, 375
DDS_DurabilityQosPolicy, 384
DDS_HistoryQosPolicy, 388
DDS_LivelinessQosPolicy, 395
DDS_Locator_t, 396
DDS_OwnershipQosPolicy, 401
DDS_ReliabilityQosPolicy, 413
NETIO_Address, 448

LIVELINESS, 253
DDS_AUTOMATIC_LIVELINESS_QOS, 253
DDS_LIVELINESS_QOS_POLICY_NAME, 254
DDS_LivelinessQosPolicyKind, 253
DDS_MANUAL_BY_PARTICIPANT_LIVELINESS_QOS, 253
DDS_MANUAL_BY_TOPIC_LIVELINESS_QOS, 253
last_instance_handle
DDS_DataReaderInstanceReplacedStatus, 360
DDS_OfferedDeadlineMissedStatus, 397
DDS_ReliableReaderActivityChangedStatus, 414
DDS_RequestedDeadlineMissedStatus, 415
DDS_SampleRejectedStatus, 429
last_policy_id
DDS_OfferedIncompatibleQosStatus, 398
DDS_RequestedIncompatibleQosStatus, 416
last_publication_handle
DDS_LivelinessChangedStatus, 392
DDS_SubscriptionMatchedStatus, 434
last_reason
DDS_SampleRejectedStatus, 429
last_replacement_instance
DDS_DataReaderInstanceReplacedStatus, 360
last_subscription_handle
DDS_PublicationMatchedStatus, 411
lease_duration
DDS_LivelinessQosPolicy, 395
listener_data
DDS_Listener, 392
liveliness
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_PublicationBuiltinTopicData, 410
DDS_SubscriptionBuiltinTopicData, 432
liveliness_lease_duration
DDS_ParticipantBuiltinTopicData, 403
local_publisher_allocation
DDS_DomainParticipantResourceLimitsQosPolicy, 381
local_reader_allocation
DDS_DomainParticipantResourceLimitsQosPolicy, 381
local_subscriber_allocation
DDS_DomainParticipantResourceLimitsQosPolicy, 381
local_topic_allocation
DDS_DomainParticipantResourceLimitsQosPolicy, 381
local_type_allocation
DDS_DomainParticipantResourceLimitsQosPolicy, 381
local_writer_allocation
DDS_DomainParticipantResourceLimitsQosPolicy, 381
Log API, 148
OSAPI_LOG_VERBOSITY_DEBUG, 149
OSAPI_LOG_VERBOSITY_ERROR, 149
OSAPI_LOG_VERBOSITY_SILENT, 149
OSAPI_LOG_VERBOSITY_WARNING, 149
OSAPI_Log_clear, 151
OSAPI_Log_get_display_handler, 151

OSAPI_Log_get_last_error_code, 152
OSAPI_Log_get_log_handler, 150
OSAPI_Log_get_property, 152
OSAPI_Log_get_trace_handler, 150
OSAPI_Log_set_display_handler, 151
OSAPI_Log_set_log_handler, 149
OSAPI_Log_set_property, 152
OSAPI_Log_set_trace_handler, 150
OSAPI_Log_set_verbosity, 151
OSAPI_Log_write_buffer_T, 149
OSAPI_LogHandler_T, 149
OSAPI_LogVerbosity_T, 149
OSAPI_Trace_set_trace_mask, 151
Log Codes, 354
log_detail
OSAPI_LogProperty, 455
lost_samples
DDS_DataReaderInstanceReplacedStatus, 360
low
DDS_SequenceNumber_t, 430

major
DDS_ProductVersion, 407
DDS_ProductVersion_t, 407
DDS_ProtocolVersion, 408
majorRev
NDDS_Type_PluginVersion, 447
mask
NETIO_Netmask, 454
matching_writer_reader_pair_allocation
DDS_DomainParticipantResourceLimitsQosPolicy, 382
max_alloc_size
FooSeq, 445
max_blocking_time
DDS_ReliabilityQosPolicy, 413
max_buffer_size
OSAPI_LogProperty, 455
max_components
DDS_SystemResourceLimitsQosPolicy, 435
max_destination_ports
DDS_DomainParticipantResourceLimitsQosPolicy, 383
max_heartbeat_retries
DDS_RtpsReliableWriterProtocol_t, 421
max_instances
DDS_ResourceLimitsQosPolicy, 418
max_locators_per_discovered_participant
DPSE_DiscoveryPluginProperty, 443
max_message_size
UDP_InterfaceFactoryProperty, 461
max_outstanding_reads
DDS_DataReaderResourceLimitsQosPolicy, 367
max_participant_locators
DPSE_DiscoveryPluginProperty, 443
max_participants
DDS_SystemResourceLimitsQosPolicy, 434
max_receive_buffer_size
UDP_InterfaceFactoryProperty, 461
max_receive_ports
DDS_DomainParticipantResourceLimitsQosPolicy, 382
max_remote_readers
DDS_DataWriterResourceLimitsQosPolicy, 373
max_remote_writers
DDS_DataReaderResourceLimitsQosPolicy, 366
max_remote_writers_per_instance
DDS_DataReaderResourceLimitsQosPolicy, 366
max_routes_per_reader
DDS_DataWriterResourceLimitsQosPolicy, 373
max_routes_per_writer
DDS_DataReaderResourceLimitsQosPolicy, 367
max_samples
DDS_ResourceLimitsQosPolicy, 418
max_samples_per_instance
DDS_ResourceLimitsQosPolicy, 419
max_samples_per_notif
ZCOPY_NotifInterfaceFactoryProperty, 463
max_samples_per_remote_writer
DDS_DataReaderResourceLimitsQosPolicy, 367
max_send_buffer_size
UDP_InterfaceFactoryProperty, 461
max_send_message_size
UDP_InterfaceFactoryProperty, 461
max_send_window
DDS_RtpsReliableWriterProtocol_t, 421
metatraffic_multicast_locators
DDS_ParticipantBuiltinTopicData, 403
metatraffic_unicast_locators
DDS_ParticipantBuiltinTopicData, 403
minor
DDS_ProductVersion, 407
DDS_ProductVersion_t, 407
DDS_ProtocolVersion, 408
minorRev
NDDS_Type_PluginVersion, 447
mode
OSAPI_SharedMemorySegmentHandle, 455
OSAPI_SharedMemorySegmentHeader, 456
mtu
NETIO_DGRAM_InterfaceTableEntry, 452
multicast_interface
UDP_InterfaceFactoryProperty, 462
multicast_locator
DDS_SubscriptionBuiltinTopicData, 433
multicast_ttl
UDP_InterfaceFactoryProperty, 461

NDDS_Discovery_Property, 446
NDDS_PluginHelper_CreateManagedPoolFunc
User-Defined Types, 111
NDDS_PluginHelper_CreateTypedDataReaderFunc
User-Defined Types, 111
NDDS_PluginHelper_CreateTypedDataWriterFunc
User-Defined Types, 111
NDDS_PluginHelper_DeleteManagedPoolFunc
User-Defined Types, 111
NDDS_PluginHelper_DeleteTypedDataReaderFunc

User-Defined Types, 111
NDDS_PluginHelper_DeleteTypedDataWriterFunc
User-Defined Types, 111
NDDS_PluginHelper_GetKeyKindFunc
User-Defined Types, 111
NDDS_PluginHelper_InstanceToKeyHashFunc
User-Defined Types, 111
NDDS_PluginHelper_ManagedPoolDiscardLoanFunc
User-Defined Types, 111
NDDS_PluginHelper_ManagedPoolGetLoanFunc
User-Defined Types, 111
NDDS_PluginHelper_ManagedPoolGetSampleIDFunc
User-Defined Types, 111
NDDS_PluginHelper_NeedsOpaqueSamples
User-Defined Types, 112
NDDS_TYPEPLUGIN_NO_KEY
User-Defined Types, 112
NDDS_TYPEPLUGIN_USER_KEY
User-Defined Types, 112
NDDS_Transport_ZeroCopy_initialize
Zero Copy v2 API, 145
NDDS_Type_CopySampleFunc
User-Defined Types, 111
NDDS_Type_CreateSampleFunc
User-Defined Types, 110
NDDS_Type_DeleteSampleFunc
User-Defined Types, 111
NDDS_Type_GetSerializedSizeFunc
User-Defined Types, 110
NDDS_Type_Plugin, 446
copy_sample, 447
create_sample, 447
delete_sample, 447
deserialize_data, 446
get_key_kind, 447
get_serialized_key_max_size, 447
get_serialized_sample_max_size, 447
instance_to_keyhash, 447
serialize_data, 446
NDDS_Type_PluginVersion, 447
majorRev, 447
minorRev, 447
User-Defined Types, 110
NDDS_TypePluginKeyKind
User-Defined Types, 112
NETIO, 196
NETIO_LOG_AR_ADD_INVALID_INTERFACE_EC, 200
NETIO_LOG_AR_ADDRESS_RESOLVE_FAILED_EC, 200
NETIO_LOG_AR_ADDRESS_STRING_EXCEEDED_EC, 201
NETIO_LOG_AR_ALLOC_FAILED_EC, 200
NETIO_LOG_AR_CONTEXT_UNINITIALIZED_EC, 200
NETIO_LOG_AR_INVALID_TOKEN_EC, 200
NETIO_LOG_AR_PORT_RESOLVE_FAILED_EC, 200
NETIO_LOG_BIND_CREATE_RTABLE_EC, 199
NETIO_LOG_BIND_DELETE_RTABLE_EC, 199
NETIO_LOG_BIND_EXTERNAL_FAILED_EC, 200
NETIO_LOG_BIND_NEW_EC, 199
NETIO_LOG_BIND_NEW_RTABLE_EC, 199
NETIO_LOG_BIND_SET_LENGTH_EC, 200
NETIO_LOG_BIND_SET_MAX_EC, 200
NETIO_LOG_GETHOST_BYNAME_EC, 199
NETIO_LOG_ILLEGAL_ROUTE_OPERATION_EC, 201
NETIO_LOG_INVALID_ADDRESS_INDEX_EC, 201
NETIO_LOG_LOOP_BINDX_DB_EC, 201
NETIO_LOG_LOOP_CREATE_TABLE_EC, 201
NETIO_LOG_LOOP_CURSOR_ERROR_EC, 202
NETIO_LOG_LOOP_DELETE_TABLE_EC, 201
NETIO_LOG_LOOP_FWD_EC, 201
NETIO_LOG_LOOP_INITIALIZE_FAILED_EC, 202
NETIO_LOG_LOOP_INVALID_COMPONENT_EC, 202
NETIO_LOG_LOOP_INVALID_FACTORY_EC, 202
NETIO_LOG_LOOP_INVALID_PROPERTY_EC, 202
NETIO_LOG_LOOP_SELECT_TABLE_EC, 201
NETIO_LOG_LOOP_SET_LENGTH_EC, 202
NETIO_LOG_LOOP_UNBINDX_DB_EC, 201
NETIO_LOG_PACKET_INITIALIZE_EC, 201
NETIO_LOG_PACKET_SET_HEAD_EC, 201
NETIO_LOG_PACKET_SET_TAIL_EC, 201
NETIO_LOG_ROUTE_RTABLE_ADD_EC, 200
NETIO_LOG_ROUTE_RTABLE_ALLOC_EC, 200
NETIO_LOG_ROUTE_RTABLE_CREATE_EC, 200
NETIO_LOG_ROUTE_RTABLE_DELETE_EC, 200
NETIO_LOG_ROUTE_RTABLE_UPDATE_EC, 200
NETIO_LOG_SET_NAME_EC, 201
NETIO_LOG_UNBIND_EXTERNAL_FAILED_EC, 200
UDP_LOG_ALLOC_EC, 204
UDP_LOG_CURSOR_ERROR_EC, 205
UDP_LOG_DROPGRP_EC, 204
UDP_LOG_FINALIZE_EC, 203
UDP_LOG_GET_BUF_ALLOC_EC, 204
UDP_LOG_GET_BUF_SIZE_EC, 204
UDP_LOG_GET_IFLIST_EC, 204
UDP_LOG_GET_LENGTH_EC, 204
UDP_LOG_GET_NAMEINFO_EC, 204
UDP_LOG_GETHOSTNAME_EC, 202
UDP_LOG_INCONSISTENT_MAX_MESSSAGE_SIZE_EC, 205
UDP_LOG_INITIALIZE_FAILED_EC, 205
UDP_LOG_LOADLIB_EC, 204
UDP_LOG_MULTICAST_ENABLE_EC, 204

UDP_LOG_MULTICAST_NOT_ENABLED_EC, 205
UDP_LOG_NAT_DELETE_EC, 203
UDP_LOG_PACKET_FWD_EC, 203
UDP_LOG_PACKET_HEAD_EC, 203
UDP_LOG_PACKET_INIT_EC, 202
UDP_LOG_PORT_ADD_EC, 203
UDP_LOG_PORT_ALLOC_EC, 203
UDP_LOG_PORT_ENTRY_EC, 203
UDP_LOG_PORT_NOT_FOUND_EC, 205
UDP_LOG_PORT_THREAD_EC, 203
UDP_LOG_PROPERTY_FINALIZE_EC, 204
UDP_LOG_RECORD_EC, 204
UDP_LOG_RECV_ERROR_EC, 205
UDP_LOG_SEND_ERROR_EC, 205
UDP_LOG_SEND_PING_EC, 204
UDP_LOG_SET_LENGTH_EC, 204
UDP_LOG_SOCKET_ADDGRP_EC, 203
UDP_LOG_SOCKET_BIND_EC, 203
UDP_LOG_SOCKET_CLOSE_EC, 203
UDP_LOG_SOCKET_CREATE_EC, 202
UDP_LOG_SOCKET_IFFLAGS_EC, 203
UDP_LOG_SOCKET_IFLIST_EC, 203
UDP_LOG_SOCKET_INIT_EC, 202
UDP_LOG_SOCKET_REUSE_PORT_EC, 203
UDP_LOG_SOCKET_RXBUF_EC, 203
UDP_LOG_SOCKET_SET_MCAST_EC, 202
UDP_LOG_SOCKET_SET_MCASTIF_EC, 202
UDP_LOG_SOCKET_SETFD_EC, 202
UDP_LOG_SOCKET_SNDBUF_EC, 202
UDP_LOG_SOCKET_TTL_EC, 202
UDP_LOG_TRANSPORT_NO_INTERFACES_EC, 205
UDP_LOG_UDP_CREATE_INDEX_EC, 204
UDP_LOG_UDP_CREATE_TABLE_EC, 204
UDP_LOG_WINSOCK_INCOMPATIBLE_EC, 205
UDP_LOG_WSA_STARTUP_EC, 205
NETIO API, 132
NETIO Address API, 355
NETIO_ADDRESS_KIND_UDPv4, 355
NETIO_ADDRESS_KIND_UDPv6, 355
NETIO_ADDRESS_RTPS_RESERVED_HIGH, 356
NETIO_ADDRESS_RTPS_RESERVED_LOW, 355
NETIO_Address_INITIALIZER, 356
NETIO_Address_set_ipv4, 356
NETIO_Address_set_multicast, 356
NETIO_Netmask_INITIALIZER, 356
NETIO DGRAM API, 137
NETIO_ADDRESS_UDPv4_ANY_MULTICAST_ROUTE, 144
NETIO_ADDRESS_UDPv4_ANY_UNICAST_ROUTE, 143
NETIO_ADDRESS_UDPv6_ANY_MULTICAST_ROUTE, 144
NETIO_ADDRESS_UDPv6_ANY_UNICAST_ROUTE, 144
NETIO_DGRAM_INTERFACE_MAX_ADDRESSES, 139
NETIO_DGRAM_INTERFACE_MAX_INTERFACES, 139
NETIO_DGRAM_INTERFACE_SHARED_PORT_FLAG, 139
NETIO_DGRAM_Interface_add_route_entry_to_seq, 143
NETIO_DGRAM_Interface_add_route_to_seq, 142
NETIO_DGRAM_Interface_createFunc, 141
NETIO_DGRAM_Interface_resolve_ipv4_address, 143
NETIO_DGRAM_Interface_upstream_receive, 142
NETIO_DGRAM_InterfaceFactory_register, 141
NETIO_DGRAM_InterfaceFactory_unregister, 141
NETIO_DGRAM_InterfaceI, 141
NETIO_DGRAM_InterfaceMultiCastGroup_INITIALIZER, 139
NETIO_DGRAM_InterfaceMultiCastGroup_Invalid, 139
NETIO_DGRAM_InterfaceMultiCastGroup_UDPv4, 139
NETIO_DGRAM_InterfaceMultiCastGroup_UDPv6, 140
NETIO_DGRAM_InterfaceTableEntry_INITIALIZER, 140
NETIO_DGRAM_InterfaceTableEntrySeq_INITIALIZER, 140
NETIO_DGRAMINTERFACETABLEENTRY_UDPV4, 140
NETIO_DGRAMUserInterface_bind_addressFunc, 141
NETIO_DGRAMUserInterface_get_interface_listFunc, 141
NETIO_DGRAMUserInterface_resolve_addressFunc, 141
NETIO Interface API, 192
get_route_table, 194
NETIO_Interface_T, 193
NETIO_Interface_get_route_tableFunc, 194
NETIO_Interface_receive, 192
NETIO_Interface_receiveFunc, 193
NETIO_Interface_resolve_addressFunc, 193
NETIO_Interface_sendFunc, 193
resolve_address, 194
send, 194
NETIO Packet API, 190
NETIO_Packet_T, 190
NETIO_Packet_get_head, 190
NETIO_Packet_get_payload_length, 191
NETIO_Packet_set_payload, 190
NETIO_ADDRESS_KIND_UDPv4
NETIO Address API, 355
NETIO_ADDRESS_KIND_UDPv6
NETIO Address API, 355
NETIO_ADDRESS_RTPS_RESERVED_HIGH
NETIO Address API, 356

NETIO_ADDRESS_RTPS_RESERVED_LOW
NETIO Address API, 355
NETIO_ADDRESS_UDPv4_ANY_MULTICAST_ROUTE
NETIO DGRAM API, 144
NETIO_ADDRESS_UDPv4_ANY_UNICAST_ROUTE
NETIO DGRAM API, 143
NETIO_ADDRESS_UDPv6_ANY_MULTICAST_ROUTE
NETIO DGRAM API, 144
NETIO_ADDRESS_UDPv6_ANY_UNICAST_ROUTE
NETIO DGRAM API, 144
NETIO_Address, 448
kind, 448
port, 448
value, 448
NETIO_Address_INITIALIZER
NETIO Address API, 356
NETIO_Address_set_ipv4
NETIO Address API, 356
NETIO_Address_set_multicast
NETIO Address API, 356
NETIO_AddressSeq, 448
NETIO_AddressUInt32, 448
value, 449
NETIO_AddressValue, 449
address, 449
as_uint32, 449
NETIO_DEFAULT_INTRA_NAME
netio_interface.h, 512
NETIO_DEFAULT_NOTIF_NAME
Zero Copy v2 API, 146
NETIO_DEFAULT_RTPS_NAME
netio_interface.h, 512
NETIO_DEFAULT_UDP_NAME
netio_interface.h, 512
NETIO_DGRAM_INTERFACE_MAX_ADDRESSES
NETIO DGRAM API, 139
NETIO_DGRAM_INTERFACE_MAX_INTERFACES
NETIO DGRAM API, 139
NETIO_DGRAM_INTERFACE_SHARED_PORT_FLAG
NETIO DGRAM API, 139
NETIO_DGRAM_Interface_add_route_entry_to_seq
NETIO DGRAM API, 143
NETIO_DGRAM_Interface_add_route_to_seq
NETIO DGRAM API, 142
NETIO_DGRAM_Interface_createFunc
NETIO DGRAM API, 141
NETIO_DGRAM_Interface_resolve_ipv4_address
NETIO DGRAM API, 143
NETIO_DGRAM_Interface_upstream_receive
NETIO DGRAM API, 142
NETIO_DGRAM_InterfaceFactory_register
NETIO DGRAM API, 141
NETIO_DGRAM_InterfaceFactory_unregister
NETIO DGRAM API, 141
NETIO_DGRAM_InterfaceI, 449
bind_address, 450
create_instance, 450
get_interface_list, 450
NETIO DGRAM API, 141
release_address, 450
NETIO_DGRAM_InterfaceMultiCastGroup, 451
address_high, 451
address_low, 451
netmask, 451
NETIO_DGRAM_InterfaceMultiCastGroup_INITIALIZER
NETIO DGRAM API, 139
NETIO_DGRAM_InterfaceMultiCastGroup_Invalid
NETIO DGRAM API, 139
NETIO_DGRAM_InterfaceMultiCastGroup_UDPv4
NETIO DGRAM API, 139
NETIO_DGRAM_InterfaceMultiCastGroup_UDPv6
NETIO DGRAM API, 140
NETIO_DGRAM_InterfaceRouteEntry, 451
address, 452
netmask, 452
NETIO_DGRAM_InterfaceTableEntry, 452
address, 452
flags, 452
ifname, 453
mtu, 452
NETIO_DGRAM_InterfaceTableEntry_INITIALIZER
NETIO DGRAM API, 140
NETIO_DGRAM_InterfaceTableEntrySeq, 453
NETIO_DGRAM_InterfaceTableEntrySeq_INITIALIZER
NETIO DGRAM API, 140
NETIO_DGRAMINTERFACETABLEENTRY_UDPV4
NETIO DGRAM API, 140
NETIO_DGRAMUserInterface_bind_addressFunc
NETIO DGRAM API, 141
NETIO_DGRAMUserInterface_get_interface_listFunc
NETIO DGRAM API, 141
NETIO_DGRAMUserInterface_resolve_addressFunc
NETIO DGRAM API, 141
NETIO_Interface, 453
NETIO_Interface_T
NETIO Interface API, 193
NETIO_Interface_get_route_tableFunc
NETIO Interface API, 194
NETIO_Interface_receive
NETIO Interface API, 192
NETIO_Interface_receiveFunc
NETIO Interface API, 193
NETIO_Interface_release_addressFunc
netio_interface.h, 512
NETIO_Interface_reserve_addressFunc
netio_interface.h, 512
NETIO_Interface_resolve_addressFunc
NETIO Interface API, 193
NETIO_Interface_sendFunc
NETIO Interface API, 193
NETIO_InterfaceI, 453
NETIO_LOG_AR_ADD_INVALID_INTERFACE_EC

NETIO, 200
NETIO_LOG_AR_ADDRESS_RESOLVE_FAILED_EC
NETIO, 200
NETIO_LOG_AR_ADDRESS_STRING_EXCEEDED_-EC
NETIO, 201
NETIO_LOG_AR_ALLOC_FAILED_EC
NETIO, 200
NETIO_LOG_AR_CONTEXT_UNINITIALIZED_EC
NETIO, 200
NETIO_LOG_AR_INVALID_TOKEN_EC
NETIO, 200
NETIO_LOG_AR_PORT_RESOLVE_FAILED_EC
NETIO, 200
NETIO_LOG_BIND_CREATE_RTABLE_EC
NETIO, 199
NETIO_LOG_BIND_DELETE_RTABLE_EC
NETIO, 199
NETIO_LOG_BIND_EXTERNAL_FAILED_EC
NETIO, 200
NETIO_LOG_BIND_NEW_EC
NETIO, 199
NETIO_LOG_BIND_NEW_RTABLE_EC
NETIO, 199
NETIO_LOG_BIND_SET_LENGTH_EC
NETIO, 200
NETIO_LOG_BIND_SET_MAX_EC
NETIO, 200
NETIO_LOG_GETHOST_BYNAME_EC
NETIO, 199
NETIO_LOG_ILLEGAL_ROUTE_OPERATION_EC
NETIO, 201
NETIO_LOG_INVALID_ADDRESS_INDEX_EC
NETIO, 201
NETIO_LOG_LOOP_BINDX_DB_EC
NETIO, 201
NETIO_LOG_LOOP_CREATE_TABLE_EC
NETIO, 201
NETIO_LOG_LOOP_CURSOR_ERROR_EC
NETIO, 202
NETIO_LOG_LOOP_DELETE_TABLE_EC
NETIO, 201
NETIO_LOG_LOOP_FWD_EC
NETIO, 201
NETIO_LOG_LOOP_INITIALIZE_FAILED_EC
NETIO, 202
NETIO_LOG_LOOP_INVALID_COMPONENT_EC
NETIO, 202
NETIO_LOG_LOOP_INVALID_FACTORY_EC
NETIO, 202
NETIO_LOG_LOOP_INVALID_PROPERTY_EC
NETIO, 202
NETIO_LOG_LOOP_SELECT_TABLE_EC
NETIO, 201
NETIO_LOG_LOOP_SET_LENGTH_EC
NETIO, 202
NETIO_LOG_LOOP_UNBINDX_DB_EC
NETIO, 201
NETIO_LOG_PACKET_INITIALIZE_EC
NETIO, 201
NETIO_LOG_PACKET_SET_HEAD_EC
NETIO, 201
NETIO_LOG_PACKET_SET_TAIL_EC
NETIO, 201
NETIO_LOG_ROUTE_RTABLE_ADD_EC
NETIO, 200
NETIO_LOG_ROUTE_RTABLE_ALLOC_EC
NETIO, 200
NETIO_LOG_ROUTE_RTABLE_CREATE_EC
NETIO, 200
NETIO_LOG_ROUTE_RTABLE_DELETE_EC
NETIO, 200
NETIO_LOG_ROUTE_RTABLE_UPDATE_EC
NETIO, 200
NETIO_LOG_SET_NAME_EC
NETIO, 201
NETIO_LOG_UNBIND_EXTERNAL_FAILED_EC
NETIO, 200
NETIO_Netmask, 453
bits, 454
mask, 454
NETIO_Netmask_INITIALIZER
NETIO Address API, 356
NETIO_NetmaskSeq, 454
NETIO_Packet, 454
NETIO_Packet_T
NETIO Packet API, 190
NETIO_Packet_get_head
NETIO Packet API, 190
NETIO_Packet_get_payload_length
NETIO Packet API, 191
NETIO_Packet_set_payload
NETIO Packet API, 190
nack_period
DDS_RtpsReliableReaderProtocol_t, 420
name
DDS_DiscoveryComponent, 375
DDS_EntityNameQosPolicy, 386
nanosec
DDS_Duration_t, 384
DDS_Time_t, 435
OSAPI_SystemTime, 457
netio_common.h, 510
netio_config.h, 510
netio_dll.h, 510
netio_interface.h, 510
NETIO_DEFAULT_INTRA_NAME, 512
NETIO_DEFAULT_RTPS_NAME, 512
NETIO_DEFAULT_UDP_NAME, 512
NETIO_Interface_release_addressFunc, 512
NETIO_Interface_reserve_addressFunc, 512
netio_log.h, 512
netio_rtps.h, 516
netio_zcopy.h, 516
netio_zcopy_data.h, 517
netio_zcopy_dll.h, 517

netio_zcopy_log.h, 517
netio_zcopy_notif_interface.h, 517
netio_zcopy_notif_mechanism.h, 518
netio_zcopy_publication.h, 519
netio_zcopy_subscription.h, 519
netio_zcopy_tdatareader_gen.h, 519
netio_zcopy_tdatawriter_gen.h, 519
netio_zcopy_whsq.h, 519
    WHSQ_HistoryFactory_get_interface, 520
netio_zcopy_whsq_log.h, 520
    WHSQ_LOG_HISTORY_OBJECT, 520
netmask
    NETIO_DGRAM_InterfaceMultiCastGroup, 451
    NETIO_DGRAM_InterfaceRouteEntry, 452
    UDP_InterfaceTableEntry, 462
not_alive_count
    DDS_LivelinessChangedStatus, 392
not_alive_count_change
    DDS_LivelinessChangedStatus, 392
Notification Interface, 346
    ZCOPY_NotifInterface_receive, 346
    ZCOPY_NotifInterfaceFactory_register, 347
    ZCOPY_NotifInterfaceFactory_unregister, 347
    ZCOPY_NotifInterfaceFactoryProperty_INITIALIZ-
        ER, 346
notify_recv_port
    ZCOPY_NotifUserInterfaceI, 465

OS API, 147
OSAPI Heap, 155
    OSAPI_ALIGNMENT_DEFAULT, 155
    OSAPI_Alignment_T, 155
    OSAPI_Heap_allocate_buffer, 155
OSAPI Memory API, 169
    OSAPI_Memory_compare, 170
    OSAPI_Memory_copy, 169
    OSAPI_Memory_fndchr, 170
    OSAPI_Memory_move, 170
    OSAPI_Memory_zero, 169
OSAPI Mutex, 156
    OSAPI_Mutex_T, 156
    OSAPI_Mutex_give, 156
    OSAPI_Mutex_new, 156
    OSAPI_Mutex_take, 156
OSAPI Process, 158
    OSAPI_Process_getpid, 158
    OSAPI_ProcessId, 158
OSAPI Semaphore, 159
    OSAPI_Semaphore_T, 159
    OSAPI_Semaphore_give, 160
    OSAPI_Semaphore_new, 159
    OSAPI_Semaphore_take, 159
OSAPI String API, 171
    OSAPI_String_cmp, 171
    OSAPI_String_length, 171
    OSAPI_String_ncmp, 171
OSAPI System API, 173
    OSAPI_System_clock_tick, 177
    OSAPI_System_clock_tick_from_time, 178
    OSAPI_System_generate_uuid_T, 175
    OSAPI_System_get_hostname_T, 175
    OSAPI_System_get_native_interface, 177
    OSAPI_System_get_property, 176
    OSAPI_System_get_ticktime_T, 176
    OSAPI_System_get_time_T, 175
    OSAPI_System_get_timer_resolution_T, 174
    OSAPI_System_initialize_T, 175
    OSAPI_System_set_interface, 176
    OSAPI_System_set_property, 177
    OSAPI_System_start_timer_T, 174
    OSAPI_SystemI, 176
    OSAPI_SystemI_INITIALIZER, 174
OSAPI Thread API, 179
    OSAPI_THREAD_DEFAULT_OPTIONS, 180
    OSAPI_THREAD_FLOATING_POINT, 180
    OSAPI_THREAD_PRIORITY_ABOVE_NORMAL,
        180
    OSAPI_THREAD_PRIORITY_BELOW_NORMAL,
        179
    OSAPI_THREAD_PRIORITY_HIGH, 180
    OSAPI_THREAD_PRIORITY_INHERIT, 180
    OSAPI_THREAD_PRIORITY_LOW, 179
    OSAPI_THREAD_PRIORITY_NORMAL, 180
    OSAPI_THREAD_PROPERTY_DEFAULT, 180
    OSAPI_THREAD_REALTIME_PRIORITY, 180
    OSAPI_THREAD_STDIO, 180
    OSAPI_THREAD_USE_OSDEFAULT_STACKSI-
        ZE, 180
    OSAPI_ThreadOptions, 180
    OSAPI_ThreadProperty_INITIALIZER, 180
OSAPI Types, 181
    RTI_BOOL, 182
    RTI_FALSE, 182
    RTI_INT16, 181
    RTI_INT32, 181
    RTI_INT64, 182
    RTI_INT8, 181
    RTI_SIZE_T, 182
    RTI_TRUE, 182
    RTI_UINT16, 181
    RTI_UINT32, 181
    RTI_UINT64, 182
    RTI_UINT8, 181
OSAPI_ALIGNMENT_DEFAULT
    OSAPI Heap, 155
OSAPI_Alignment_T
    OSAPI Heap, 155
OSAPI_Heap_allocate_buffer
    OSAPI Heap, 155
OSAPI_LOG_VERBOSITY_DEBUG
    Log API, 149
OSAPI_LOG_VERBOSITY_ERROR
    Log API, 149
OSAPI_LOG_VERBOSITY_SILENT
    Log API, 149
OSAPI_LOG_VERBOSITY_WARNING
    Log API, 149

OSAPI_Log_clear
Log API, 151
OSAPI_Log_get_display_handler
Log API, 151
OSAPI_Log_get_last_error_code
Log API, 152
OSAPI_Log_get_log_handler
Log API, 150
OSAPI_Log_get_property
Log API, 152
OSAPI_Log_get_trace_handler
Log API, 150
OSAPI_Log_set_display_handler
Log API, 151
OSAPI_Log_set_log_handler
Log API, 149
OSAPI_Log_set_property
Log API, 152
OSAPI_Log_set_trace_handler
Log API, 150
OSAPI_Log_set_verbosity
Log API, 151
OSAPI_Log_write_buffer_T
Log API, 149
OSAPI_LogHandler_T
Log API, 149
OSAPI_LogProperty, 454
log_detail, 455
max_buffer_size, 455
write_buffer, 455
OSAPI_LogVerbosity_T
Log API, 149
OSAPI_Memory_compare
OSAPI Memory API, 170
OSAPI_Memory_copy
OSAPI Memory API, 169
OSAPI_Memory_fndchr
OSAPI Memory API, 170
OSAPI_Memory_move
OSAPI Memory API, 170
OSAPI_Memory_zero
OSAPI Memory API, 169
OSAPI_Mutex_T
OSAPI Mutex, 156
OSAPI_Mutex_give
OSAPI Mutex, 156
OSAPI_Mutex_new
OSAPI Mutex, 156
OSAPI_Mutex_take
OSAPI Mutex, 156
OSAPI_Process_getpid
OSAPI Process, 158
OSAPI_ProcessId
OSAPI Process, 158
OSAPI_SHMEM_ATTACH_STATUS
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_ATTACH_STATUS_OK
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_ATTACH_STATUS_SEGMENT_NOT_FOUND
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_MODE
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_MODE_LOCKABLE
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_MODE_READ
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_MODE_ROBUST
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_MODE_UNKNOWN
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_MODE_WRITE
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_STATUS
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_STATUS_OK
OSAPI_SharedMemorySegment, 165
OSAPI_SHMEM_STATUS_OWNER_DEAD
OSAPI_SharedMemorySegment, 165
OSAPI_Semaphore_T
OSAPI Semaphore, 159
OSAPI_Semaphore_give
OSAPI Semaphore, 160
OSAPI_Semaphore_new
OSAPI Semaphore, 159
OSAPI_Semaphore_take
OSAPI Semaphore, 159
OSAPI_SharedMemoryMonitor, 161
OSAPI_SharedMemoryMonitor_acquire, 161
OSAPI_SharedMemoryMonitor_get_size, 161
OSAPI_SharedMemoryMonitor_initialize, 161
OSAPI_SharedMemoryMonitor_mark_consistent, 163
OSAPI_SharedMemoryMonitor_release, 162
OSAPI_SharedMemoryMonitor_signal, 162
OSAPI_SharedMemoryMonitor_wait, 162
OSAPI_SharedMemoryMonitor_acquire
OSAPI_SharedMemoryMonitor, 161
OSAPI_SharedMemoryMonitor_get_size
OSAPI_SharedMemoryMonitor, 161
OSAPI_SharedMemoryMonitor_initialize
OSAPI_SharedMemoryMonitor, 161
OSAPI_SharedMemoryMonitor_mark_consistent
OSAPI_SharedMemoryMonitor, 163
OSAPI_SharedMemoryMonitor_release
OSAPI_SharedMemoryMonitor, 162
OSAPI_SharedMemoryMonitor_signal
OSAPI_SharedMemoryMonitor, 162
OSAPI_SharedMemoryMonitor_wait
OSAPI_SharedMemoryMonitor, 162
OSAPI_SharedMemorySegment, 164
OSAPI_SHMEM_ATTACH_STATUS, 165
OSAPI_SHMEM_ATTACH_STATUS_OK, 165
OSAPI_SHMEM_ATTACH_STATUS_SEGMENT_NOT_FOUND, 165
OSAPI_SHMEM_MODE, 165

OSAPI_SHMEM_MODE_LOCKABLE, 165
OSAPI_SHMEM_MODE_READ, 165
OSAPI_SHMEM_MODE_ROBUST, 165
OSAPI_SHMEM_MODE_UNKNOWN, 165
OSAPI_SHMEM_MODE_WRITE, 165
OSAPI_SHMEM_STATUS, 165
OSAPI_SHMEM_STATUS_OK, 165
OSAPI_SHMEM_STATUS_OWNER_DEAD, 165
OSAPI_SharedMemorySegment_attach_impl, 167
OSAPI_SharedMemorySegment_create_impl, 166
OSAPI_SharedMemorySegment_delete_impl, 167
OSAPI_SharedMemorySegment_detach_impl, 167
OSAPI_SharedMemorySegment_get_max_size, 165
OSAPI_SharedMemorySegment_is_owner_alive, 165
OSAPI_SharedMemorySegment_lock_impl, 168
OSAPI_SharedMemorySegment_mark_consistent_impl, 168
OSAPI_SharedMemorySegment_unlock_impl, 168
OSAPI_SharedMemorySegmentHandle_get_size, 166
OSAPI_SharedMemorySegmentHeader_get_size, 166
OSAPI_SharedMemorySegment_attach_impl
OSAPI_SharedMemorySegment, 167
OSAPI_SharedMemorySegment_create_impl
OSAPI_SharedMemorySegment, 166
OSAPI_SharedMemorySegment_delete_impl
OSAPI_SharedMemorySegment, 167
OSAPI_SharedMemorySegment_detach_impl
OSAPI_SharedMemorySegment, 167
OSAPI_SharedMemorySegment_get_max_size
OSAPI_SharedMemorySegment, 165
OSAPI_SharedMemorySegment_is_owner_alive
OSAPI_SharedMemorySegment, 165
OSAPI_SharedMemorySegment_lock_impl
OSAPI_SharedMemorySegment, 168
OSAPI_SharedMemorySegment_mark_consistent_impl
OSAPI_SharedMemorySegment, 168
OSAPI_SharedMemorySegment_unlock_impl
OSAPI_SharedMemorySegment, 168
OSAPI_SharedMemorySegmentHandle, 455
mode, 455
ptr_header, 455
ptr_user_data, 455
OSAPI_SharedMemorySegmentHandle_get_size
OSAPI_SharedMemorySegment, 166
OSAPI_SharedMemorySegmentHeader, 455
mode, 456
size, 456
OSAPI_SharedMemorySegmentHeader_get_size
OSAPI_SharedMemorySegment, 166
OSAPI_String_cmp
OSAPI String API, 171
OSAPI_String_length
OSAPI String API, 171
OSAPI_String_ncmp
OSAPI String API, 171
OSAPI_System_clock_tick
OSAPI System API, 177
OSAPI_System_clock_tick_from_time
OSAPI System API, 178
OSAPI_System_generate_uuid_T
OSAPI System API, 175
OSAPI_System_get_hostname_T
OSAPI System API, 175
OSAPI_System_get_native_interface
OSAPI System API, 177
OSAPI_System_get_property
OSAPI System API, 176
OSAPI_System_get_ticktime_T
OSAPI System API, 176
OSAPI_System_get_time_T
OSAPI System API, 175
OSAPI_System_get_timer_resolution_T
OSAPI System API, 174
OSAPI_System_initialize_T
OSAPI System API, 175
OSAPI_System_set_interface
OSAPI System API, 176
OSAPI_System_set_property
OSAPI System API, 177
OSAPI_System_start_timer_T
OSAPI System API, 174
OSAPI_SystemI, 456
generate_uuid, 456
get_hostname, 456
get_ticktime, 457
get_time, 456
get_timer_resolution, 456
OSAPI System API, 176
start_timer, 456
OSAPI_SystemI_INITIALIZER
OSAPI System API, 174
OSAPI_SystemProperty, 457
hostname, 457
OSAPI_SystemTime, 457
nanosec, 457
osapi_time.h, 524
sec, 457
OSAPI_THREAD_DEFAULT_OPTIONS
OSAPI Thread API, 180
OSAPI_THREAD_FLOATING_POINT
OSAPI Thread API, 180
OSAPI_THREAD_PRIORITY_ABOVE_NORMAL
OSAPI Thread API, 180
OSAPI_THREAD_PRIORITY_BELOW_NORMAL
OSAPI Thread API, 179
OSAPI_THREAD_PRIORITY_HIGH
OSAPI Thread API, 180
OSAPI_THREAD_PRIORITY_INHERIT
OSAPI Thread API, 180
OSAPI_THREAD_PRIORITY_LOW
OSAPI Thread API, 179

OSAPI_THREAD_PRIORITY_NORMAL
OSAPI Thread API, 180
OSAPI_THREAD_PROPERTY_DEFAULT
OSAPI Thread API, 180
OSAPI_THREAD_REALTIME_PRIORITY
OSAPI Thread API, 180
OSAPI_THREAD_STDIO
OSAPI Thread API, 180
OSAPI_THREAD_USE_OSDEFAULT_STACKSIZE
OSAPI Thread API, 180
OSAPI_ThreadOptions
OSAPI Thread API, 180
OSAPI_ThreadProperty, 458
options, 458
priority, 458
stack_size, 458
OSAPI_ThreadProperty_INITIALIZER
OSAPI Thread API, 180
OSAPI_TimerProperty, 458
OSAPI_Trace_set_trace_mask
Log API, 151
OWNERSHIP, 251
DDS_EXCLUSIVE_OWNERSHIP_QOS, 251
DDS_OWNERSHIP_QOS_POLICY_NAME, 251
DDS_OwnershipQosPolicyKind, 251
DDS_SHARED_OWNERSHIP_QOS, 251
OWNERSHIP_STRENGTH, 252
DDS_OWNERSHIPSTRENGTH_QOS_POLICY_-NAME, 252
on_before_sample_commit
DDS_DataReaderListener, 362
on_before_sample_deserialize
DDS_DataReaderListener, 362
on_data_available
DDS_DataReaderListener, 362
on_data_on_readers
DDS_SubscriberListener, 430
on_inconsistent_topic
DDS_TopicListener, 436
on_instance_replaced
DDS_DataReaderListener, 363
on_liveliness_changed
DDS_DataReaderListener, 362
on_liveliness_lost
DDS_DataWriterListener, 369
on_offered_deadline_missed
DDS_DataWriterListener, 368
on_offered_incompatible_qos
DDS_DataWriterListener, 369
on_publication_matched
DDS_DataWriterListener, 369
on_reliable_reader_activity_changed
DDS_DataWriterListener, 369
on_reliable_sample_unacknowledged
DDS_DataWriterListener, 369
on_requested_deadline_missed
DDS_DataReaderListener, 362
on_requested_incompatible_qos
DDS_DataReaderListener, 362
on_sample_lost
DDS_DataReaderListener, 362
on_sample_rejected
DDS_DataReaderListener, 362
on_subscription_matched
DDS_DataReaderListener, 362
options
OSAPI_ThreadProperty, 458
ordered_access
DDS_PresentationQosPolicy, 406
osapi_cc.h, 520
RTI_PRIVATE, 521
osapi_cc_gcc.h, 521
osapi_cc_qcc.h, 521
osapi_cc_stdc.h, 521
osapi_config.h, 521
osapi_heap_impl.h, 522
osapi_log_impl.h, 522
osapi_os.h, 522
osapi_os_def.h, 522
osapi_os_posix.h, 522
osapi_process.h, 522
osapi_system.h, 523
osapi_time.h, 524
OSAPI_SystemTime, 524
ownership
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_PublicationBuiltinTopicData, 410
DDS_SubscriptionBuiltinTopicData, 432
ownership_strength
DDS_DataWriterQos, 372
DDS_PublicationBuiltinTopicData, 410

PRESENTATION, 259
DDS_GROUP_PRESENTATION_QOS, 259
DDS_HIGHEST_OFFERED_PRESENTATION_Q-OS, 259
DDS_INSTANCE_PRESENTATION_QOS, 259
DDS_PresentationQosPolicyAccessScopeKind, 259
DDS_TOPIC_PRESENTATION_QOS, 259
Participant Built-in Topic, 225
DDS_PARTICIPANT_TOPIC_NAME, 225
DDS_ParticipantBuiltinTopicData_copy, 226
DDS_ParticipantBuiltinTopicData_initialize, 225
DDS_ParticipantBuiltinTopicData_is_equal, 226
participant_id
DDS_WireProtocolQosPolicy, 439
participant_id_gain
DDS_RtpsWellKnownPorts_t, 424
participant_key
DDS_PublicationBuiltinTopicData, 409
DDS_SubscriptionBuiltinTopicData, 432
participant_liveliness_assert_period
DPSE_DiscoveryPluginProperty, 442
participant_liveliness_lease_duration
DPSE_DiscoveryPluginProperty, 442

participant_name
DDS_DomainParticipantQos, 380
DDS_ParticipantBuiltinTopicData, 403
period
DDS_DeadlineQosPolicy, 374
PluginHelper_get_key_kind
User-Defined Types, 112
PluginHelper_instance_to_keyhash
User-Defined Types, 112
policies
DDS_OfferedIncompatibleQosStatus, 398
DDS_RequestedIncompatibleQosStatus, 416
port
DDS_Locator_t, 396
NETIO_Address, 448
port_base
DDS_RtpsWellKnownPorts_t, 423
priority
OSAPI_ThreadProperty, 458
product_version
DDS_ParticipantBuiltinTopicData, 403
protocol
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_DomainParticipantQos, 380
ptr_header
OSAPI_SharedMemorySegmentHandle, 455
ptr_user_data
OSAPI_SharedMemorySegmentHandle, 455
Publication Built-in Topic, 227
DDS_PUBLICATION_TOPIC_NAME, 227
DDS_PublicationBuiltinTopicData_copy, 228
DDS_PublicationBuiltinTopicData_initialize, 227
DDS_PublicationBuiltinTopicData_is_equal, 228
Publication Module, 46
publication_handle
DDS_DataReaderInstanceReplacedStatus, 360
DDS_SampleInfo, 427
publication_sequence_number
DDS_SampleInfo, 427
Publisher, 47
DDS_DATAWRITER_QOS_DEFAULT, 51
DDS_Publisher, 48
DDS_Publisher_as_entity, 50
DDS_Publisher_create_datawriter, 50
DDS_Publisher_get_participant, 51
DDS_Publisher_lookup_datawriter, 50
DDS_PublisherListener_INITIALIZER, 48
DDS_PublisherQos_INITIALIZER, 48
DDS_PublisherQos_copy, 49
DDS_PublisherQos_initialize, 48
DDS_PublisherQos_is_equal, 49

QoS Policies, 246
DDS_DEADLINE_QOS_POLICY_ID, 249
DDS_ENTITYFACTORY_QOS_POLICY_ID, 249
DDS_HISTORY_QOS_POLICY_ID, 249
DDS_INVALID_QOS_POLICY_ID, 249
DDS_LIVELINESS_QOS_POLICY_ID, 249
DDS_OWNERSHIP_QOS_POLICY_ID, 249
DDS_OWNERSHIPSTRENGTH_QOS_POLICY_ID, 249
DDS_PRESENTATION_QOS_POLICY_ID, 249
DDS_QosPolicyId_t, 249
DDS_RELIABILITY_QOS_POLICY_ID, 249

REDA, 153
REDA_LOG_BUFFERPOOL_BUFFER_INITIALIZATION_FAILED_EC, 153
REDA_LOG_BUFFERPOOL_DOUBLE_FREE_EC, 154
REDA_LOG_BUFFERPOOL_NOT_EMPTY_EC, 153
REDA_LOG_BUFFERPOOL_OUT_OF_RESOURCES_EC, 153
REDA_LOG_INDEX_ENTRY_EXISTS_EC, 154
REDA_LOG_INDEX_FULL_EC, 154
REDA_LOG_INVALID_LIST_NODE_EC, 154
REDA_LOG_SEQUENCE_ALLOC_FAILED_EC, 154
REDA_LOG_SEQUENCE_COPY_FAILED_EC, 154
REDA_LOG_SEQUENCE_INDEX_OUT_OF_BOUNDS_EC, 154
REDA_LOG_SEQUENCE_INVALID_LENGTH_EC, 154
REDA_LOG_SEQUENCE_INVALID_OPERATION_EC, 154
REDA_LOG_SEQUENCE_REALLOCATION_EC, 154
REDA_LOG_SEQUENCE_SET_MAX_FAILED_EC, 154
REDA_LOG_STRING_ALLOC_FAILED_EC, 154
REDA_LOG_BUFFERPOOL_BUFFER_INITIALIZATION_FAILED_EC
REDA, 153
REDA_LOG_BUFFERPOOL_DOUBLE_FREE_EC
REDA, 154
REDA_LOG_BUFFERPOOL_NOT_EMPTY_EC
REDA, 153
REDA_LOG_BUFFERPOOL_OUT_OF_RESOURCES_EC
REDA, 153
REDA_LOG_INDEX_ENTRY_EXISTS_EC
REDA, 154
REDA_LOG_INDEX_FULL_EC
REDA, 154
REDA_LOG_INVALID_LIST_NODE_EC
REDA, 154
REDA_LOG_SEQUENCE_ALLOC_FAILED_EC
REDA, 154
REDA_LOG_SEQUENCE_COPY_FAILED_EC
REDA, 154
REDA_LOG_SEQUENCE_INDEX_OUT_OF_BOUNDS_EC
REDA, 154
REDA_LOG_SEQUENCE_INVALID_LENGTH_EC
REDA, 154

REDA_LOG_SEQUENCE_INVALID_OPERATION_EC
REDA, 154
REDA_LOG_SEQUENCE_REALLOCATION_EC
REDA, 154
REDA_LOG_SEQUENCE_SET_MAX_FAILED_EC
REDA, 154
REDA_LOG_STRING_ALLOC_FAILED_EC
REDA, 154
RELIABILITY, 255
DDS_BEST_EFFORT_RELIABILITY_QOS, 255
DDS_RELIABILITY_QOS_POLICY_NAME, 255
DDS_RELIABLE_RELIABILITY_QOS, 255
DDS_ReliabilityQosPolicyKind, 255
RESOURCE_LIMITS, 258
DDS_LENGTH_AUTO, 258
DDS_LENGTH_UNLIMITED, 258
DDS_RESOURCELIMITS_QOS_POLICY_NAME, 258
RH, 333
RHSM_LOG_DESTINATION_BY_SOURCE_NOT_SUPPORTED_EC, 335
RHSM_LOG_ENTRY_RESERVATION_FAILED_EC, 335
RHSM_LOG_FAILED_TO_REMOVE_OLDEST_EC, 335
RHSM_LOG_GET_RECEPTION_TIMESTAMP_EC, 334
RHSM_LOG_GET_TIME_EC, 335
RHSM_LOG_HISTORY_UPDATE_EC, 335
RHSM_LOG_INFO_ARRAY_EC, 334
RHSM_LOG_INVALID_INSTANCE_REPLACEMENT_EC, 334
RHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC, 334
RHSM_LOG_MAX_SAMPLE_TOO_SMALL_EC, 334
RHSM_LOG_NO_PROPERTY_EC, 335
RHSM_LOG_NO_PROPERTY_QOS_EC, 334
RHSM_LOG_OBJECT_ALLOCATE_EC, 335
RHSM_LOG_OBJECT_DELETE_EC, 335
RHSM_LOG_OBJECT_INDEX_EC, 335
RHSM_LOG_OBJECT_INVALID_EC, 335
RHSM_LOG_OUTSTANDING_SAMPLE_EC, 334
RHSM_LOG_RECOMMIT_FAILURE_EC, 335
RHSM_LOG_RETURN_SAMPLE_EC, 335
RHSM_LOG_RW_PRUNE_FAILED_EC, 335
RHSM_LOG_SAMPLE_INFO_POOL_EC, 334
RHSM_LOG_SAMPLE_POOL_EC, 334
RHSM_LOG_SAMPLE_POOL_EMPTY_EC, 335
RHSM_LOG_SAMPLE_PTR_ARRAY_EC, 334
RHSM_LOG_TBF_NOT_SUPPORTED_EC, 334
RHSM Reader History API, 130
RHSM_HistoryFactory_get_interface, 130
RHSM_HistoryFactory_get_interface
RHSM Reader History API, 130
RHSM_LOG_DESTINATION_BY_SOURCE_NOT_SUPPORTED_EC
RH, 335
RHSM_LOG_ENTRY_RESERVATION_FAILED_EC
RH, 335
RHSM_LOG_FAILED_TO_REMOVE_OLDEST_EC
RH, 335
RHSM_LOG_GET_RECEPTION_TIMESTAMP_EC
RH, 334
RHSM_LOG_GET_TIME_EC
RH, 335
RHSM_LOG_HISTORY_OBJECT
rh_sm_log.h, 528
RHSM_LOG_HISTORY_UPDATE_EC
RH, 335
RHSM_LOG_INFO_ARRAY_EC
RH, 334
RHSM_LOG_INSTANCEHANDLE_OBJECT
rh_sm_log.h, 528
RHSM_LOG_INVALID_INSTANCE_REPLACEMENT_EC
RH, 334
RHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC
RH, 334
RHSM_LOG_KEY_OBJECT
rh_sm_log.h, 528
RHSM_LOG_KEYINDEXPOOL_OBJECT
rh_sm_log.h, 528
RHSM_LOG_KEYPOOL_OBJECT
rh_sm_log.h, 528
RHSM_LOG_MAX_SAMPLE_TOO_SMALL_EC
RH, 334
RHSM_LOG_NO_PROPERTY_EC
RH, 335
RHSM_LOG_NO_PROPERTY_QOS_EC
RH, 334
RHSM_LOG_OBJECT_ALLOCATE_EC
RH, 335
RHSM_LOG_OBJECT_DELETE_EC
RH, 335
RHSM_LOG_OBJECT_INDEX_EC
RH, 335
RHSM_LOG_OBJECT_INVALID_EC
RH, 335
RHSM_LOG_OUTSTANDING_SAMPLE_EC
RH, 334
RHSM_LOG_OWNER_OBJECT
rh_sm_log.h, 528
RHSM_LOG_RECOMMIT_FAILURE_EC
RH, 335
RHSM_LOG_REMOVELIST_OBJECT
rh_sm_log.h, 528
RHSM_LOG_RETURN_SAMPLE_EC
RH, 335
RHSM_LOG_RW_OBJECT
rh_sm_log.h, 528
RHSM_LOG_RW_PRUNE_FAILED_EC
RH, 335
RHSM_LOG_RWINDEX_OBJECT
rh_sm_log.h, 528

RHSM_LOG_RWPOOL_OBJECT
rh_sm_log.h, 528
RHSM_LOG_SAMPLE_INFO_POOL_EC
RH, 334
RHSM_LOG_SAMPLE_POOL_EC
RH, 334
RHSM_LOG_SAMPLE_POOL_EMPTY_EC
RH, 335
RHSM_LOG_SAMPLE_PTR_ARRAY_EC
RH, 334
RHSM_LOG_SAMPLEINFO_OBJECT
rh_sm_log.h, 528
RHSM_LOG_SAMPLEPOOL_OBJECT
rh_sm_log.h, 528
RHSM_LOG_SAMPLEPTPOOL_OBJECT
rh_sm_log.h, 528
RHSM_LOG_TBF_NOT_SUPPORTED_EC
RH, 334
RT, 188
RT_LOG_REGISTRY_EXISTS_EC, 188
RT_LOG_REGISTRY_FINALIZE_EC, 188
RT_LOG_REGISTRY_INCONSISTENT_CID_EC, 189
RT_LOG_REGISTRY_INIT_FACTORY_EC, 188
RT_LOG_REGISTRY_INIT_FAILURE_EC, 188
RT_LOG_REGISTRY_NAME_TOO_LONG_EC, 188
RT_LOG_REGISTRY_NOT_INITIALIZED_EC, 189
RT_LOG_REGISTRY_REGISTER_EC, 188
RT_LOG_SET_IMMUTABLE_PROPERTY_EC, 188
RT Run-Time API, 128
RT_ComponentFactoryId_T, 128
RT_ComponentFactoryId_set_name, 129
RT_Registry_register, 128
RT_Registry_unregister, 129
RT_ComponentFactoryId_T
RT Run-Time API, 128
RT_ComponentFactoryId_set_name
RT Run-Time API, 129
RT_LOG_REGISTRY_EXISTS_EC
RT, 188
RT_LOG_REGISTRY_FINALIZE_EC
RT, 188
RT_LOG_REGISTRY_INCONSISTENT_CID_EC
RT, 189
RT_LOG_REGISTRY_INIT_FACTORY_EC
RT, 188
RT_LOG_REGISTRY_INIT_FAILURE_EC
RT, 188
RT_LOG_REGISTRY_NAME_TOO_LONG_EC
RT, 188
RT_LOG_REGISTRY_NOT_INITIALIZED_EC
RT, 189
RT_LOG_REGISTRY_REGISTER_EC
RT, 188
RT_LOG_SET_IMMUTABLE_PROPERTY_EC
RT, 188
RT_Registry_register
RT Run-Time API, 128
RT_Registry_unregister
RT Run-Time API, 129
RTI_BOOL
OSAPI Types, 182
RTI_FALSE
OSAPI Types, 182
RTI_INT16
OSAPI Types, 181
RTI_INT32
OSAPI Types, 181
RTI_INT64
OSAPI Types, 182
RTI_INT8
OSAPI Types, 181
RTI_ManagementQosPolicy, 458
is_anonymous, 459
RTI_PRIVATE
osapi_cc.h, 521
RTI_SIZE_T
OSAPI Types, 182
RTI_TRUE
OSAPI Types, 182
RTI_UINT16
OSAPI Types, 181
RTI_UINT32
OSAPI Types, 181
RTI_UINT64
OSAPI Types, 182
RTI_UINT8
OSAPI Types, 181
RTPS, 206
RTPS_LOG_ACK_EC, 214
RTPS_LOG_ALLOCATE_EC, 211
RTPS_LOG_ASSERT_PEER_EC, 214
RTPS_LOG_ASSERT_TRANSPORT_EC, 214
RTPS_LOG_BAD_PARAMETER_EC, 213
RTPS_LOG_BITCOUNT_OUT_OF_BOUNDS_EC, 215
RTPS_LOG_BITMAP_SET_BIT_EC, 216
RTPS_LOG_BITMAP_SHIFT_EC, 218
RTPS_LOG_BUFFERPOOL_DELETE_EC, 217
RTPS_LOG_CHECKSUM_CHECKSUM_CALCULATION_FAILED_EC, 220
RTPS_LOG_CHECKSUM_CHECKSUM_DESERIALIZE_ERROR_EC, 220
RTPS_LOG_CHECKSUM_CHECKSUM_ERROR_EC, 220
RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM128_EC, 220
RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM32_EC, 220
RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM64_EC, 220
RTPS_LOG_CHECKSUM_INCONSISTENT_LENGTH_EC, 219

RTPS_LOG_CHECKSUM_INVALID_CRC32_FLAGS_EC, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM128_EC, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM256_EC, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM32_EC, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM64_EC, 220
RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN128_EC, 220
RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN32_EC, 220
RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN64_EC, 220
RTPS_LOG_CHECKSUM_LENGTH_DESERIALIZE_ERROR_EC, 219
RTPS_LOG_CHECKSUM_MISSING_CHECKSUM_LENGTH_EC, 219
RTPS_LOG_CHECKSUM_UNSUPPORTED_EC, 219
RTPS_LOG_CREATE_BIND_TABLE_EC, 217
RTPS_LOG_CREATE_BUFFER_POOL_EC, 218
RTPS_LOG_CREATE_EVENT_TIMER_EC, 215
RTPS_LOG_CREATE_HB_EVENT_EC, 215
RTPS_LOG_CREATE_PEERS_INDEX_EC, 218
RTPS_LOG_CREATE_ROUTE_TABLE_EC, 211
RTPS_LOG_DATA_ALREADY_RECEIVED_EC, 215
RTPS_LOG_DATA_OUT_OF_RANGE_EC, 215
RTPS_LOG_DB_CREATE_BIND_ENTRY_EC, 212
RTPS_LOG_DB_CREATE_BIND_PEER_INDEX_EC, 212
RTPS_LOG_DB_CREATE_DIRECT_INDEX_EC, 219
RTPS_LOG_DB_CREATE_EXT_BIND_ENTRY_EC, 216
RTPS_LOG_DB_CREATE_GROUP_INDEX_EC, 219
RTPS_LOG_DB_CREATE_ROUTE_ENTRY_EC, 212
RTPS_LOG_DB_CREATE_ROUTE_PEER_INDEX_EC, 211
RTPS_LOG_DB_CREATE_ROUTE_XPORT_INDEX_EC, 211
RTPS_LOG_DB_CREATE_SELECTED_GROUP_INDEX_EC, 219
RTPS_LOG_DB_DELETE_EXTERNAL_BIND_ENTRY_EC, 213
RTPS_LOG_DB_DELETE_ROUTE_ENTRY_EC, 212
RTPS_LOG_DB_INSERT_BIND_ENTRY_EC, 212
RTPS_LOG_DB_INSERT_ROUTE_ENTRY_EC, 212
RTPS_LOG_DB_LOCK_EC, 217
RTPS_LOG_DB_REMOVE_BIND_ENTRY_EC, 212
RTPS_LOG_DB_REMOVE_EXTERNAL_BIND_ENTRY_EC, 212
RTPS_LOG_DB_SELECT_ALL_EC, 212
RTPS_LOG_DB_SELECT_MATCH_EC, 212
RTPS_LOG_DB_SELECT_RANGE_EC, 212
RTPS_LOG_DB_UNLOCK_EC, 217
RTPS_LOG_DELETE_BUFFER_POOL_EC, 218
RTPS_LOG_DELETE_INDEX_EC, 216
RTPS_LOG_DELETE_INDEXER_EC, 219
RTPS_LOG_DELETE_TABLE_EC, 217
RTPS_LOG_DESERIALIZE_APP_ID_EC, 216
RTPS_LOG_DESERIALIZE_HOST_ID_EC, 216
RTPS_LOG_DESERIALIZE_INSTANCE_ID_EC, 216
RTPS_LOG_DESERIALIZE_OBJECT_ID_EC, 216
RTPS_LOG_DIRECT_SEND_EC, 217
RTPS_LOG_EXCEEDED_LIMIT_PEERS_EC, 213
RTPS_LOG_EXCEEDED_LIMIT_TRANSPORTS_EC, 213
RTPS_LOG_FINALIZE_INTERFACE_EC, 218
RTPS_LOG_FIND_EXTERNAL_INTERFACE_EC, 214
RTPS_LOG_FIRST_SN_GREATER_LAST_SN_EC, 215
RTPS_LOG_FORWARD_UPSTREAM_EC, 213
RTPS_LOG_FULL_SEND_WINDOW_EC, 213
RTPS_LOG_FULLY_ACKED_READER_EC, 214
RTPS_LOG_FUNC_UNSUPPORTED_EC, 213
RTPS_LOG_GET_PEER_BUFFER_EC, 219
RTPS_LOG_GET_WINDOW_BUFFER_EC, 217
RTPS_LOG_INCONSISTENT_HDR_EXT_LENGTH_ERROR_EC, 221
RTPS_LOG_INDEX_ADD_ENTRY_EC, 212
RTPS_LOG_INDEXER_REMOVE_ENTRY_EC, 218
RTPS_LOG_INITIALIZE_INTERFACE_EC, 211
RTPS_LOG_INSERT_WINDOW_FULL_EC, 217
RTPS_LOG_INTERFACE_MISMATCH_EC, 213
RTPS_LOG_INTERFACE_NOT_ENABLED_EC, 215
RTPS_LOG_INTF_MODE_UNDEF_EC, 218
RTPS_LOG_INVALID_HDR_EXT_PID_ERROR_EC, 221
RTPS_LOG_INVALID_HDR_EXT_PID_LIST_EC, 221
RTPS_LOG_INVALID_PACKET_EC, 215
RTPS_LOG_LOOKUP_EXT_BIND_ENTRY_EC, 218
RTPS_LOG_LOOKUP_PEER_EC, 218
RTPS_LOG_MESSAGE_EXCEED_LENGTH_ERROR_EC, 221
RTPS_LOG_NEGATIVE_RESERVATION_COUNT_EC, 219
RTPS_LOG_NETIO_PACKET_INSUFFICIENT_BUFFER_EC, 219

RTPS_LOG_NETIO_PACKET_SET_HEAD_EC, 214
RTPS_LOG_NETIO_PACKET_SET_TAIL_EC, 213
RTPS_LOG_NONEXISTENT_BIND_EC, 214
RTPS_LOG_NONEXISTENT_EXTERNAL_BIND_EC, 214
RTPS_LOG_NONEXISTENT_ROUTE_EC, 214
RTPS_LOG_NOT_ENABLED_EC, 213
RTPS_LOG_PACKET_INITIALIZE_EC, 217
RTPS_LOG_PROCESS_ACKNACK_EC, 215
RTPS_LOG_PROCESS_DATA_BATCH_EC, 219
RTPS_LOG_PROCESS_DATA_EC, 215
RTPS_LOG_PROCESS_GAP_EC, 215
RTPS_LOG_PROCESS_HEARTBEAT_BATCH_EC, 219
RTPS_LOG_PROCESS_HEARTBEAT_EC, 215
RTPS_LOG_READER_SEND_ACKNACK_EC, 218
RTPS_LOG_READER_UNSUPPORTED_EC, 213
RTPS_LOG_RECEIVE_EC, 213
RTPS_LOG_REQUEST_EC, 214
RTPS_LOG_RETURN_LOAN_EC, 214
RTPS_LOG_ROUTE_PACKET_EC, 213
RTPS_LOG_SEND_EC, 213
RTPS_LOG_SEND_HEARTBEAT_EC, 214
RTPS_LOG_SEQ_INIT_EC, 218
RTPS_LOG_SEQ_LEN_EC, 218
RTPS_LOG_SEQ_MAX_EC, 218
RTPS_LOG_SEQ_NUM_OUT_OF_RANGE_EC, 215
RTPS_LOG_SERIALIZE_APP_ID_EC, 216
RTPS_LOG_SERIALIZE_HOST_ID_EC, 216
RTPS_LOG_SERIALIZE_INSTANCE_ID_EC, 216
RTPS_LOG_SERIALIZE_OBJECT_ID_EC, 216
RTPS_LOG_SET_GAP_EC, 217
RTPS_LOG_SHIFT_BITMAP_EC, 214
RTPS_LOG_SHIFT_SN_OUT_OF_RANGE_EC, 216
RTPS_LOG_STALE_ACK_EPOCH__EC, 214
RTPS_LOG_STALE_HB_EPOCH_EC, 214
RTPS_LOG_STATUS_CHANGE_EC, 217
RTPS_LOG_TIMER_DELETE_TIMEOUT_EC, 217
RTPS_LOG_UNKNOWN_SUBMESSAGE_EC, 215
RTPS_LOG_UNSUPPORTED_INFO_REPLY_EC, 216
RTPS_LOG_UNSUPPORTED_INFO_REPLY_IP4_EC, 216
RTPS_LOG_UNSUPPORTED_INFO_SRC_EC, 216
RTPS_LOG_UNSUPPORTED_INTERFACE_EC, 215
RTPS_LOG_UNSUPPORTED_MANDTORY_HDR_EXT_PID_EC, 220
RTPS_LOG_WINDOW_ADVANCE_EC, 218
RTPS_LOG_WINDOW_ADVANCE_UNACKED_EC, 218
RTPS_LOG_WINDOW_INSERT_EC, 217
RTPS_LOG_WINDOW_POOL_ALLOC_EC, 217
RTPS_LOG_WRITER_REQUEST_SENT_HISTORY_EC, 217
RTPS Transport API, 133
allow_builtin_override, 136
builtin_checksum128_class, 135
builtin_checksum32_class, 135
builtin_checksum64_class, 135
checksum, 136
checksum128, 135
checksum32, 135
checksum64, 135
checksum_calculate, 135
checksum_tx_mode, 136
class_id, 135
context, 135
RTPS_CHECKSUM_TXMODE_OMG, 135
RTPS_CHECKSUM_TXMODE_RTICRC32, 135
RTPS_Checksum_T, 134
RTPS_ChecksumCalculate_T, 134
RTPS_ChecksumClass_T, 134
RTPS_ChecksumTxMode_T, 134
RTPS_INTERFACE_FACTORY_DEFAULT, 136
RTPS_InterfaceFactory_get_interface, 135
RTPS_InterfaceFactoryProperty_T, 134
RTPS_CHECKSUM_TXMODE_OMG
RTPS Transport API, 135
RTPS_CHECKSUM_TXMODE_RTICRC32
RTPS Transport API, 135
RTPS_Checksum, 459
RTPS_Checksum_T
RTPS Transport API, 134
RTPS_ChecksumCalculate_T
RTPS Transport API, 134
RTPS_ChecksumClass, 459
RTPS_ChecksumClass_T
RTPS Transport API, 134
RTPS_ChecksumProperty, 460
RTPS_ChecksumTxMode_T
RTPS Transport API, 134
RTPS_INTERFACE_FACTORY_DEFAULT
RTPS Transport API, 136
RTPS_InterfaceFactory_get_interface
RTPS Transport API, 135
RTPS_InterfaceFactoryProperty, 460
RTPS_InterfaceFactoryProperty_T
RTPS Transport API, 134
RTPS_LOG_ACK_EC
RTPS, 214
RTPS_LOG_ALLOCATE_EC
RTPS, 211
RTPS_LOG_ASSERT_PEER_EC
RTPS, 214
RTPS_LOG_ASSERT_TRANSPORT_EC
RTPS, 214
RTPS_LOG_BAD_PARAMETER_EC

RTPS, 213
RTPS_LOG_BITCOUNT_OUT_OF_BOUNDS_EC
RTPS, 215
RTPS_LOG_BITMAP_SET_BIT_EC
RTPS, 216
RTPS_LOG_BITMAP_SHIFT_EC
RTPS, 218
RTPS_LOG_BUFFERPOOL_DELETE_EC
RTPS, 217
RTPS_LOG_CHECKSUM_CHECKSUM_CALCULATION_FAILED_EC
RTPS, 220
RTPS_LOG_CHECKSUM_CHECKSUM_DESERIALIZE_ERROR_EC
RTPS, 220
RTPS_LOG_CHECKSUM_CHECKSUM_ERROR_EC
RTPS, 220
RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM128_EC
RTPS, 220
RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM32_EC
RTPS, 220
RTPS_LOG_CHECKSUM_ILLEGAL_OVERRIDE_CHECKSUM64_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INCONSISTENT_LENGTH_EC
RTPS, 219
RTPS_LOG_CHECKSUM_INVALID_CRC32_FLAGS_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM128_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM256_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM32_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INVALID_CUSTOM64_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN128_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN32_EC
RTPS, 220
RTPS_LOG_CHECKSUM_INVALID_OVERRIDE_BUILTIN64_EC
RTPS, 220
RTPS_LOG_CHECKSUM_LENGTH_DESERIALIZE_ERROR_EC
RTPS, 219
RTPS_LOG_CHECKSUM_MISSING_CHECKSUM_LENGTH_EC
RTPS, 219
RTPS_LOG_CHECKSUM_UNSUPPORTED_EC
RTPS, 219
RTPS_LOG_CREATE_BIND_TABLE_EC
RTPS, 217
RTPS_LOG_CREATE_BUFFER_POOL_EC
RTPS, 218
RTPS_LOG_CREATE_EVENT_TIMER_EC
RTPS, 215
RTPS_LOG_CREATE_HB_EVENT_EC
RTPS, 215
RTPS_LOG_CREATE_PEERS_INDEX_EC
RTPS, 218
RTPS_LOG_CREATE_ROUTE_TABLE_EC
RTPS, 211
RTPS_LOG_DATA_ALREADY_RECEIVED_EC
RTPS, 215
RTPS_LOG_DATA_OUT_OF_RANGE_EC
RTPS, 215
RTPS_LOG_DB_CREATE_BIND_ENTRY_EC
RTPS, 212
RTPS_LOG_DB_CREATE_BIND_PEER_INDEX_EC
RTPS, 212
RTPS_LOG_DB_CREATE_DIRECT_INDEX_EC
RTPS, 219
RTPS_LOG_DB_CREATE_EXT_BIND_ENTRY_EC
RTPS, 216
RTPS_LOG_DB_CREATE_GROUP_INDEX_EC
RTPS, 219
RTPS_LOG_DB_CREATE_ROUTE_ENTRY_EC
RTPS, 212
RTPS_LOG_DB_CREATE_ROUTE_PEER_INDEX_EC
RTPS, 211
RTPS_LOG_DB_CREATE_ROUTE_XPORT_INDEX_EC
RTPS, 211
RTPS_LOG_DB_CREATE_SELECTED_GROUP_INDEX_EC
RTPS, 219
RTPS_LOG_DB_DELETE_EXTERNAL_BIND_ENTRY_EC
RTPS, 213
RTPS_LOG_DB_DELETE_ROUTE_ENTRY_EC
RTPS, 212
RTPS_LOG_DB_INSERT_BIND_ENTRY_EC
RTPS, 212
RTPS_LOG_DB_INSERT_ROUTE_ENTRY_EC
RTPS, 212
RTPS_LOG_DB_LOCK_EC
RTPS, 217
RTPS_LOG_DB_REMOVE_BIND_ENTRY_EC
RTPS, 212
RTPS_LOG_DB_REMOVE_EXTERNAL_BIND_ENTRY_EC
RTPS, 212
RTPS_LOG_DB_REMOVE_ROUTE_ENTRY_EC
rtps_log.h, 535
RTPS_LOG_DB_SELECT_ALL_EC
RTPS, 212
RTPS_LOG_DB_SELECT_MATCH_EC
RTPS, 212

RTPS_LOG_DB_SELECT_RANGE_EC
RTPS, 212
RTPS_LOG_DB_UNLOCK_EC
RTPS, 217
RTPS_LOG_DELETE_BUFFER_POOL_EC
RTPS, 218
RTPS_LOG_DELETE_INDEX_EC
RTPS, 216
RTPS_LOG_DELETE_INDEXER_EC
RTPS, 219
RTPS_LOG_DELETE_TABLE_EC
RTPS, 217
RTPS_LOG_DESERIALIZE_APP_ID_EC
RTPS, 216
RTPS_LOG_DESERIALIZE_HOST_ID_EC
RTPS, 216
RTPS_LOG_DESERIALIZE_INSTANCE_ID_EC
RTPS, 216
RTPS_LOG_DESERIALIZE_OBJECT_ID_EC
RTPS, 216
RTPS_LOG_DIRECT_SEND_EC
RTPS, 217
RTPS_LOG_EXCEEDED_LIMIT_PEERS_EC
RTPS, 213
RTPS_LOG_EXCEEDED_LIMIT_TRANSPORTS_EC
RTPS, 213
RTPS_LOG_FINALIZE_INTERFACE_EC
RTPS, 218
RTPS_LOG_FIND_EXTERNAL_INTERFACE_EC
RTPS, 214
RTPS_LOG_FIRST_SN_GREATER_LAST_SN_EC
RTPS, 215
RTPS_LOG_FORWARD_UPSTREAM_EC
RTPS, 213
RTPS_LOG_FULL_SEND_WINDOW_EC
RTPS, 213
RTPS_LOG_FULLY_ACKED_READER_EC
RTPS, 214
RTPS_LOG_FUNC_UNSUPPORTED_EC
RTPS, 213
RTPS_LOG_GET_PEER_BUFFER_EC
RTPS, 219
RTPS_LOG_GET_WINDOW_BUFFER_EC
RTPS, 217
RTPS_LOG_INCONSISTENT_HDR_EXT_LENGTH_ERROR_EC
RTPS, 221
RTPS_LOG_INDEX_ADD_ENTRY_EC
RTPS, 212
RTPS_LOG_INDEXER_REMOVE_ENTRY_EC
RTPS, 218
RTPS_LOG_INITIALIZE_INTERFACE_EC
RTPS, 211
RTPS_LOG_INSERT_WINDOW_FULL_EC
RTPS, 217
RTPS_LOG_INTERFACE_MISMATCH_EC
RTPS, 213
RTPS_LOG_INTERFACE_NOT_ENABLED_EC
RTPS, 215
RTPS_LOG_INTF_MODE_UNDEF_EC
RTPS, 218
RTPS_LOG_INVALID_HDR_EXT_PID_ERROR_EC
RTPS, 221
RTPS_LOG_INVALID_HDR_EXT_PID_LIST_EC
RTPS, 221
RTPS_LOG_INVALID_PACKET_EC
RTPS, 215
RTPS_LOG_LOOKUP_EXT_BIND_ENTRY_EC
RTPS, 218
RTPS_LOG_LOOKUP_PEER_EC
RTPS, 218
RTPS_LOG_MESSAGE_EXCEED_LENGTH_ERROR_EC
RTPS, 221
RTPS_LOG_NEGATIVE_RESERVATION_COUNT_EC
RTPS, 219
RTPS_LOG_NETIO_PACKET_INSUFFICIENT_BUFFER_EC
RTPS, 219
RTPS_LOG_NETIO_PACKET_SET_HEAD_EC
RTPS, 214
RTPS_LOG_NETIO_PACKET_SET_TAIL_EC
RTPS, 213
RTPS_LOG_NONEXISTENT_BIND_EC
RTPS, 214
RTPS_LOG_NONEXISTENT_EXTERNAL_BIND_EC
RTPS, 214
RTPS_LOG_NONEXISTENT_ROUTE_EC
RTPS, 214
RTPS_LOG_NOT_ENABLED_EC
RTPS, 213
RTPS_LOG_PACKET_INITIALIZE_EC
RTPS, 217
RTPS_LOG_PROCESS_ACKNACK_EC
RTPS, 215
RTPS_LOG_PROCESS_DATA_BATCH_EC
RTPS, 219
RTPS_LOG_PROCESS_DATA_EC
RTPS, 215
RTPS_LOG_PROCESS_GAP_EC
RTPS, 215
RTPS_LOG_PROCESS_HEARTBEAT_BATCH_EC
RTPS, 219
RTPS_LOG_PROCESS_HEARTBEAT_EC
RTPS, 215
RTPS_LOG_READER_SEND_ACKNACK_EC
RTPS, 218
RTPS_LOG_READER_UNSUPPORTED_EC
RTPS, 213
RTPS_LOG_RECEIVE_EC
RTPS, 213
RTPS_LOG_REQUEST_EC
RTPS, 214
RTPS_LOG_RETURN_LOAN_EC
RTPS, 214
RTPS_LOG_ROUTE_PACKET_EC

RTPS, 213
RTPS_LOG_SEND_EC
RTPS, 213
RTPS_LOG_SEND_HEARTBEAT_EC
RTPS, 214
RTPS_LOG_SEQ_INIT_EC
RTPS, 218
RTPS_LOG_SEQ_LEN_EC
RTPS, 218
RTPS_LOG_SEQ_MAX_EC
RTPS, 218
RTPS_LOG_SEQ_NUM_OUT_OF_RANGE_EC
RTPS, 215
RTPS_LOG_SERIALIZE_APP_ID_EC
RTPS, 216
RTPS_LOG_SERIALIZE_HOST_ID_EC
RTPS, 216
RTPS_LOG_SERIALIZE_INSTANCE_ID_EC
RTPS, 216
RTPS_LOG_SERIALIZE_OBJECT_ID_EC
RTPS, 216
RTPS_LOG_SET_GAP_EC
RTPS, 217
RTPS_LOG_SHIFT_BITMAP_EC
RTPS, 214
RTPS_LOG_SHIFT_SN_OUT_OF_RANGE_EC
RTPS, 216
RTPS_LOG_STALE_ACK_EPOCH__EC
RTPS, 214
RTPS_LOG_STALE_HB_EPOCH_EC
RTPS, 214
RTPS_LOG_STATUS_CHANGE_EC
RTPS, 217
RTPS_LOG_TIMER_DELETE_TIMEOUT_EC
RTPS, 217
RTPS_LOG_UNKNOWN_SUBMESSAGE_EC
RTPS, 215
RTPS_LOG_UNSUPPORTED_INFO_REPLY_EC
RTPS, 216
RTPS_LOG_UNSUPPORTED_INFO_REPLY_IP4_EC
RTPS, 216
RTPS_LOG_UNSUPPORTED_INFO_SRC_EC
RTPS, 216
RTPS_LOG_UNSUPPORTED_INTERFACE_EC
RTPS, 215
RTPS_LOG_UNSUPPORTED_MANDTORY_HDR_E-
XT_PID_EC
RTPS, 220
RTPS_LOG_WINDOW_ADVANCE_EC
RTPS, 218
RTPS_LOG_WINDOW_ADVANCE_UNACKED_EC
RTPS, 218
RTPS_LOG_WINDOW_INSERT_EC
RTPS, 217
RTPS_LOG_WINDOW_POOL_ALLOC_EC
RTPS, 217
RTPS_LOG_WRITER_REQUEST_SENT_HISTORY_-
EC
RTPS, 217
reader_resource_limits
DDS_DataReaderQos, 365
reason
DDS_SampleLostStatus, 428
reception_timestamp
DDS_SampleInfo, 427
recv_thread
UDP_InterfaceFactoryProperty, 462
reda_dll.h, 524
reda_epoch.h, 524
reda_log.h, 525
reda_sequence_defn.h, 525
release
DDS_ProductVersion, 407
DDS_ProductVersion_t, 407
release_address
NETIO_DGRAM_InterfaceI, 450
ZCOPY_NotifUserInterfaceI, 464
reliability
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_PublicationBuiltinTopicData, 410
DDS_SubscriptionBuiltinTopicData, 432
remote_participant_allocation
DDS_DomainParticipantResourceLimitsQosPolicy,
382
remote_reader_allocation
DDS_DomainParticipantResourceLimitsQosPolicy,
382
remote_writer_allocation
DDS_DomainParticipantResourceLimitsQosPolicy,
382
require_crc
DDS_ChecksumProperty, 359
DDS_ChecksumProperty_t, 359
DDS_WireProtocolQosPolicy, 441
reserve_address
ZCOPY_NotifUserInterfaceI, 464
resolve_address
NETIO Interface API, 194
ZCOPY_NotifUserInterfaceI, 464
resource_limits
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
DDS_DomainParticipantFactoryQos, 378
DDS_DomainParticipantQos, 380
Return Codes, 236
DDS_RETCODE_ALREADY_DELETED, 237
DDS_RETCODE_BAD_PARAMETER, 236
DDS_RETCODE_ERROR, 236
DDS_RETCODE_ILLEGAL_OPERATION, 237
DDS_RETCODE_IMMUTABLE_POLICY, 237
DDS_RETCODE_INCONSISTENT_POLICY, 237
DDS_RETCODE_NO_DATA, 237
DDS_RETCODE_NOT_ENABLED, 237
DDS_RETCODE_OK, 236
DDS_RETCODE_OUT_OF_RESOURCES, 237

DDS_RETCODE_PRECONDITION_NOT_MET, 237
DDS_RETCODE_TIMEOUT, 237
DDS_RETCODE_UNSUPPORTED, 236
DDS_ReturnCode_t, 236
revision
DDS_ProductVersion, 407
DDS_ProductVersion_t, 407
rh_sm_dll.h, 525
rh_sm_history.h, 526
rh_sm_log.h, 526
RHSM_LOG_HISTORY_OBJECT, 528
RHSM_LOG_INSTANCEHANDLE_OBJECT, 528
RHSM_LOG_KEY_OBJECT, 528
RHSM_LOG_KEYINDEXPOOL_OBJECT, 528
RHSM_LOG_KEYPOOL_OBJECT, 528
RHSM_LOG_OWNER_OBJECT, 528
RHSM_LOG_REMOVELIST_OBJECT, 528
RHSM_LOG_RW_OBJECT, 528
RHSM_LOG_RWINDEX_OBJECT, 528
RHSM_LOG_RWPOOL_OBJECT, 528
RHSM_LOG_SAMPLEINFO_OBJECT, 528
RHSM_LOG_SAMPLEPOOL_OBJECT, 528
RHSM_LOG_SAMPLEPTPOOL_OBJECT, 528
rt_dll.h, 528
rt_log.h, 529
rtps_app_id
DDS_WireProtocolQosPolicy, 440
rtps_config.h, 529
rtps_dll.h, 529
rtps_host_id
DDS_WireProtocolQosPolicy, 440
rtps_instance_id
DDS_WireProtocolQosPolicy, 440
rtps_log.h, 529
RTPS_LOG_DB_REMOVE_ROUTE_ENTRY_EC, 535
rtps_object_id
DDS_DataReaderProtocolQosPolicy, 364
DDS_DataWriterProtocolQosPolicy, 370
rtps_protocol_version
DDS_ParticipantBuiltinTopicData, 403
rtps_reliable_reader
DDS_DataReaderProtocolQosPolicy, 364
rtps_reliable_writer
DDS_DataWriterProtocolQosPolicy, 370
rtps_rtps_impl.h, 535
rtps_vendor_id
DDS_ParticipantBuiltinTopicData, 403
rtps_well_known_ports
DDS_WireProtocolQosPolicy, 440

SYSTEM_RESOURCE_LIMITS, 265
DDS_SYSTEMRESOURCELIMITS_QOS_POLICY_NAME, 265
Sample States, 102
DDS_ANY_SAMPLE_STATE, 103
DDS_NOT_READ_SAMPLE_STATE, 102
DDS_READ_SAMPLE_STATE, 102
DDS_SampleStateKind, 102
DDS_SampleStateMask, 102
sample_info
DDS_SampleLostStatus, 428
sample_state
DDS_SampleInfo, 426
sec
DDS_Duration_t, 384
DDS_Time_t, 435
OSAPI_SystemTime, 457
send
NETIO Interface API, 194
ZCOPY_NotifUserInterfaceI, 465
Sequence Number Support, 235
Sequence Support, 323
DDS_SEQUENCE_INITIALIZER, 323
FooSeq_copy, 327
FooSeq_copy_w_max, 327
FooSeq_get_length, 326
FooSeq_get_maximum, 324
FooSeq_get_reference, 327
FooSeq_initialize, 324
FooSeq_set_length, 326
FooSeq_set_maximum, 324
FooSeq_set_maximum_w_max, 325
sequence_number
DDS_ReliableSampleUnacknowledgedStatus, 415
serialize_data
NDDS_Type_Plugin, 446
size
OSAPI_SharedMemorySegmentHeader, 456
source_timestamp
DDS_SampleInfo, 426
source_timestamp_tolerance
DDS_DestinationOrderQosPolicy, 375
stack_size
OSAPI_ThreadProperty, 458
start_timer
OSAPI_SystemI, 456
Status Kinds, 238
DDS_DATA_AVAILABLE_STATUS, 244
DDS_DATA_ON_READERS_STATUS, 244
DDS_INCONSISTENT_TOPIC_STATUS, 242
DDS_INSTANCE_REPLACED_STATUS, 245
DDS_LIVELINESS_CHANGED_STATUS, 244
DDS_LIVELINESS_LOST_STATUS, 244
DDS_OFFERED_DEADLINE_MISSED_STATUS, 243
DDS_OFFERED_INCOMPATIBLE_QOS_STATUS, 243
DDS_PUBLICATION_MATCHED_STATUS, 245
DDS_RELIABLE_READER_ACTIVITY_CHANGED_STATUS, 245
DDS_REQUESTED_DEADLINE_MISSED_STATUS, 243
DDS_REQUESTED_INCOMPATIBLE_QOS_STATUS, 243
DDS_SAMPLE_LOST_STATUS, 243

DDS_SAMPLE_REJECTED_STATUS, 244
DDS_STATUS_MASK_ALL, 242
DDS_STATUS_MASK_NONE, 242
DDS_SUBSCRIPTION_MATCHED_STATUS, 245
DDS_StatusKind, 242
DDS_StatusMask, 242
String Support, 329
DDS_String_alloc, 330
DDS_String_cmp, 330
DDS_String_dup, 330
DDS_String_length, 331
DDS_String_ncmp, 331
DDS_Wstring_cmp, 331
Subscriber, 73
DDS_DATAREADER_QOS_DEFAULT, 79
DDS_Subscriber, 74
DDS_Subscriber_as_entity, 76
DDS_Subscriber_create_datareader, 77
DDS_Subscriber_get_default_datareader_qos, 76
DDS_Subscriber_get_participant, 79
DDS_Subscriber_lookup_datareader, 78
DDS_Subscriber_set_default_datareader_qos, 77
DDS_SubscriberListener_DataOnReadersCallback, 75
DDS_SubscriberListener_INITIALIZER, 74
DDS_SubscriberQos_INITIALIZER, 74
DDS_SubscriberQos_copy, 75
DDS_SubscriberQos_initialize, 75
DDS_SubscriberQos_is_equal, 76
Subscription Built-in Topic, 229
DDS_SUBSCRIPTION_TOPIC_NAME, 229
DDS_SubscriptionBuiltinTopicData_copy, 230
DDS_SubscriptionBuiltinTopicData_initialize, 229
DDS_SubscriptionBuiltinTopicData_is_equal, 230
Subscription Module, 71

T
dds_c_infrastructure.h, 479, 480
TRANSPORT, 272
DDS_TRANSPORT_QOS_POLICY_NAME, 272
Time Support, 232
DDS_DURATION_INFINITE, 233
DDS_DURATION_INFINITE_NSEC, 233
DDS_DURATION_INFINITE_SEC, 233
DDS_DURATION_ZERO, 233
DDS_DURATION_ZERO_NSEC, 233
DDS_DURATION_ZERO_SEC, 233
DDS_TIME_INVALID, 233
DDS_TIME_INVALID_NSEC, 232
DDS_TIME_INVALID_SEC, 232
DDS_TIME_ZERO, 232
Topic, 36, 37
DDS_InconsistentTopicStatus_INITIALIZER, 38
DDS_Topic, 38
DDS_Topic_as_entity, 40
DDS_Topic_as_topicdescription, 40
DDS_Topic_get_inconsistent_topic_status, 40
DDS_Topic_narrow, 40
DDS_TopicDescription, 38
DDS_TopicDescription_get_name, 39
DDS_TopicDescription_get_participant, 40
DDS_TopicDescription_get_type_name, 39
DDS_TopicListener_INITIALIZER, 38
DDS_TopicListener_InconsistentTopicCallback, 39
topic_name
DDS_PublicationBuiltinTopicData, 409
DDS_SubscriptionBuiltinTopicData, 432
total_count
DDS_DataReaderInstanceReplacedStatus, 360
DDS_LivelinessLostStatus, 393
DDS_OfferedDeadlineMissedStatus, 397
DDS_OfferedIncompatibleQosStatus, 398
DDS_PublicationMatchedStatus, 411
DDS_RequestedDeadlineMissedStatus, 415
DDS_RequestedIncompatibleQosStatus, 416
DDS_SampleLostStatus, 428
DDS_SampleRejectedStatus, 429
DDS_SubscriptionMatchedStatus, 433
total_count_change
DDS_DataReaderInstanceReplacedStatus, 360
DDS_LivelinessLostStatus, 393
DDS_OfferedDeadlineMissedStatus, 397
DDS_OfferedIncompatibleQosStatus, 398
DDS_PublicationMatchedStatus, 411
DDS_RequestedDeadlineMissedStatus, 415
DDS_RequestedIncompatibleQosStatus, 416
DDS_SampleLostStatus, 428
DDS_SampleRejectedStatus, 429
DDS_SubscriptionMatchedStatus, 433
transport
DDS_DataReaderQos, 365
DDS_DataWriterQos, 372
transports
DDS_DomainParticipantQos, 380
type_name
DDS_PublicationBuiltinTopicData, 409
DDS_SubscriptionBuiltinTopicData, 432

UDP_InterfaceFactoryProperty, 460
allow_interface, 461
deny_interface, 461
if_table, 462
is_default_interface, 462
max_message_size, 461
max_receive_buffer_size, 461
max_send_buffer_size, 461
max_send_message_size, 461
multicast_interface, 462
multicast_ttl, 461
recv_thread, 462
UDP_InterfaceTableEntry, 462
address, 462
flags, 462
ifname, 462
netmask, 462
UDP_InterfaceTableEntrySeq, 463
UDP_LOG_ALLOC_EC
NETIO, 204

UDP_LOG_CURSOR_ERROR_EC
NETIO, 205
UDP_LOG_DROPGRP_EC
NETIO, 204
UDP_LOG_FINALIZE_EC
NETIO, 203
UDP_LOG_GET_BUF_ALLOC_EC
NETIO, 204
UDP_LOG_GET_BUF_SIZE_EC
NETIO, 204
UDP_LOG_GET_IFLIST_EC
NETIO, 204
UDP_LOG_GET_LENGTH_EC
NETIO, 204
UDP_LOG_GET_NAMEINFO_EC
NETIO, 204
UDP_LOG_GETHOSTNAME_EC
NETIO, 202
UDP_LOG_INCONSISTENT_MAX_MESSSAGE_SIZ-
E_EC
NETIO, 205
UDP_LOG_INITIALIZE_FAILED_EC
NETIO, 205
UDP_LOG_LOADLIB_EC
NETIO, 204
UDP_LOG_MULTICAST_ENABLE_EC
NETIO, 204
UDP_LOG_MULTICAST_NOT_ENABLED_EC
NETIO, 205
UDP_LOG_NAT_DELETE_EC
NETIO, 203
UDP_LOG_PACKET_FWD_EC
NETIO, 203
UDP_LOG_PACKET_HEAD_EC
NETIO, 203
UDP_LOG_PACKET_INIT_EC
NETIO, 202
UDP_LOG_PORT_ADD_EC
NETIO, 203
UDP_LOG_PORT_ALLOC_EC
NETIO, 203
UDP_LOG_PORT_ENTRY_EC
NETIO, 203
UDP_LOG_PORT_NOT_FOUND_EC
NETIO, 205
UDP_LOG_PORT_THREAD_EC
NETIO, 203
UDP_LOG_PROPERTY_FINALIZE_EC
NETIO, 204
UDP_LOG_RECORD_EC
NETIO, 204
UDP_LOG_RECV_ERROR_EC
NETIO, 205
UDP_LOG_SEND_ERROR_EC
NETIO, 205
UDP_LOG_SEND_PING_EC
NETIO, 204
UDP_LOG_SET_LENGTH_EC
NETIO, 204
UDP_LOG_SOCKET_ADDGRP_EC
NETIO, 203
UDP_LOG_SOCKET_BIND_EC
NETIO, 203
UDP_LOG_SOCKET_CLOSE_EC
NETIO, 203
UDP_LOG_SOCKET_CREATE_EC
NETIO, 202
UDP_LOG_SOCKET_IFFLAGS_EC
NETIO, 203
UDP_LOG_SOCKET_IFLIST_EC
NETIO, 203
UDP_LOG_SOCKET_INIT_EC
NETIO, 202
UDP_LOG_SOCKET_REUSE_PORT_EC
NETIO, 203
UDP_LOG_SOCKET_RXBUF_EC
NETIO, 203
UDP_LOG_SOCKET_SET_MCAST_EC
NETIO, 202
UDP_LOG_SOCKET_SET_MCASTIF_EC
NETIO, 202
UDP_LOG_SOCKET_SETFD_EC
NETIO, 202
UDP_LOG_SOCKET_SNDBUF_EC
NETIO, 202
UDP_LOG_SOCKET_TTL_EC
NETIO, 202
UDP_LOG_TRANSPORT_NO_INTERFACES_EC
NETIO, 205
UDP_LOG_UDP_CREATE_INDEX_EC
NETIO, 204
UDP_LOG_UDP_CREATE_TABLE_EC
NETIO, 204
UDP_LOG_WINSOCK_INCOMPATIBLE_EC
NETIO, 205
UDP_LOG_WSA_STARTUP_EC
NETIO, 205
unacknowledged_count
DDS_ReliableSampleUnacknowledgedStatus, 415
unbind
ZCOPY_NotifUserInterfaceI, 464
unicast_locator
DDS_PublicationBuiltinTopicData, 410
DDS_SubscriptionBuiltinTopicData, 433
User Data Type Support, 42
DDS_HANDLE_NIL, 45
DDS_InstanceHandle_equals, 45
DDS_InstanceHandle_is_nil, 45
DDS_InstanceHandle_t, 43
DDS_TYPESUPPORT_C, 43
FooTypeSupport_create_data, 44
FooTypeSupport_register_type, 44
User notification mechanism interface, 348
ZCOPY_NotifUserInterface_add_routeFunc, 349
ZCOPY_NotifUserInterface_bindFunc, 350
ZCOPY_NotifUserInterface_createFunc, 348

ZCOPY_NotifUserInterface_delete_routeFunc, 349
ZCOPY_NotifUserInterface_notify_portFunc, 350
ZCOPY_NotifUserInterface_release_addressFunc, 349
ZCOPY_NotifUserInterface_reserve_address-Func, 349
ZCOPY_NotifUserInterface_sendFunc, 350
ZCOPY_NotifUserInterface_unbindFunc, 350
User-Defined Types, 109
DDS_InstanceId_t, 110
NDDS_PluginHelper_CreateManagedPoolFunc, 111
NDDS_PluginHelper_CreateTypedDataReader-Func, 111
NDDS_PluginHelper_CreateTypedDataWriter-Func, 111
NDDS_PluginHelper_DeleteManagedPoolFunc, 111
NDDS_PluginHelper_DeleteTypedDataReader-Func, 111
NDDS_PluginHelper_DeleteTypedDataWriterFunc, 111
NDDS_PluginHelper_GetKeyKindFunc, 111
NDDS_PluginHelper_InstanceToKeyHashFunc, 111
NDDS_PluginHelper_ManagedPoolDiscardLoan-Func, 111
NDDS_PluginHelper_ManagedPoolGetLoanFunc, 111
NDDS_PluginHelper_ManagedPoolGetSampleID-Func, 111
NDDS_PluginHelper_NeedsOpaqueSamples, 112
NDDS_TYPEPLUGIN_NO_KEY, 112
NDDS_TYPEPLUGIN_USER_KEY, 112
NDDS_Type_CopySampleFunc, 111
NDDS_Type_CreateSampleFunc, 110
NDDS_Type_DeleteSampleFunc, 111
NDDS_Type_GetSerializedSizeFunc, 110
NDDS_Type_PluginVersion, 110
NDDS_TypePluginKeyKind, 112
PluginHelper_get_key_kind, 112
PluginHelper_instance_to_keyhash, 112
user_intf
ZCOPY_NotifInterfaceFactoryProperty, 463
user_multicast_port_offset
DDS_RtpsWellKnownPorts_t, 424
user_property
ZCOPY_NotifInterfaceFactoryProperty, 463
user_traffic
DDS_DomainParticipantQos, 380
user_unicast_port_offset
DDS_RtpsWellKnownPorts_t, 424

valid_data
DDS_SampleInfo, 427
value
DDS_BuiltinTopicKey_t, 358
DDS_GUID_t, 386
DDS_OwnershipStrengthQosPolicy, 402
NETIO_Address, 448
NETIO_AddressUInt32, 449
vendorId
DDS_VendorId, 438
DDS_VendorId_t, 438
View States, 104
DDS_ANY_VIEW_STATE, 105
DDS_NEW_VIEW_STATE, 105
DDS_NOT_NEW_VIEW_STATE, 105
DDS_ViewStateKind, 104
DDS_ViewStateMask, 104
view_state
DDS_SampleInfo, 426

WH, 336
WHSM_LOG_INVALID_INDEX_OBJECT_EC, 337
WHSM_LOG_KEEP_ALL_HISTORY_NOT_SUP-PORTED_EC, 336
WHSM_LOG_MAX_SAMPLES_TOO_SMALL_E-C, 336
WHSM_LOG_NO_PROPERTY_EC, 336
WHSM_LOG_OBJECT_ALLOCATE_EC, 336
WHSM_LOG_OBJECT_DELETE_EC, 336
WHSM_LOG_OBJECT_INDEX_EC, 336
WHSM_LOG_UNLIMITED_HISTORY_NOT_SUP-PORTED_EC, 336
WHSM Writer History API, 131
WHSM_HistoryFactory_get_interface, 131
WHSM_HistoryFactory_get_interface
WHSM Writer History API, 131
WHSM_LOG_HISTORY_OBJECT
wh_sm_log.h, 537
WHSM_LOG_HISTORYINDEX_OBJECT
wh_sm_log.h, 537
WHSM_LOG_INVALID_INDEX_OBJECT_EC
WH, 337
WHSM_LOG_KEEP_ALL_HISTORY_NOT_SUPPOR-TED_EC
WH, 336
WHSM_LOG_KEY_OBJECT
wh_sm_log.h, 537
WHSM_LOG_KEYINDEX_OBJECT
wh_sm_log.h, 537
WHSM_LOG_KEYPOOL_OBJECT
wh_sm_log.h, 537
WHSM_LOG_MAX_SAMPLES_TOO_SMALL_EC
WH, 336
WHSM_LOG_NO_PROPERTY_EC
WH, 336
WHSM_LOG_OBJECT_ALLOCATE_EC
WH, 336
WHSM_LOG_OBJECT_DELETE_EC
WH, 336
WHSM_LOG_OBJECT_INDEX_EC
WH, 336
WHSM_LOG_SAMPLE_OBJECT
wh_sm_log.h, 537
WHSM_LOG_SAMPLEINDEX_OBJECT

wh_sm_log.h, 537
WHSM_LOG_SAMPLEPOOL_OBJECT
wh_sm_log.h, 537
WHSM_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC
WH, 336
WHSQ, 352
WHSQ_LOG_COMMIT_FAILED_EC, 353
WHSQ_LOG_CONFIGURATION_EC, 353
WHSQ_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC, 352
WHSQ_LOG_MAX_SAMPLES_TOO_SMALL_EC, 352
WHSQ_LOG_NO_DDS_FACTORY_EC, 353
WHSQ_LOG_OBJECT_ALLOCATE_EC, 352
WHSQ_LOG_OBJECT_DELETE_EC, 353
WHSQ_LOG_PURGE_FAILED_EC, 352
WHSQ_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC, 352
WHSQ_LOG_WH_CREATE_FAILED_EC, 352
WHSQ_HistoryFactory_get_interface
netio_zcopy_whsq.h, 520
WHSQ_LOG_COMMIT_FAILED_EC
WHSQ, 353
WHSQ_LOG_CONFIGURATION_EC
WHSQ, 353
WHSQ_LOG_HISTORY_OBJECT
netio_zcopy_whsq_log.h, 520
WHSQ_LOG_KEEP_ALL_HISTORY_NOT_SUPPORTED_EC
WHSQ, 352
WHSQ_LOG_MAX_SAMPLES_TOO_SMALL_EC
WHSQ, 352
WHSQ_LOG_NO_DDS_FACTORY_EC
WHSQ, 353
WHSQ_LOG_OBJECT_ALLOCATE_EC
WHSQ, 352
WHSQ_LOG_OBJECT_DELETE_EC
WHSQ, 353
WHSQ_LOG_PURGE_FAILED_EC
WHSQ, 352
WHSQ_LOG_UNLIMITED_HISTORY_NOT_SUPPORTED_EC
WHSQ, 352
WHSQ_LOG_WH_CREATE_FAILED_EC
WHSQ, 352
WIRE_PROTOCOL, 266
DDS_CHECKSUM_AUTO, 267
DDS_CHECKSUM_BUILTIN128, 267
DDS_CHECKSUM_BUILTIN32, 267
DDS_CHECKSUM_BUILTIN64, 267
DDS_CHECKSUM_NONE, 267
DDS_ChecksumKind_t, 267
DDS_ChecksumKindMask_t, 267
DDS_INTEROPERABLE_RTPS_WELL_KNOWN_PORTS, 268
DDS_RTI_BACKWARDS_COMPATIBLE_RTPS_WELL_KNOWN_PORTS, 268
DDS_WIREPROTOCOL_QOS_POLICY_NAME, 268
wh_sm_dll.h, 536
wh_sm_history.h, 536
wh_sm_log.h, 536
WHSM_LOG_HISTORY_OBJECT, 537
WHSM_LOG_HISTORYINDEX_OBJECT, 537
WHSM_LOG_KEY_OBJECT, 537
WHSM_LOG_KEYINDEX_OBJECT, 537
WHSM_LOG_KEYPOOL_OBJECT, 537
WHSM_LOG_SAMPLE_OBJECT, 537
WHSM_LOG_SAMPLEINDEX_OBJECT, 537
WHSM_LOG_SAMPLEPOOL_OBJECT, 537
write_buffer
OSAPI_LogProperty, 455
writer_resource_limits
DDS_DataWriterQos, 372

ZCOPY_NotifInterface_receive
Notification Interface, 346
ZCOPY_NotifInterfaceFactory_register
Notification Interface, 347
ZCOPY_NotifInterfaceFactory_unregister
Notification Interface, 347
ZCOPY_NotifInterfaceFactoryProperty, 463
max_samples_per_notif, 463
user_intf, 463
user_property, 463
ZCOPY_NotifInterfaceFactoryProperty_INITIALIZER
Notification Interface, 346
ZCOPY_NotifUserInterface_add_routeFunc
User notification mechanism interface, 349
ZCOPY_NotifUserInterface_bindFunc
User notification mechanism interface, 350
ZCOPY_NotifUserInterface_createFunc
User notification mechanism interface, 348
ZCOPY_NotifUserInterface_delete_routeFunc
User notification mechanism interface, 349
ZCOPY_NotifUserInterface_notify_portFunc
User notification mechanism interface, 350
ZCOPY_NotifUserInterface_release_addressFunc
User notification mechanism interface, 349
ZCOPY_NotifUserInterface_reserve_addressFunc
User notification mechanism interface, 349
ZCOPY_NotifUserInterface_sendFunc
User notification mechanism interface, 350
ZCOPY_NotifUserInterface_unbindFunc
User notification mechanism interface, 350
ZCOPY_NotifUserInterfaceI, 463
add_route, 464
bind, 464
create_instance, 464
delete_route, 464
get_route_table, 464
notify_recv_port, 465
release_address, 464
reserve_address, 464
resolve_address, 464
send, 465

unbind, 464
Zero Copy v2 API, 145
NDDS_Transport_ZeroCopy_initialize, 145
NETIO_DEFAULT_NOTIF_NAME, 146